Time Series Analysis using Explainable AI

Francesca Silvio

Supervisor: Prof Matthew Montebello

Co-supervisor(s): Dr Vincent Vella

Department of Computer Information Systems

Faculty of ICT

University of Malta

May 2022

Submitted in partial fulfilment of the requirements for the degree of Master of Science in Artificial Intelligence

To my family and friends

for their constant support throughout this academic journey.

Acknowledgements

I would like to thank Prof. Matthew Montebello and Dr. Vince Vella for their support, encouragement and patience from the beginning up until the end of this project. I would also like to thank my colleagues for giving me this opportunity and helping me along the way.

Abstract

In the last couple of years, great leaps have been made in the field of Machine Learning. Despite this, understanding how and why a machine learning model makes a decision is still a challenge faced by non-expert users, for which solutions are being actively developed. Moreover, studies on techniques which may be used to evaluate such solutions quantitatively are very scarce.

In this research, Explainable AI techniques are used on LSTM model predictions for forecasting customer depositing data using a time-series dataset which was created for the purpose of this study. An index was also developed in order to quantitatively evaluate the quality and stability of such predictions. In order to achieve this, raw data was extracted from a customer transactional database and analysed in order to create a suitable time-series dataset. This time-series was then applied to an LSTM model as well as a Naive model and an ARIMA model for benchmark purposes. Results showed that the ARIMA model outperformed the LSTM model in most cases. LSTM predictions were generated using a Monte Carlo simulation in order to get measures on prediction confidence. LIME and SHAP explanations were generated for these predictions.

The explanations were evaluated both quantitatively, by using the stability index created, as well was qualitatively through an evaluation by human domain experts. The quantitative evaluation performed concluded that SHAP explanations are generally more stable than LIME explanations and that there is no correlation between a prediction's accuracy and the stability index. Through the qualitative evaluation, it was found that the stability index helped increase the trust of the domain experts in the explanations and predictions.

Contents

List of Figures

List of Tables

List of Abbreviations

AI	Artificial Intelligence
ANN	Artificial Neural Network
ARIMA	Autoregressive Integrated Moving Average
BNN	Bayesian Neural Network
CNN	Convolutional Neural Networks
CSI	Coefficients Stability Index
EDA	Exploratory Data Analysis
GDPR	General Data Protection Regulation
GOC	Games of Chance
Grad-CAM	Gradientweighted Class Activation Mapping
GRU	Gated Recurrent Unit
KNN	K Nearest Neighbour
LIME	Local Interpretable Model-Agnostic Explanations
LSTM	Long Short-Term Memory Networks
MAD	Mean Absolute Deviation
MAE	Mean Absolute Error
MAPE	Mean Absolute Percentage Error
MCMC	Monte Carlo Simulations and Markov Chains method
MDA	Mean Decrease Accuracy
MSE	Mean Squared Error
RMSE	Root Mean Squared Error
RNN	Recurrent Neural Network
SARIMA	Seasonal ARIMA
SB	Sportsbook

SHAP SHapley Additive exPlanations

SPEC Stock-keepingoriented Prediction Error Costs

SVM Support Vector Machines

SVR Support Vector Regression

TG-LSTM Transformation-gated LSTM

UI User Interface

VAR Vector Autoregression

VSI Variable Stability Index

List of Definitions

Wallet The money a customer has on their account.

Wallet Transactions Transactions made using the wallet balance. Bets and wins are examples of such transactions.

Deposit Adding money to the customer wallet using an external payment method such as a credit-card.

Withdrawal Transferring money from the customer wallet to an external account using a payment method such as a bank transfer.

Payment Transactions Transactions which deposit or withdraw money from the customer wallet.

Chapter 1 - Introduction

1.1 Problem Definition

In any business dealing with customer payments, the ability to forecast when a customer is likely to make a payment would be very advantageous. This would help in several business aspects, from choosing target customers for marketing to making strategic business decisions. Such a system might also detect any anomalies which might help pinpoint any entities which are not working effectively as well as highlight improvements which could be made.
Such payment data can be transformed into a time-series, allowing one to be able to forecast the data at a future point in time. Such predictions may be made using a combination of Statistical Methods and Machine Learning.
Once a prediction has been made, one might wish to know why that prediction was made. This is a challenge when making use of Machine Learning models, which are typically referred to as being black-box models due to the difficulty in understanding why a prediction was made. Therefore, Explainable AI techniques must be used in order for the predictions made to have a level of transparency. It is also very important to find a way to evaluate the explanations produced through these Explainable AI techniques.

1.2 Motivation

Explainable AI is a relatively new concept, providing vast research opportunities. In this study, Explainable AI techniques will be used in order to interpret predictions made on a custom payments time-series dataset. Most research on Explainable AI lacks evaluation or only presents a qualitative evaluation [1]. Furthermore, studies on a quantitative evaluation technique for explainable AI

methods used on time-series forecasting models are even more scarce. Due to the vast and ubiquitous nature of time-series, quantitatively evaluating the explanation generated on such forecasts is very important [1].
Since outputs from Explainable AI techniques are ultimately aimed for use by human subjects, qualitative evaluations are very important. However a quantifiable method to evaluate how good or bad an explanation is would also be very useful.
The main motivation behind this study is to determine whether Explainable AI may be used to interpret forecasts on time-series payment data and then to go a step further by finding a method to evaluate the quality of the generated explanations.

1.3 Aim and Objectives

The aim of this study is to obtain accurate and explainable predictions on a customer′s depositing behaviour and evaluate the produced explanations such that well-informed business decisions can be made. Below are the objectives which have been set in order to reach the defined aim:

(O1) Perform Exploratory Data Analysis (EDA) and create a time series dataset based on this analysis by pre-processing and aggregating the data.

(O2) Configure a machine learning model for time-series forecasting.

(O3) Explore whether Explainable AI may be used to make model results more understandable and transparent.

(O4) Determine a method to evaluate the explainability results, analyse the results obtained, compare them to results from other studies and implement a system to show the results.

1.4 Approach

In order to fulfil the specified objectives, the planned approach may be split into eight steps:

1. Extract the required data from the database using SQL and create a time-series dataset from the raw data.
2. Perform EDA on the dataset, outputting visuals and aggregated data in order to identify trends in the data, correlation between features and spot anomalies amongst other things.
3. Use the results from the EDA to refine the time-series dataset.
4. Configure an LSTM model which uses the time-series dataset in order to make forecasts on customer deposit amounts.
5. Create naive and auto-arima models as benchmark models to compare results produced by these models to those forecast by the LSTM.
6. Use Monte-Carlo dropout, LIME and SHAP in order to create a layer of transparency for the LSTM forecasts.
7. Determine a method to quantitatively evaluate the explanations.
8. Create a web application showing the predictions and explanations in order to qualitatively evaluate the whole system.

1.5 Report Layout

The following chapter discusses literature on the background of the techniques used in this study and reviews similar studies. Chapter 3 details the steps taken in order to implement all the points outlined in the previous section whilst Chapter 4 discusses the results obtained through this implementation. In Chapter 5, these results are evaluated and compared to results obtained in similar studies. Chapter 6, the final chapter, discusses limitations, future improvements and the conclusions reached through this project.

Chapter 2 - Background and Literature Review

This chapter reviews the literature related to the areas being researched in this paper. The two main areas of research in this study are *Explainable AI* and *Time-Series Analysis*. Similar studies are also outlined and their results are analyzed.

2.1 Explainable AI Techniques

Throughout recent years, great advances have been made in the field of Artificial Intelligence (AI). It has been found that using machine learning as a method to solve certain problems is better than using statistical methods. There have also been cases where deep learning models have surpassed human performance, such as when playing strategic games like chess and GO [13]. However, despite these leaps in the field of Artificial Intelligence, the problem of explainability is one which is very predominant. When AI is used in sensitive environments or to make expensive decisions, it is imperative for the user of a machine learning model to understand what led to the output of a result. Challenging the idea that a machine learning model is a black-box and having some form of transparency will increase the trust in a model, especially if critical situations might be encountered [13]. Rojat et al. [1] explain the different purposes of Explainable AI, as shown in Figure 2.1.

In [14], Kalyanathaya et al. identify 5 reasons on why Explainable AI is so important :

1. Accountability : In order for the person using the model to take responsibility in making well-informed decisions, they should be able to know how a model ended up at that decision.

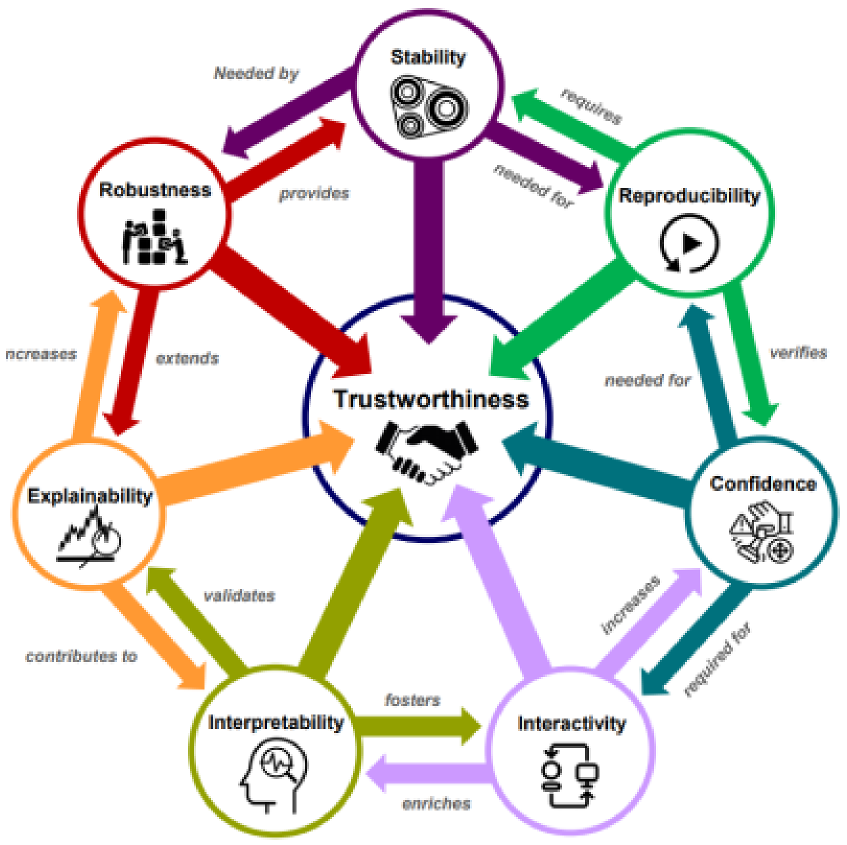


Figure 2.1: Purposes of Explainable AI which contribute to the trustworthiness of model predictions and their relationships, obtained from [1].

2. Reliance : Explainability is a tool which helps users gain trust by adding a level of transparency to the model's decision making. This is critical in certain sectors such as healthcare.
3. Compliance : Article 14 of the General Data Protection Regulation (GDPR) [15] states that when a company uses AI models for automated decision making, rationale, importance and consequences of such decisions must be available.
4. Performance : Transparency of a machine learning model will help the creator tune its parameters and therefore improve its performance.
5. Control : Model interperebality can help users identify vulnerabilities and flaws of a model.

There are two main types of Explainable AI - transparency design and post-hoc explanation. Transparency design aims to reveal how the model functions by attempting to understand model structure, single components and training algorithms. On the other hand, post-hoc explanations try to explain why a result was inferred [16]. This study will mainly focus on post-hoc methods, which have the huge advantage of being model agnostic, meaning that an explanation may be given irrelevant of the model used for training. Figure 2.2 shows a number of different types of post-hoc methods used in order to interpret black-box models [2].

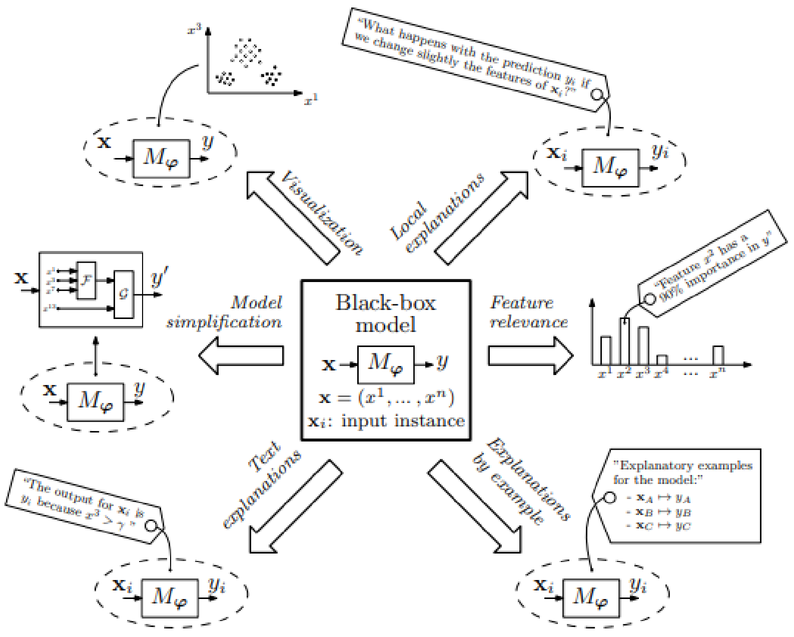

Black-box model
$\mathbf{x} = (x^1, \dots, x^n)$
$\mathbf{x}_i$: input instance
Visualization
Local explanations
"What happens with the prediction y_i if we change slightly the features of $\mathbf{x}_i$?"
Model simplification
Feature relevance
"Feature x^2 has a 90% importance in y"
Text explanations
"The output for $\mathbf{x}_i$ is y_i because $x^3 > \gamma$"
Explanations by example
"Explanatory examples for the model:"
- $\mathbf{x}_A \mapsto y_A$
- $\mathbf{x}_B \mapsto y_B$
- $\mathbf{x}_C \mapsto y_C$

Figure 2.2: Different post-hoc methods used to derive explanations from black-box models [2]

2.1.1 Local Interpretable Model-Agnostic Explanations (LIME)

One post-hoc explanation method is LIME, which aims to identify an "interpretable model over the interperetable representation that is locally faithful to the classifier" [17]. This is accomplished by building a local linear model on the predictions of a black-box model in order to identify the feature relevance of a model. Weights are assigned to each sample based on the closeness to the point of interest, which are then used to train another model which outputs the LIME explanation.

Since LIME is model-agnostic, it may be used on predictions coming from a wide range of different types of machine learning models, including random forests, support vector machines (SVM) and neural networks, which may use data in both text and image formats. Figure 2.3, obtained from [3], shows how an explanation on the predictions of tabular data is reached. Image A shows 2 possible predictions, represented by the different colours, given 2 features x_1 and x_2. Image B shows the instance of interest, represented by the yellow dot and the data sampled from a normal distribution, shown as the smaller dots. Image C shows that points closer to the instance of interest are given higher weights. The final image D shows the classifications of the local model, created using the random samples. The decision boundary is marked by the white line [3].

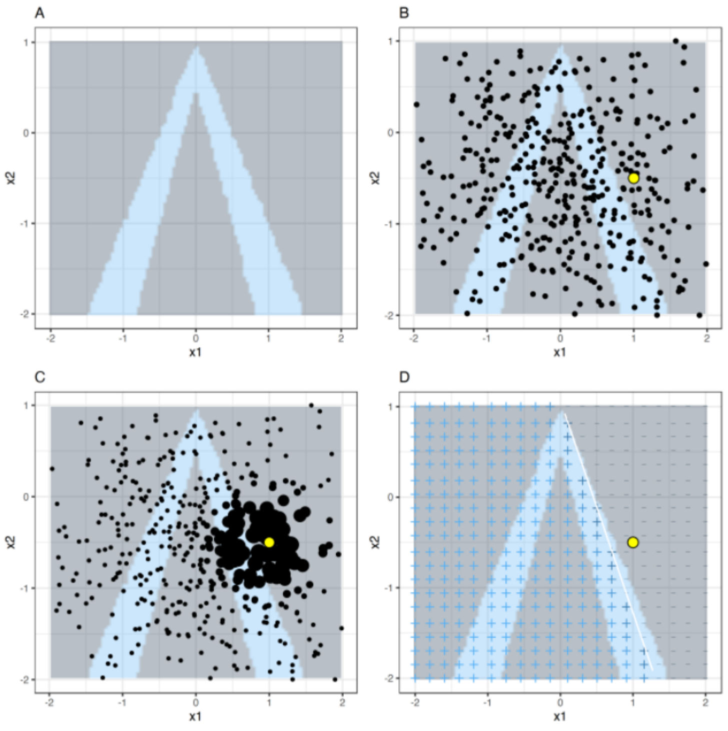


Figure 2.3: Figure showing the process of extracting LIME explanations for forecasts on tabular data [3]

2.1.2 Shapely Values

Another example of a post-hoc model agnostic interperatibility method is Shapely values. These values determine how much each feature in a model has contributed to a prediction made by the same model [3]. Such a value can help the user of a model understand better why a particular prediction was made, providing a level of transparency into the previously black-box model.

A shapely value is calculated using Equation 2.1, where S is the subset of features used in the model and p is the number of features [3]. *val* is a function which assigns a real number to each subset (S).

$$\phi_j(val) = \sum_{S \subseteq \{1,\ldots,p\} \setminus \{j\}} \frac{|S|!\,(p - |S| - 1)!}{p!} \left(val\left(S \cup \{j\}\right) - val(S)\right) \tag{2.1}$$

Shapely values ensure a fair payout by satisfying the below criteria :

- Efficiency - feature contributions must make up the differences between the prediction and the average.
- Symmetry - two features with the same added values will have the same contribution.
- Dummy - a feature that does not change the predicted output will have a shapely value of 0.
- Additivity - shapely values from different models may be added in order to get one shapely value.

2.1.3 SHapley Additive exPlanations (SHAP)

Lundberg and Lee [18] present an explainability approach connecting LIME and Shapely values as well as DeepLIFT and Shapely values for deep learning models. SHAP values are ultimately "the Shapley values of a conditional expectation function of the original model" [18]. This means that SHAP values are the Shapely values which are defined by a value function. The following are three properties of SHAP methods specified in [18] :

- Local Accuracy - when approximating the original model, local accuracy requires the explanation model to produce the same output as the original model for the simplified input.
- Missingness - features missing in the original input cannot have impact.
- Consistency - if a model changes such that a simplified input’s contribution does not decrease, the same input’s attribution should not decrease.

2.1.4 Counterfactual Explanations

Another mode-agnostic, post-hoc explainability method is Counterfactual Explanations. This method follows the intuition that if one event had occurred, the other would not have occurred. This is used to determine the relationship between input features and a prediction.
Molar et al. [3] discuss what is needed for a counterfactual explanation to be a good explanation. These include that the counterfactual explanation should be as close as possible to the predefined prediction. It should also describe the smallest change to the features that change the prediction to a predefined output. In order to obtain as much transparency as possible, multiple counterfactual explanations should be generated in order to have visibility of multiple ways to obtain the same prediction. The simplest way to generate counterfactual explanations is by trial and error, however more efficient methods using loss functions have been developed [3].

2.2 Time-Series Analysis

Time-series analysis can be described as the "endeavor of extracting meaningful summary and statistical information from points arranged in chronological order" [4, Chapter 1]. Time-series forecasting is an area of study in time-series analysis which entails predicting future values of a time-series. The most common methods to tackle time-series forecasting can be grouped into statistical approaches and machine learning approaches. There is no method which is considered to be the best in each situation, therefore most often, a model is chosen for an application based on results obtained through experimentation [19].

2.2.1 Statistical Methods

Throughout the years, several statistical methods have been developed in order to deal with the problem of time-series forecasting for both univariate and multivariate time-series. As the name suggests, univariate time-series contain only one variable whilst multivariate time-series contain multiple variables. Some of these methods are discussed below.

Exponential Smoothing

Exponential Smoothing is a time-series forecasting method which allocates weights that exponentially decay over time to time-series observations. Equation 2.2 shows the structure of simple Exponential Smoothing, where L represents the level at time t, α denotes the historical values' weights and z represents the observed values [19].

$$L_t = \alpha z_t + \alpha(1-\alpha)z_{t-1} + ... + \alpha(1-\alpha)^{m-1}z_1 \tag{2.2}$$

More complex exponential smoothing techniques include Holt's Exponential Smoothing, which takes into consideration trends in the time-series and Holt-Winter's Seasonal Exponential Smoothing, which extend Holt's method to also take into consideration seasonality [19]. A trend in time-series data can be defined as a long-term change in data which follows a pattern over time whilst seasonality describes cyclic patterns of variation which repeat over relatively constant time intervals [19].

Autoregressive Integrated Moving Average (ARIMA) Models

ARIMA models combine the intuitions used in Autoregressive models, Integration and Moving Average models. Autoregressive models use the intuition that the past movement can be used to predict future movement of a time-series. Integration refers to the difference between observations, such that the time-series is made stationary. Moving Average models specify an output variable depending on values of stochastic errors.

ARIMA models use past values, lags and lagged forecast errors in order to forecast future values. These models are sometimes referred to as ARIMA (p, d, q) models, with the letters in parenthesis representing the model's parameters. p represents the lag order, d represents the degree of differencing and q represents the order of moving average [20].

A variation of the ARIMA model is the Seasonal ARIMA (SARIMA) model, which takes 2 sets of parameters - (p, d, q), which are the non-seasonal parameters and $(P, D, Q)_s$, which represent the seasonal parameters, specifically set for the seasonal component of the time-series [4].

Vector Autoregression (VAR)

In contrast with the other statistical methods discussed in this section, VAR models are used for multivariate time-series. Forecasts are generated by essentially extending autoregression to multiple time-series regressions. This is done by using the movement of the prediction variable along with the movement of other variables in the time-series [4].

2.2.2 Machine Learning Methods

Throughout the past 2 decades, machine learning methods for time-series forecasting have proven that in certain cases they can be as successful, or even more successful than the classical statistical methods [21]. Such models use historical data in order to learn the relationship between the past and the future. Some commonly used machine learning time-series forecasting methods will be discussed in this section.

K Nearest Neighbour (KNN)

KNNs are pattern recognition models which can be used to solve classification and regression tasks. These models calculate the Euclidean distance between a point and all the other points in a training set. The closest training data points are identified and the prediction is given as the average of the target values of these points [22].

Support Vector Regression (SVR)

Support Vector Machines (SVM) models aim at finding a hyperplane in an N-dimensional space that separates different classes. SVR models use the same intuition as SVMs in order to forecast continuous variables. However, this is done by finding the closest point to the hyperplane, rather than using the hyperplane to separate the points as is done by SVMs. The ultimate goal for an SVR model is to fit the error within a certain threshold. Figure 2.4 depicts the intuition behind such models.

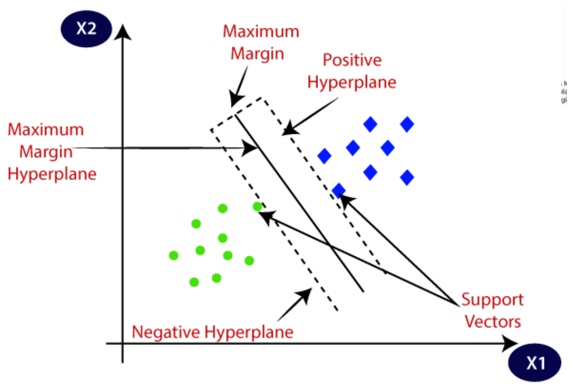


Figure 2.4: A depiction of how an SVR makes a prediction. The predicted value is the closest point to the hyperplane [1]

Artificial Neural Network (ANN)

ANNs are a deep learning network of connected weighted nodes which were inspired by biological neural networks. An ANN takes in an input and forward propagates it through a network depending on features and weights of the network, finally producing an output. This output is back-propogated through the model and the error generated is used to update the model's weights [19]. Models in which the output is not back-propogated are called *feed-forward* ANNs. Figure 2.5 depicts a simple feed-forward ANN.

Recurrent Neural Network (RNN)

RNNs are ANNs which allow for connections between neurons to form a cycle and for signals to travel in different directions, meaning that information may be persisted all throughout the network. This is accomplished by making use of a recurrent condition on the hidden state, allowing the network to use both the input to the current state and all of the previous outputs in order to produce a final output [19].

[1] https://medium.com/essence-of-learning/intuition-behind-support-vector-regression\-3601f670a2ef

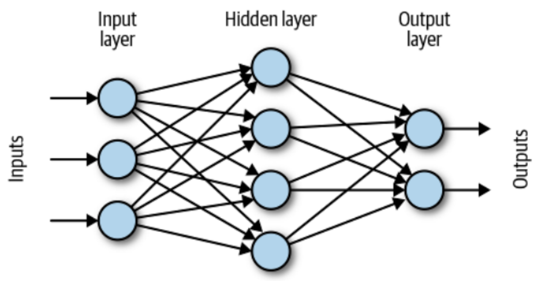


Figure 2.5: A simple feed-forward ANN consisting of an input layer, hidden layer and an output layer. The arrows represent the connections whilst the blue circles represent the nodes [4].

Bayesian Neural Network (BNN)

BNNs use Bayes' Theorem in order to express the model's weights and parameters as a probability distribution [22]. Equation 2.3 shows the formula used to calculate the posterior parameter distribution, where $P(\theta|D)$ is the posterior probability. In the numerator, $P(D|\theta)$ is the likelihood of the observations, $P(\theta)$ is the prior probability and the denominator represents the normalizing constant [22].

$$P(\theta|D) = \frac{P(D|\theta)P(\theta)}{\int P(D|\theta)P(\theta)d\theta} \tag{2.3}$$

Since such models produce a distribution as predictions, the results need to be approximated from this distribution. One example of an approximation method which may be used to produce a final prediction is the Monte Carlo Simulations and Markov Chains method (*MCMC*). A Monte Carlo Simulation is used to predict the probability of different outcomes when random variables are present. The Monte Carlo simulation assigns a random value to such variables and the model is run repeatedly. At the end of the simulation, the results are averaged in order to provide an estimate. Markov Chains are systems in which states are connected by transitions which are all assigned a probability, where the probability of moving to a new state from a current state is only dependent on the current state [4].

Monte Carlo Simulations and Markov Chains (MCMC) may be combined in order to approximate the prediction. In order to do this, a Markov Chain with random probability distributions across states is created such that the chain ultimately converges towards a stationary distribution [23].

Long Short-Term Memory Networks (LSTM)

LSTMs are a special kind of RNN which contain multiple single-network layers instead of just one, each containing a forget-gate in order to determine which data should be kept and which shouldn't. This is done in order to tackle the problem of long-term dependencies in regular RNNs, which maintain information throughout the whole network. Figure 2.6 depicts the difference in the architecture between a traditional RNN and an LSTM.

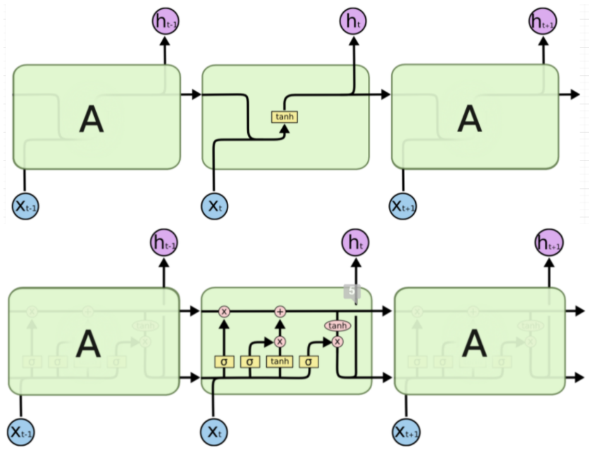


Figure 2.6: Traditional RNN architecture (top) and LSTM architecture (bottom) [2].

Transformation-gated LSTM (TG-LSTM)

Hu and Zheng [5] propose an LSTM architecture for capturing the short-term mutation dependencies of multivariate time-series. This is done by introducing a transformation gate which updates the state of the memory cells using a non-linear function on the forget gate. The output of the current state of the input gate and the output of the previous state of memory cells are activated via a hyperbolic tangent function. Figure 2.7 shows the difference between a traditional LSTM and the one proposed in [5]. One layer and multi-layer baseline machine learning models, including LSTM and IndRNN, were used in order to evaluate the performance of the proposed model by calculating the Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE) of each model. The TG-LSTM was found to learn short-term mutations better than the baseline models.

[2] `https://colah.github.io/posts/2015-08-Understanding-LSTMs`

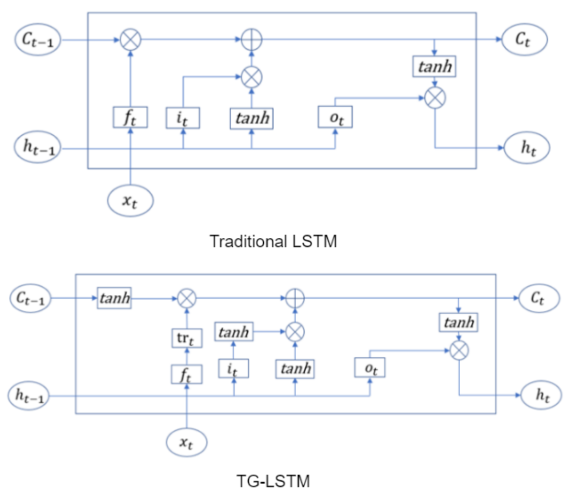


Figure 2.7: Difference between the traditional LSTM and the Transform-Gated LSTM proposed by Hu and Zheng in [5].

2.2.3 Statistical Methods vs. Machine Learning Methods

Several studies using diverse types of time-series datasets have been conducted to identify whether statistical methods or machine learning methods are most suitable to solve a given problem [24] [25] [26] [27] [7] [6].
In [6], Makridakis et al. states that machine learning models are inferior to statistical models in time-series forecasting by using several statistical and machine learning models on 1045 time-series. It was concluded that predictions from simpler statistical methods were overall at least as accurate as more sophisticated machine learning ones. Figure 2.8 shows the Mean Absolute Percentage Error (MAPE) values of each model studied. Since this is an error value, the lower it is, the more accurate a model is considered to be, meaning that in this study, statistical models were proven to be superior to machine learning models.

However, in [25], it is argued that the results presented by Makridakis et al [6] are biased since although many time-series are used, the number of observations in each time-series is small, with the average being 118. Cerqueira et al. [25] use 90 time-series, each containing 1000 observations in order to test the hypothesis of whether the datasets used in [6] were too small to generalize properly. [25] concludes that machine learning methods may be very useful for large, high frequency time-series, however could not state that in general, such methods outperform statistical methods. The importance of using both methods

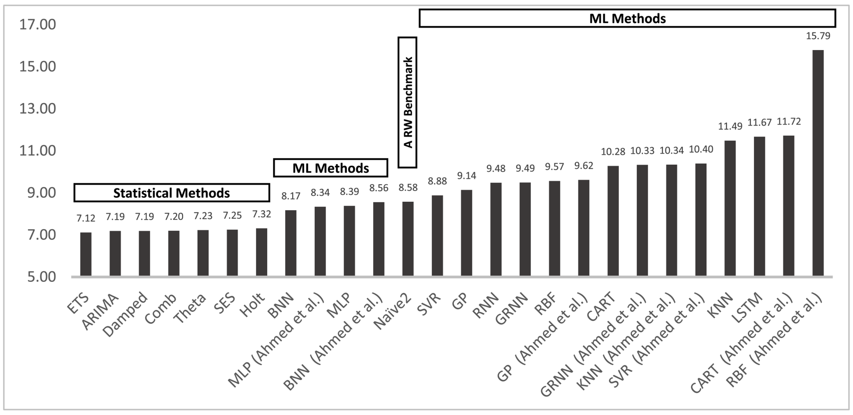
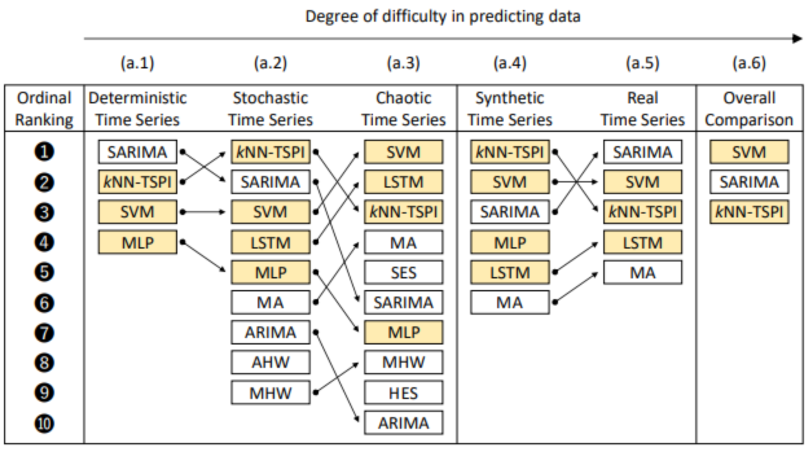


Figure 2.8: MAPE values of different models studied in [6].

for forecasting when doing experiments is highlighted.

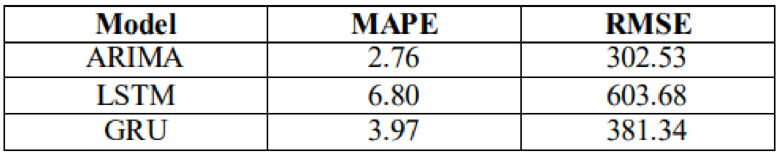
In [7], Parmezan et al. use 95 datasets, a mix of 11 statistical and machine learning predictors and 4 different types of evaluation metrics in order to extensively evaluate the performance of the different models. Along with this, an in-depth literature review of 117 studies on the use of different models for time-series forecasting carried out between 2009 and 2018. From these studies, the most popular models used were ANN, ARIMA and SVM models. Figure 2.9 shows the overall performance of the models implemented on the different dataset types. One can note that the top three models overall were SVM, SARIMA and kNN-TSPI. This study concludes that one may not state that one type of method (statistical or machine learning) is superior to the other.


Degree of difficulty in predicting data

	(a.1)	(a.2)	(a.3)	(a.4)	(a.5)	(a.6)
Ordinal Ranking	Deterministic Time Series	Stochastic Time Series	Chaotic Time Series	Synthetic Time Series	Real Time Series	Overall Comparison
❶	SARIMA	*k*NN-TSPI	SVM	*k*NN-TSPI	SARIMA	SVM
❷	*k*NN-TSPI	SARIMA	LSTM	SVM	SVM	SARIMA
❸	SVM	SVM	*k*NN-TSPI	SARIMA	*k*NN-TSPI	*k*NN-TSPI
❹	MLP	LSTM	MA	MLP	LSTM	
❺		MLP	SES	LSTM	MA	
❻		MA	SARIMA	MA		
❼		ARIMA	MLP			
❽		AHW	MHW			
❾		MHW	HES			
❿			ARIMA			

Figure 2.9: Performances of models implemented on different datasets in [7].

Yamak et al. [8] compare the performance of an ARIMA model, LSTM model and Gated Recurrent Unit (GRU) on predicting Bitcoin's price using a dataset of almost 4 and a half years. The accuracy of the models was calculated using MAPE and RMSE and the results are presented in Figure 2.10, showing that the ARIMA model produced the best results. All models used the same dataset. The authors note that adding more data and features might help improve the performance of the LSTM model [8].



Model	MAPE	RMSE
ARIMA	2.76	302.53
LSTM	6.80	603.68
GRU	3.97	381.34

Figure 2.10: MAPE and RMSE values of different models studied in [8].

In real world applications, time-series data might not always be consistent over time. Such datasets are referred to as intermittent time-series and the most common intermittent data is related to demand forecasting. Kiefer et al. [28] study the performance on different statistical and machine learning methods on intermittent time-series. In order to evaluate these models, the Stock-keeping-oriented Prediction Error Costs (SPEC) metric is used. The statistical Croston method produced better results than the LSTM model on the stock demand dataset used. However, the authors noted that univariate time-series were used and stated that by using more features, results could be more likely to lean in favour of machine learning methods [28].

In order to make use of the advantages of both statistical and machine learning methods, Zhang [29] experiments using a hybrid ARIMA and ANN architecture. The data is fit to the ARIMA model to analyze the linear part of the problem. Next, the residuals from the ARIMA model are fit to an ANN, thus exploiting the strength of ARIMA and the pattern-finding capabilities of an ANN. The hybrid and individual models were tested out on 3 datasets across different fields and the hybrid model always outperformed the individual ones. Mean Squared Error (MSE) and Mean Absolute Deviation (MAD) metrics were used to evaluate the performance of the models.

2.3 Similar Studies

In this section, some similar studies and the approaches taken in their methodologies and evaluations will be reviewed.

Explainable AI in Time-Series Forecasting

In any kind of business or field of work, forecasting the future is always very useful as this knowledge would be of great value when making decisions. Furthermore, some form of interpretability on the predictions which the model makes would help users of the systems make even more informed decisions. Several studies have been conducted exploring the use of Explainable AI on Time-Series Forecasting predictive machine learning models. Rojat et al. [1] present a survey of different studies done on this topic before April 2021. For each study, the authors document the following :

1. Ante-hoc/Post-hoc - this states whether the explainability being explored is obtained throughout the training of the model or upon the prediction of an instance.
2. Methodology - this specifies the explainable AI methodology used in the study.
3. Model Specific/Model Agnostic - this specifies whether the methodology used would work on any model or whether it is specific for one particular model.
4. Scope - the scope states whether the explanation is local, meaning that the explanation is given on a sample and is independent of all other samples or global, which means that the explanation method does not process the data sample by sample, but more of an average across all predictions.
5. Target audience - this denotes whether the target users for the systems developed in the studies are technical people (Developers) or non-technical ones, implying towards the user-friendliness of the explanations.
6. Explanation evaluation - the explanation evaluation specifies whether an evaluation of the study was held and if it was Qualitative or Quantitative.

Over 30 studies were surveyed and from the results, one can note that most explainability methods produce outputs which are aimed at being used by Developers. It may also be noted that the majority of the studies do not evaluate the explainability, and most that do use Qualitative methods for this evaluation, many of which were expert assessments of the systems.

One explainable convolutional neural network created specifically for Multivariate time-series classification by Fauvel et al. [9] is XCM. XCM uses a particular CNN architecture utilising fully padded 2D and 1D convolution filters such that it is as generic as possible and performs well on both smaller and larger sized datasets and also produces accurate identification of which variables and timestamps were important for the predictions at each layer by using Gradient-weighted Class Activation Mapping (Grad-CAM) [30]. Grad-CAM creates post-hoc visual explanations for decisions obtained from Convolutional Neural Networks (CNN) by producing saliency maps highlighting the important attributes which were used to make a prediction. A simple example of such a visualization may be seen in Figure 2.11, where dummy data representing a sine wave as a negative class and square sine wave as a positive class is used.

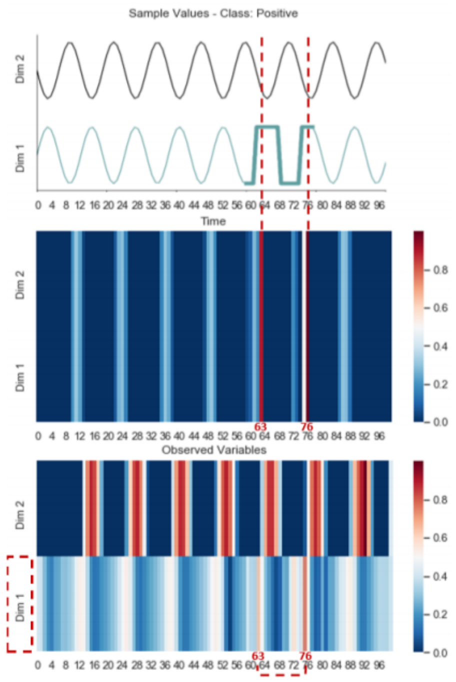


Figure 2.11: A very simple example of a feature attribution visualization shown in [9].

Using these visualizations, the exact point of the square wave may be seen as well as which dimension contains the square wave, thus causing the result to be negative.
In [10], Souza and Leung propose an XAI system which aims at creating an interactive user interface in which one may see model predictions and explanations on predicting customer churn, consisting of the following :

1. *Back-end Component* which consists of the model architecture, shown in Figure 2.12 on which training, predictions and explanations are performed.
2. *Front-end Component* which outputs the results obtained from the back-end component to an understandable, interactive user interface.

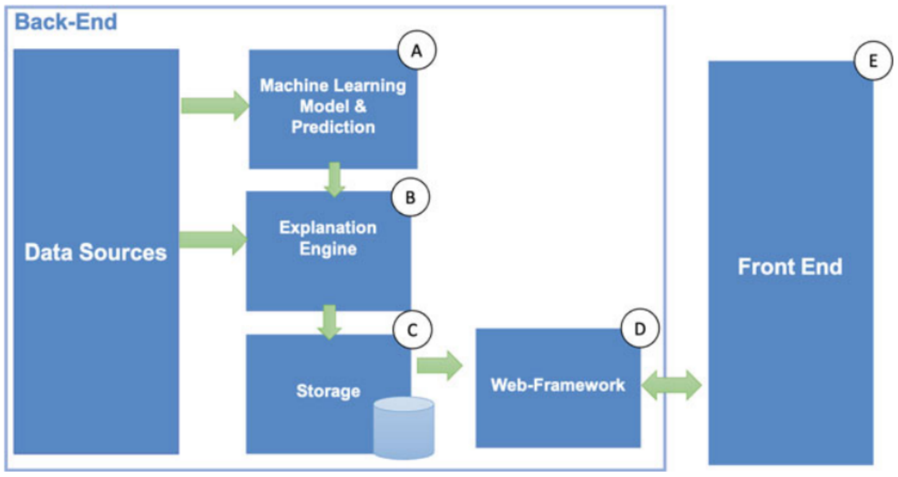
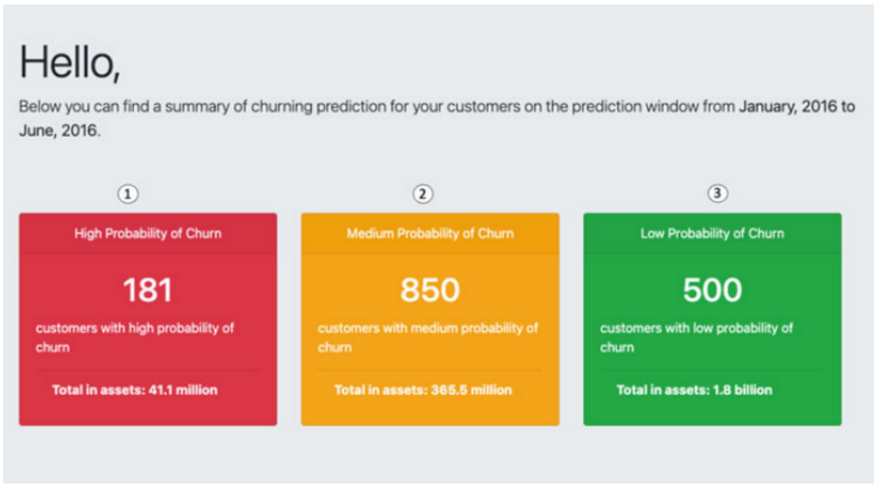


Figure 2.12: Architecture of solution proposed by Souza and Leung [10].

A random forest model was applied and the explanation engine produced explanations based on Shapley values for local and global explanations and contrastive explanations for model recommendation. The storage is a database used in order to store the generated explanations which may be used by the front-end interface.
The front-end interface consists of the following five sections:

1. *Home screen* showing a summary of the customers in each group, as shown in Figure 2.13.



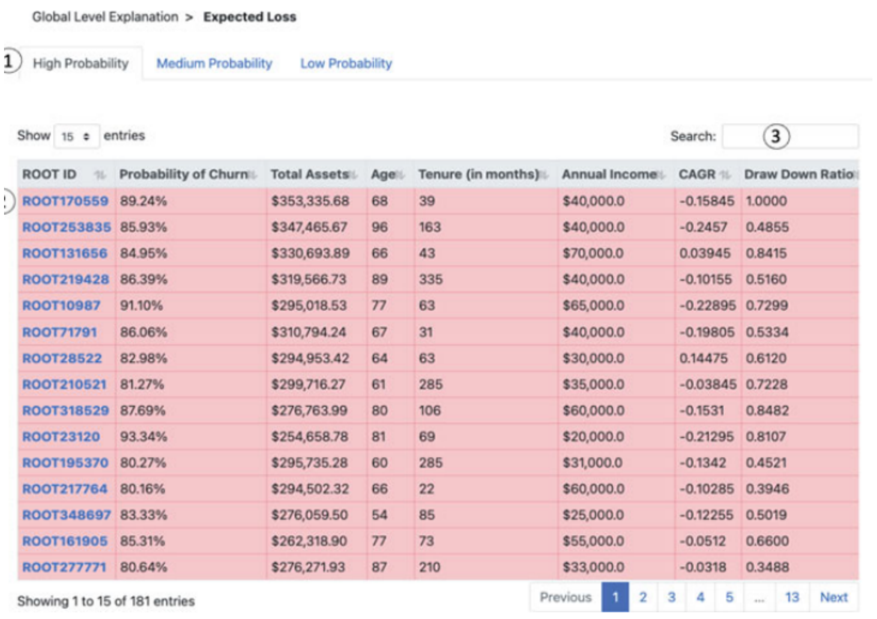
Hello,

Below you can find a summary of churning prediction for your customers on the prediction window from **January, 2016** to **June, 2016**.

① High Probability of Churn	② Medium Probability of Churn	③ Low Probability of Churn
181	850	500
customers with high probability of churn	customers with medium probability of churn	customers with low probability of churn
Total in assets: 41.1 million	**Total in assets: 365.5 million**	**Total in assets: 1.8 billion**

Figure 2.13: Home screen for the interface created in [10]

2. *Expected Losses* are displayed in a table which also allows users to search for an instance. This table is shown in Figure2.14



Global Level Explanation > **Expected Loss**

1 High Probability | Medium Probability | Low Probability

Show 15 entries

Search: 3

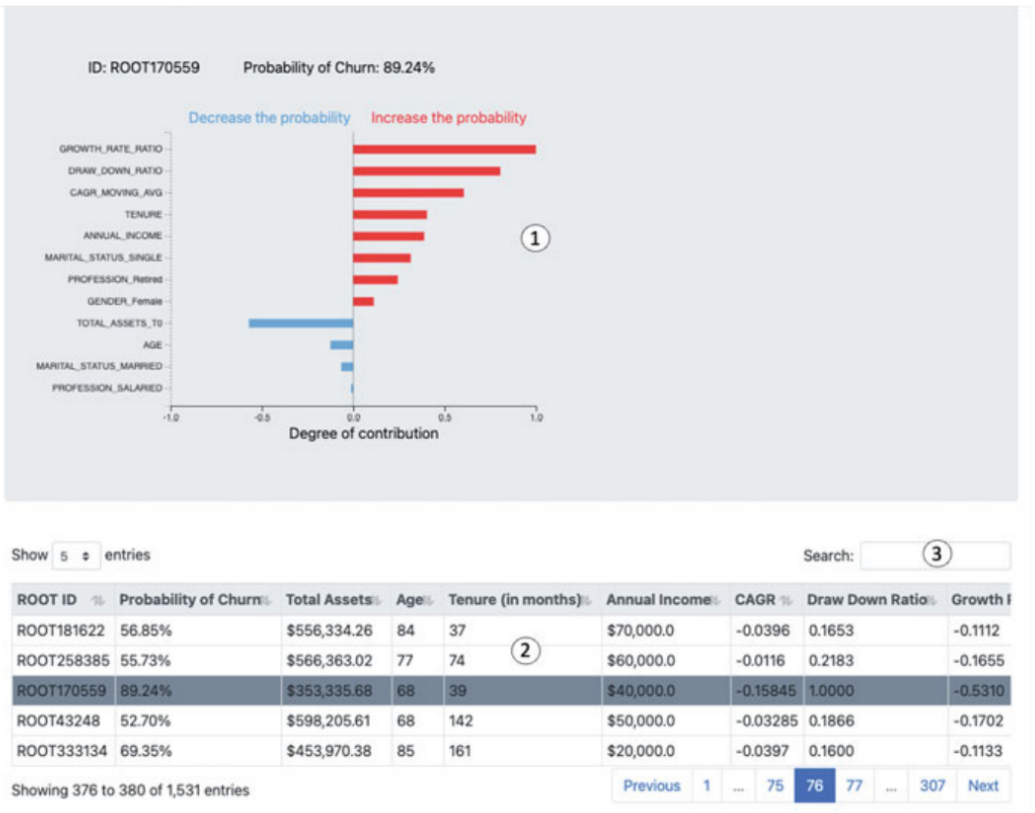
ROOT ID	Probability of Churn	Total Assets	Age	Tenure (in months)	Annual Income	CAGR	Draw Down Ratio
ROOT170559	89.24%	$353,335.68	68	39	$40,000.0	-0.15845	1.0000
ROOT253835	85.93%	$347,465.67	96	163	$40,000.0	-0.2457	0.4855
ROOT131656	84.95%	$330,693.89	66	43	$70,000.0	0.03945	0.8415
ROOT219428	86.39%	$319,566.73	89	335	$40,000.0	-0.10155	0.5160
ROOT10987	91.10%	$295,018.53	77	63	$65,000.0	-0.22895	0.7299
ROOT71791	86.06%	$310,794.24	67	31	$40,000.0	-0.19805	0.5334
ROOT28522	82.98%	$294,953.42	64	63	$30,000.0	0.14475	0.6120
ROOT210521	81.27%	$299,716.27	61	285	$35,000.0	-0.03845	0.7228
ROOT318529	87.69%	$276,763.99	80	106	$60,000.0	-0.1531	0.8482
ROOT23120	93.34%	$254,658.78	81	69	$20,000.0	-0.21295	0.8107
ROOT195370	80.27%	$295,735.28	60	285	$31,000.0	-0.1342	0.4521
ROOT217764	80.16%	$294,502.32	66	22	$60,000.0	-0.10285	0.3946
ROOT348697	83.33%	$276,059.50	54	85	$25,000.0	-0.12255	0.5019
ROOT161905	85.31%	$262,318.90	77	73	$55,000.0	-0.0512	0.6600
ROOT277771	80.64%	$276,271.93	87	210	$33,000.0	-0.0318	0.3488

Showing 1 to 15 of 181 entries

Previous 1 2 3 4 5 ... 13 Next

Figure 2.14: Expected losses displayed in the interface created in [10]

3. *Local Feature Importances* are depicted by a tornado plot showing the positive and negative contributions of the features as well as a table with a search component. as shown in Figure 2.15



ROOT ID	Probability of Churn	Total Assets	Age	Tenure (in months)	Annual Income	CAGR	Draw Down Ratio	Growth I
ROOT181622	56.85%	$556,334.26	84	37	$70,000.0	-0.0396	0.1653	-0.1112
ROOT258385	55.73%	$566,363.02	77	74	$60,000.0	-0.0116	0.2183	-0.1655
ROOT170559	89.24%	$353,335.68	68	39	$40,000.0	-0.15845	1.0000	-0.5310
ROOT43248	52.70%	$598,205.61	68	142	$50,000.0	-0.03285	0.1866	-0.1702
ROOT333134	69.35%	$453,970.38	85	161	$20,000.0	-0.0397	0.1600	-0.1133

Showing 376 to 380 of 1,531 entries

Figure 2.15: Local Feature Importance displayed in the interface created in [10].

4. *Global Feature Importances* are depicted on a bar chart showing the contribution of each feature, as shown in Figure 2.16

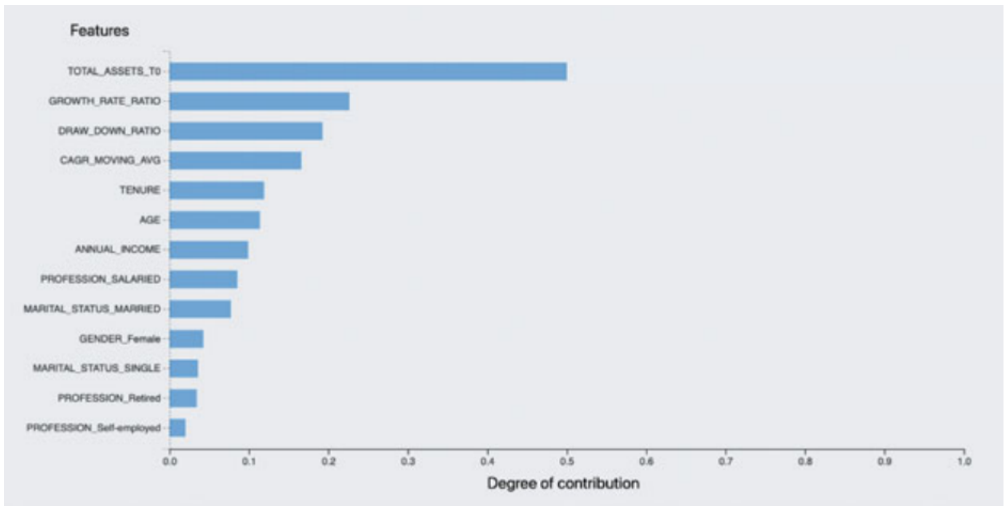
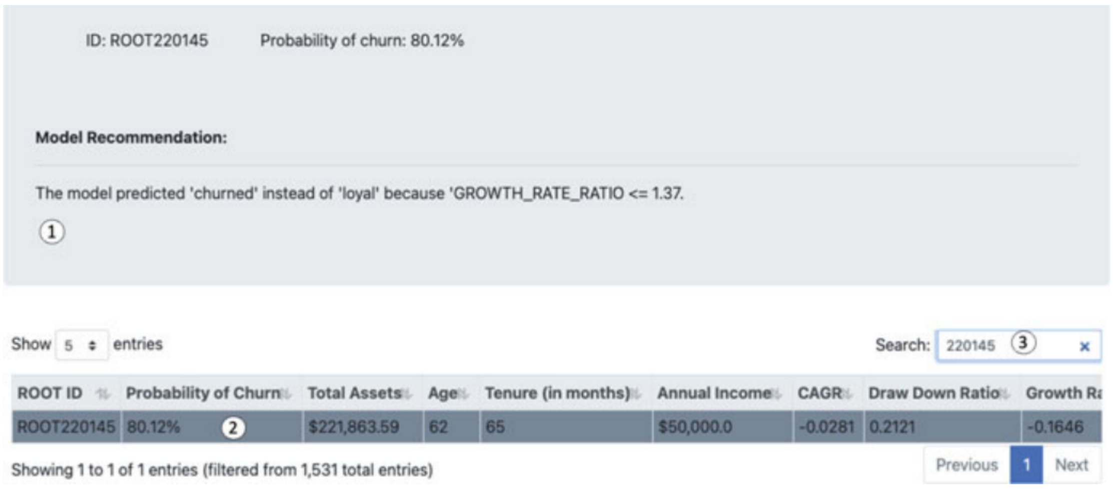


Figure 2.16: Global Feature Importance displayed in the interface created in [10].

5. *Model Recommendations* are represented by a textual explanation contrasting the fact (true output class) and foil (constructive class). A table having a search functionality is also shown. These results are depicted in Figure 2.17


ID: ROOT220145 Probability of churn: 80.12%

Model Recommendation:

The model predicted 'churned' instead of 'loyal' because 'GROWTH_RATE_RATIO <= 1.37.

①

Show 5 entries Search: 220145 ③

ROOT ID	Probability of Churn	Total Assets	Age	Tenure (in months)	Annual Income	CAGR	Draw Down Ratio	Growth Ra
ROOT220145	80.12% ②	$221,863.59	62	65	$50,000.0	-0.0281	0.2121	-0.1646

Showing 1 to 1 of 1 entries (filtered from 1,531 total entries)

Figure 2.17: Model Recommendations as displayed in the interface created in [10].

In order to evaluate this solution, it was compared to three other similar studies. The conclusion of this evaluation stated that the created solution delivers an adequate amount of features when compared to the solutions proposed in the other studies. Souza and Lang [10] also conclude that the created system generates explanations which may be understood by non-expert users. They also plan on conducting a human evaluation of the created solution as future work.

Stability of Explanations

Due to intrinsic randomness of explainability methods, running the explanations multiple times might produce different results. The more similar the results are, the more stable the explanations are. The following are two methods used in order to measure the stability of these explanations. Stability is an important measure as it helps one determine how much they should trust the explanations produced.
Credit scoring is an analysis of a person's finances which determines whether

they are eligible or not to apply for a loan. Both Machine Learning and Statistical methods have been applied to this problem, with Machine Learning showing great promise in this field of forecasting [11]. Visani et al. [11] go a step further from simply making predictions by applying post-hoc explainability methods to extracting LIME values for these predictions and also assessing the stability of these LIME values. When extracting LIME values for a prediction, the best case scenario is getting the same values in each run, however every time LIME explanations are generated, new random datapoints are generated, meaning that getting the same values is not guaranteed. In order to determine the stability of LIME explanations, Visani et al. [11] propose performing multiple LIME calls and comparing the values produced. Two indices were created in order to make such comparisons - the Variable Stability Index (VSI) and the Coefficients Stability Index (CSI). The VSI is used in order to compare the variable composition obtained from the different explanations whilst the CSI is used to obtain the equality between coefficient of all the different Weighted Ridge Regression models output by LIME.

These experiments were applied to the Credit scoring problem by using Logistic Regression and Gradient Boosting Trees. Gini values, a reliable measure of credit scoring model performance, was used in order to conduct an evaluation of the models, the results of which concluded that the Gradient Boosting Tree model produced better results than the Logistic Regression model. LIME was then tested out on the Gradient Boosting model predictions. In order to obtain the VSI and CSI values, LIME was applied ten times, obtaining the top seven most important features.

Figure 2.18 shows an example of an explanation with high stability values (left), and an explanation with low stability values (right). One can note that the explanation bar charts for the high stability explanations are very similar whilst those coming from the low stability explanation are very diverse.

In [31], Man and Chan compare the stability of Mean Decrease Accuracy (MDA), LIME and SHAP by using an instability index which uses the average of the variances of all feature explanations. Experiments were performed on two synthetic datasets, a proprietary financial trading dataset and two public datasets. These

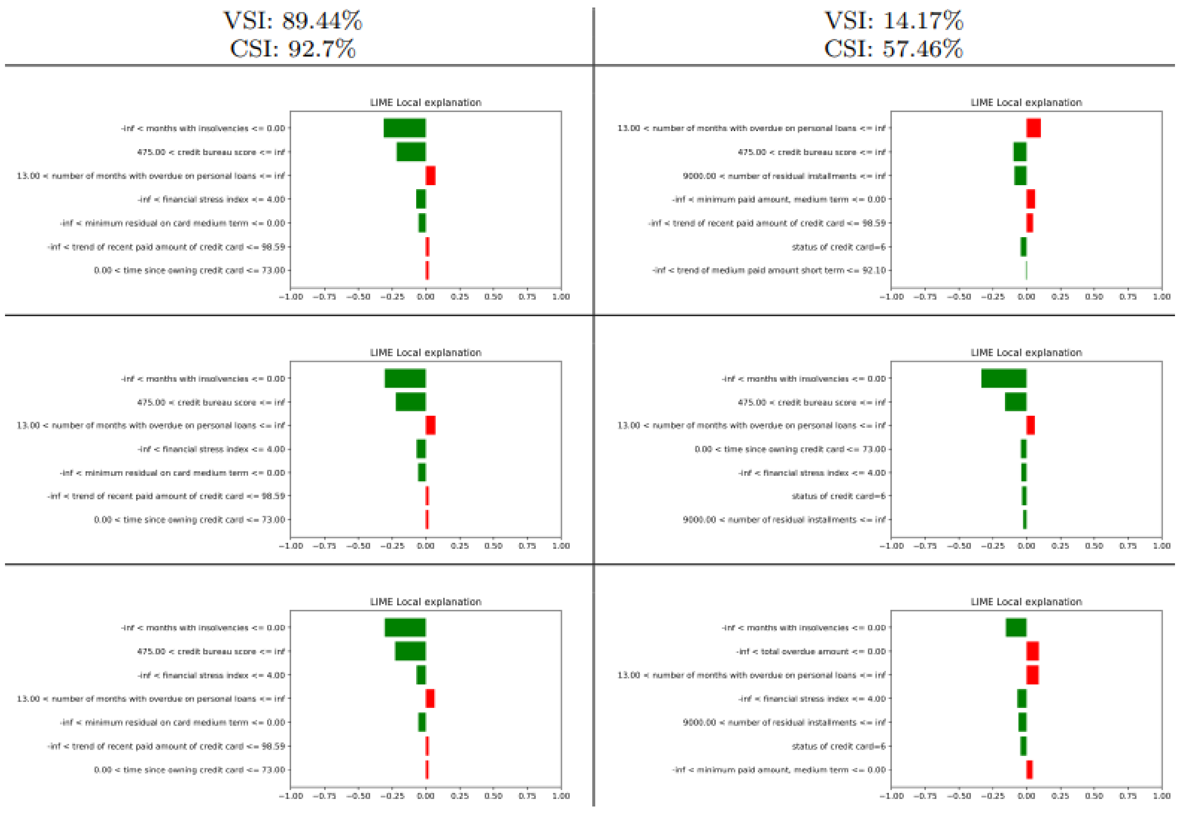


Figure 2.18: Figure showing results obtained in [11].

experiments resulted that LIME and SHAP are generally more stable than MDA and LIME is at least as stable as SHAP for the top ranked features.

Forecasting Individual Panel Data

In statistics, panel data, also referred to as cross-sectional time-series, follows a sample of individuals along time, consisting of repeated observations related to each individual in this sample [32]. Such datasets are commonly used in economic time-series, an example of which is trading time-series, where one will find multiple rows related to the same stock and corresponding to different days within the same time-series.

In [12], such a time-series is used at an individual level, meaning that the training dataset only contained data related to 1 stock. This data was fit to LSTM, Random Forests, Multilayer Perceptrons and Pseudo-random models. Figure 2.19, shows the accuracy results obtained by the different models on the different stocks, where one may note that the majority of times, LSTM has a higher accuracy than the other models. The author concludes that forecasting stock predictions on an individual stock panel data using an LSTM manages to return a positive result for all stocks, despite not constantly returning the best results when compared the the other models. The LSTM was also very successful in identifying variations, therefore also helping to mitigate risks.

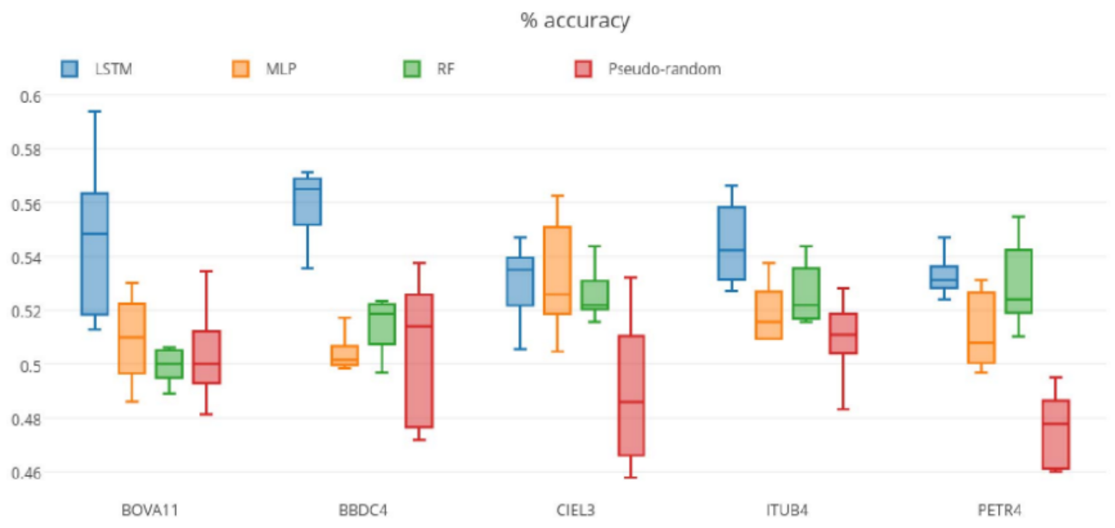


Figure 2.19: Figure showing accuracy of models tested out on different stock panel datasets in [12].

One problem which might arise when using individual data is having an insufficiently sized dataset. In order to tackle this problem, Nguyen et al. [33] propose first training an LSTM model using a lot of data from different stocks. The weights generated by this model are then used by a second model via transfer learning. The second model's training data relates to only one stock, making it an individual panel dataset. The results obtained by these models is compared to four baseline machine learning models - Support Vector Machine, Random Forest, K-Nearest Neighbour and an exiting LSTM architecture. The model using transfer learning resulted in more accurate predictions than the baseline models for all stocks the models were tested on. The authors finally proposed a more advanced system which also allows for transfer learning using other sources of data including more numerical data as well as sentiment information.

2.4 Conclusion

In this chapter, different methods of explainability were discussed, along with different measures which may be used to evaluate such methods. A gap in research for evaluating explainability was identified. Studies on forecasting time-series data using statistical and machine learning methods were discussed and compared, concluding that in general, no method may be considered superior to the other. Studies on explainability in time-series forecasting models and creating understandable and user-friendly explanations were also reviewed. The following chapter goes through how the research documented in this chapter was used in order to implement a solution for the purpose of this study.

Chapter 3 - Materials and Methods

In this chapter, all the experiments carried out in order to fulfil the objectives outlined in Chapter 1 will be discussed. Figure 3.1 depicts an overview of the pipeline that was used in order to obtain the required results, which are discussed in the following chapter. This pipeline consists of the following steps:

1. Extracting the wallet and payments data from the database.
2. Performing Exploratory Data Analysis (EDA) on the extracted data.
3. Using the EDA results to create a time-series dataset.
4. Finding and training the best model for the created dataset.
5. Obtaining predictions along with their LIME and SHAP explanations and confidences.

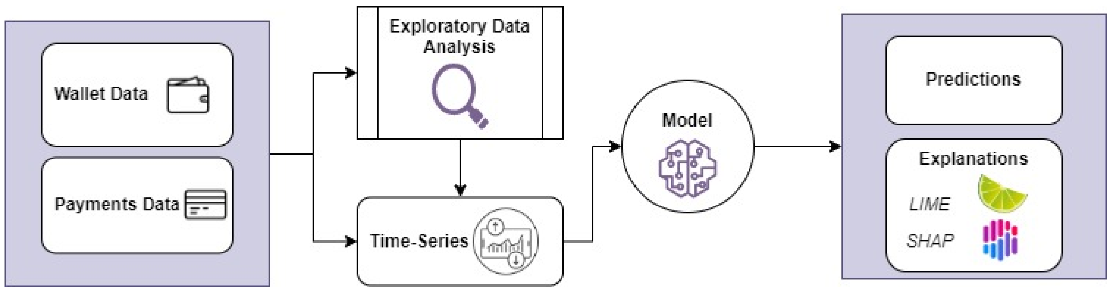


Figure 3.1: Outline of the methodology applied in this research.

By implementing this pipeline, the Aim and Objectives described in the first chapter were all achieved. The first aim (O1) is achieved through steps two and three. The second aim (O2) is achieved through step four. The third and fourth aims (O3 and O4 respectively) are achieved through the final step.

3.1 Data Extraction

For the purpose of this research, a time-series dataset will be created using data from an i-gaming company. The first step to creating such a dataset is extracting the data required from the company's Data Warehouse. The company owns over 20 brands, housing the data for most of these brands in the Data Warehouse. For the purpose of this study, only data from three larger brands is being considered. This was done using MySQL [1] queries. Two extractions were made from the database, one for payments data and another for wallet data. The timeframe used for both extractions was 1st January 2021 up to 1st February 2022 (both inclusive).

3.1.1 Payments Data Query

First, a list of customers who made more than 2 successful payments in December 2021 and registered with the brand before 2021 is extracted. This will exclude any customers who definitely did not make payments throughout the whole year of 2021 due to not registering in the beginning of the year and also customers who do not often make payments. In order to obtain the required data, the payment transactional table was joined to several dimensional tables. Figure 3.2 shows how the dimensional tables were used in order to extract the data.

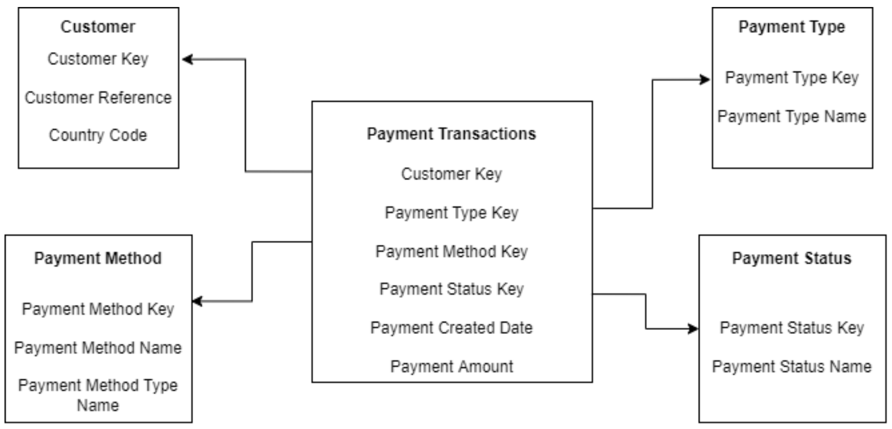


Figure 3.2: Structure of the tables used to extract the payments data.

[1]https://www.mysql.com/

3.1.2 Wallet Data Query

For the customers specified in the previous sub-section, wallet transactional data was also extracted. The wallet table contains a row for each and every transaction affecting a customer's wallet. Due to this low level of granularity, this table was very large and the amounts were aggregated by the extracted dimensions in order to reduce the time required to extract the data as well as the size of the extract. The only other condition applied to the data was to exclude rows amounting to 0 in the extract. Figure 3.3 shows how the dimensional tables were used in order to extract the data.

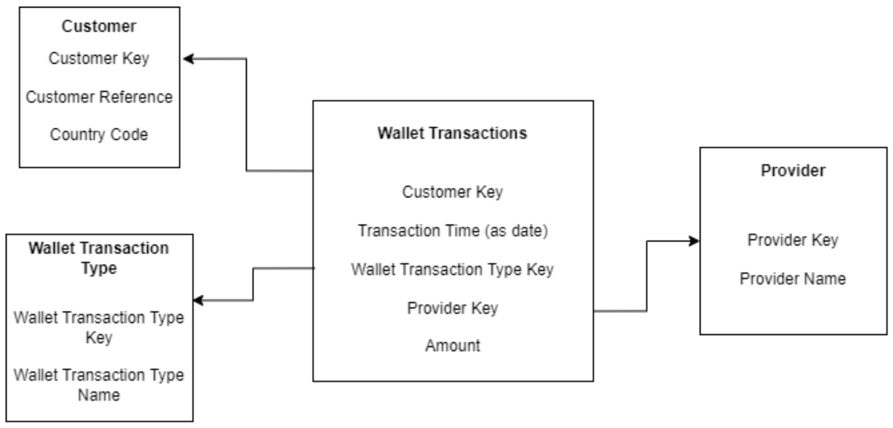


Figure 3.3: Structure of the tables used to extract the wallet data.

The following section will provide a description of the data extracted along with an in-depth analysis of this data.

3.2 Exploratory Data Analysis (EDA)

The initial step was to perform an in depth analysis in order to understand the data and determine which features to take into consideration when creating the time-series. For most of this analysis, 1 month of data was used. All the plots generated throughout this analysis may be found in Appendices C and D.
In order to properly analyze the data, it was transformed into a test time-series. One must note that this is not the final time-series used to train the model. The data was aggregated into 4 hour time-segments and a time-series containing the fields described in Table 3.1 was created.

Column Name	Column Description
customerGuid	Anonymized reference string for a customer.
countryCode	Signifies the country a customer resides in.
sum_n_SB	Negative transactions on the Sportsbook product.
sum_p_SB	Positive transactions on the Sportsbook product.
sum_n_GOC	Negative transactions on the Games of Chance product.
sum_p_GOC	Positive transactions on the Games of Chance product.
sum_n_O	Negative non-provider transactions.
sum_p_O	Positive non-provider transactions.
sum_Deposit_Wallet_f	Sum of failed deposits using a wallet payment method type.
sum_Deposit_Wallet_s	Sum of successful deposits using a wallet payment method type.
sum_Deposit_CreditCard_f	Sum of failed deposits using a credit card payment method type.
sum_Deposit_CreditCard_s	Sum of successful deposits using a credit card payment method type.
sum_Withdrawal_Bank_f	Sum of failed withdrawals using a bank payment method type.
sum_Withdrawal_Bank_s	Sum of successful withdrawals using a bank payment method type.
sum_Withdrawal_Wallet_f	Sum of failed withdrawals using a wallet payment method type.
sum_Withdrawal_Wallet_s	Sum of successful withdrawals using a wallet payment method type.
sum_Withdrawal_CreditCard_f	Sum of failed withdrawals using a credit card payment method type.
sum_Withdrawal_CreditCard_s	Sum of successful withdrawals using a credit card payment method type.
count_Deposit_Wallet_f	Count of failed deposits using a wallet payment method type.
count_Deposit_Wallet_s	Count of successful deposits using a wallet payment method type.
count_Deposit_CreditCard_f	Count of failed deposits using a credit card payment method type.
count_Deposit_CreditCard_s	Count of successful deposits using a credit card payment method type.
count_Withdrawal_Bank_f	Count of failed withdrawals using a bank payment method type.
count_Withdrawal_Bank_s	Count of successful withdrawals using a bank payment method type.
count_Withdrawal_Wallet_f	Count of failed withdrawals using a wallet payment method type.
count_Withdrawal_Wallet_s	Count of successful withdrawals using a wallet payment method type.
count_Withdrawal_CreditCard_f	Count of failed withdrawals using a credit card payment method type.
count_Withdrawal_CreditCard_s	Count of successful withdrawals using a credit card payment method type.

Table 3.1: Table showing columns used for Exploratory Data Analysis.

An important thing to note is that the polarity of values coming from the customer's wallet is from the customer's point of view, meaning that positive values represent money moving into a customer's wallet, and negative values represent money moving out of a customer's wallet. For the purpose of the Data Analysis, the wallet data was extracted at its lowest granulairty, meaning a row for each transaction was extracted. This time-series was created by applying the following transformations on the data :

- Split up payments into successful and failed payments by using the payment status.
- Aggregate the payments by customer, country, time-segments, payment type, payment method type and status.
- Pivot the table so as to have a column for each payment type, payment method type and success/failure.
- Split up wallet transactions by product using the provider column. All transactions under the Sportsbook (SB) product come from one provider, making these transactions easy to identify. Transactions with a -1 provider

reference signify that the transaction is not related to a provider. These are usually related to bonuses. The rest of the transactions go under the Games of Chance (GOC) product.

- Split up the wallet transactions by polarity.
- Aggregate these transactions by the customer, country, time-segments, polarity and product.
- Pivot the tables so as to have a column for each polarity and product.
- Join the payments and wallet data on the customer, country and time-segments in order to obtain one dataset, as described in Table 3.1.

Initially, all the numerical columns were displayed as bar charts. Figure 3.4 shows the plots related to wallet data. From these plots, it may be remarked that most times, customers get more positive transactions when playing on the Sportsbook product than the Games of Chance product. This means that customers tend to win more when playing on Sportsbook than on Games of Chance. Another observation which may be noted is that the transactions tagged as *Other* are mostly positive transactions. This is because, as previously stated, most of these transactions are related to bonuses given to customers, meaning that a customer gains money from such a transaction.

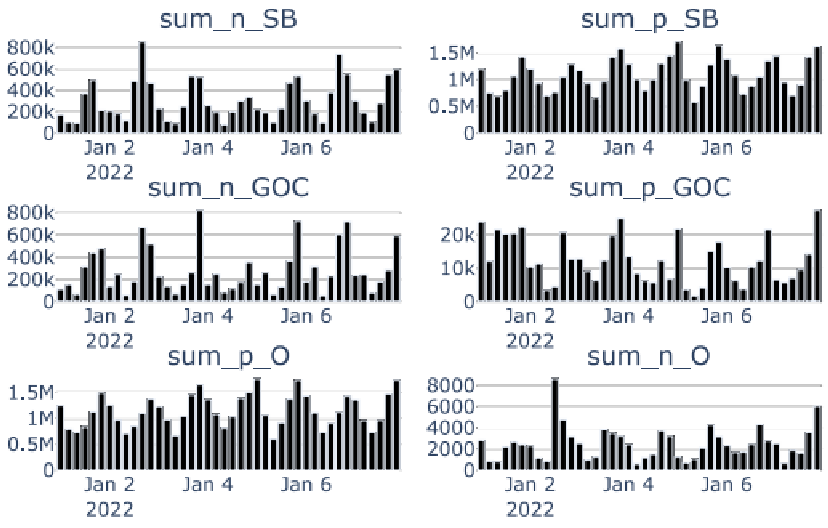


Figure 3.4: Barcharts showing the wallet numerical data.

Figure 3.5 shows all the plots related to successful deposit numerical data. Upon looking at these charts, one might note a cycle in the deposit amounts and counts over the times of day. This may also be noted for failed deposits as well as successful and failed withdrawals. In order to confirm this pattern, the payments were split based only on time, instead of datetime. It was noted that the volume and amount of successful deposits reach their peak from 4pm until 12am. All times are in UTC+1. On the other hand, for withdrawals, the peak is reached from 8pm until 12am.

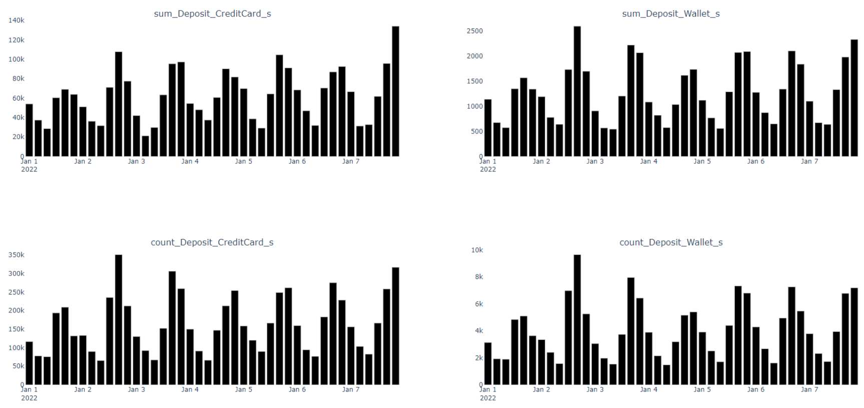


Figure 3.5: Barcharts showing the deposits numerical data.

An analysis on the payment patterns of customers playing different products was also conducted. Figure 3.6 shows the depositing behaviour of customers who have played SB and GOC.

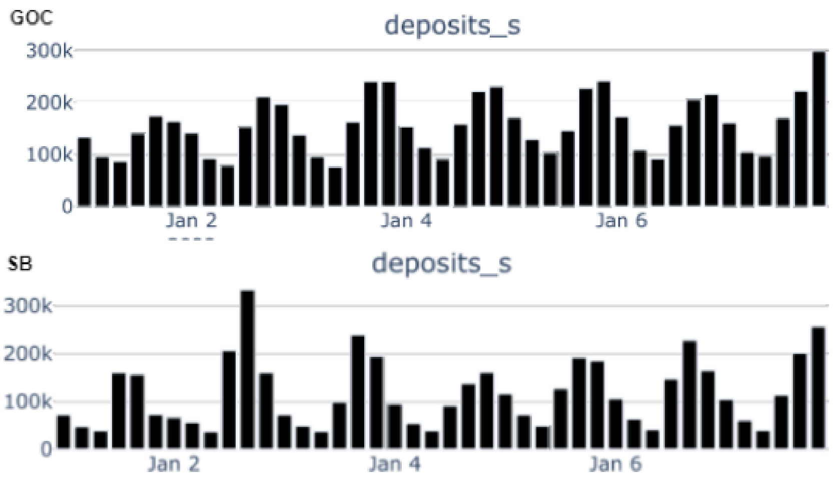


Figure 3.6: Barcharts showing the depositing behaviour of customers playing on GOC (top) and customers playing on SB (bottom).

One may note that the flow of deposits for customers playing GOC seems to be steadier than that for customers playing SB. This is likely due to the fact that more SB customers would deposit when events are actually taking place, meaning that more deposits would come in during the same time. A similar pattern was also observed for withdrawals.

Figure 3.7 shows all the successful deposits in relation to the wallet transactions. It may be noted that there seems to be a negative relationship between these two variables, implying that when a lot of negative transactions are affecting customers' wallets, therefore lowering their balances, customers tend to deposit more.

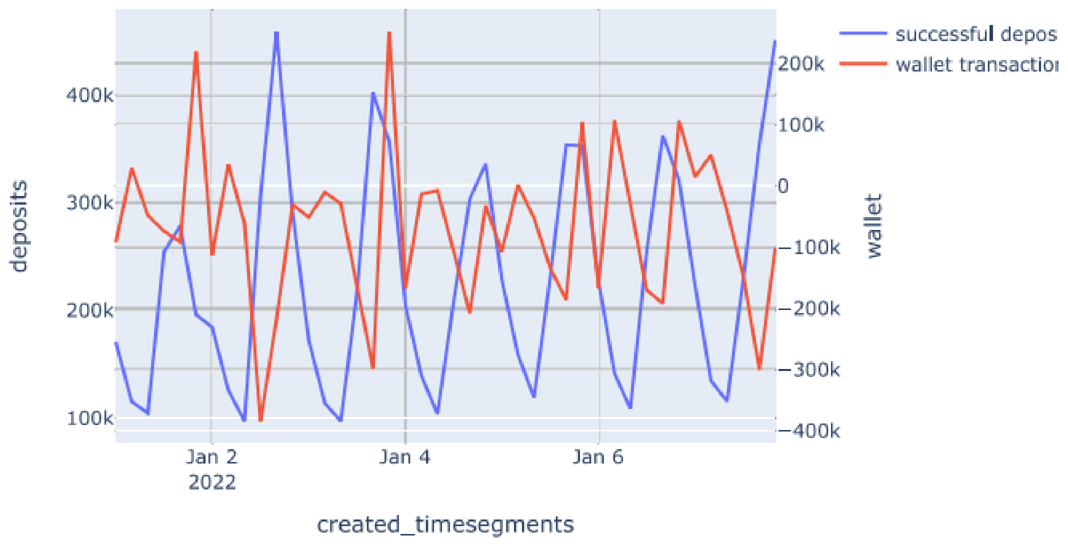


Figure 3.7: A chart showing successful deposits and wallet transactions along time.

Next, the correlations between variables was measured using both Pearson and Spearman Correlations. Figure 3.8 shows the correlation between successful withdrawals, successful deposits and all the aggregated wallet amounts. It may be seen that a positive correlation exists between the successful withdrawal amounts and the total wallet transaction amounts and a negative correlation exists between successful deposits and the total wallet transaction amounts. There is almost no correlation between successful deposit and withdrawal amounts.
An analysis of the amount of successful deposits coming from the different countries was also performed. It was noted that the country with most deposit

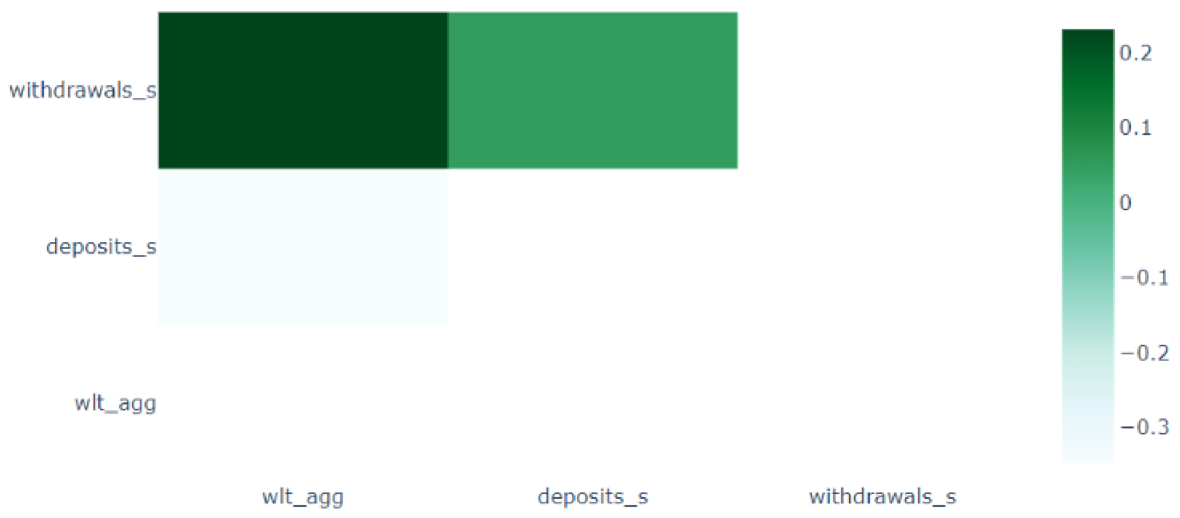


Figure 3.8: A chart showing the pearson correlation between variables.

amounts was Peru and that mostly wallet payment method types were used. Another observation was that wallet method deposits were more likely to fail than credit card method deposits.

Finally, an analysis of the number of depositing dates per customer was carried out. This was done using data for 13 months. Figure 3.9 depicts the count of customers per count of depositing dates. One can note that as the count of dates gets higher, the count of deposits gets lower. An analysis on the number of months customers deposited in was also carried out in order to get an idea of the distribution of the deposits along time. Interestingly, the number of months with most customer date counts was 13, meaning that the data being used should have a good distribution of data along time.

3.2.1 Creating the time-series

Using the results obtained from the EDA, a time-series was created. The customer country and the payment method types were omitted from the dataset and the values for these features were aggregated. The positive and negative values for the wallet transactions were also aggregated into one column. Instead of using four hour time-segments, daily time-segments were used. This helped make the dataset much less sparse. These columns were aggregated in order to reduce the size of the dataset whilst attempting not to lose any detail

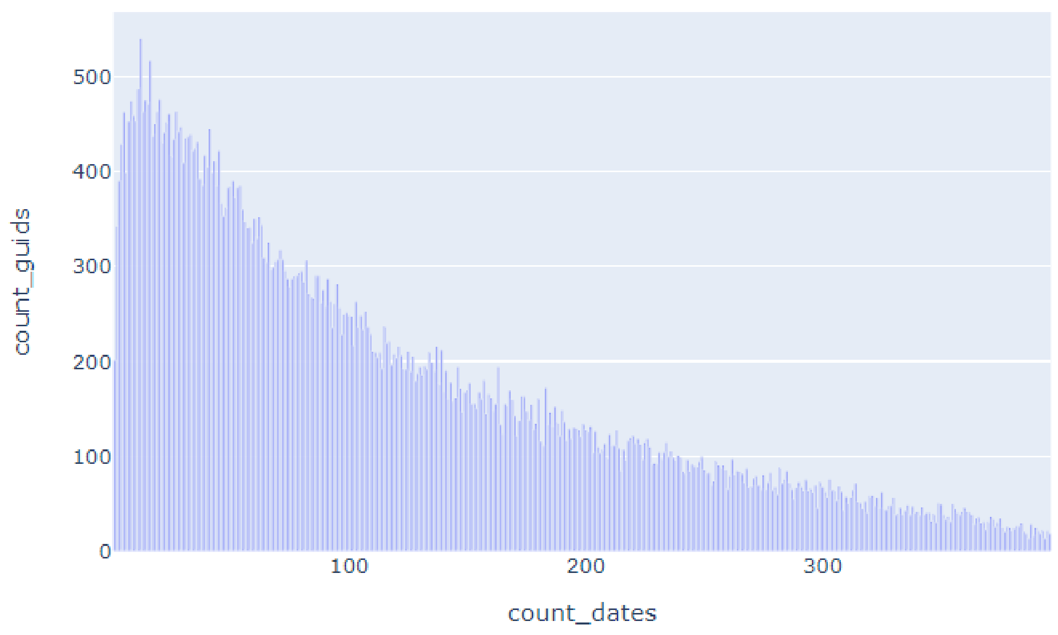


Figure 3.9: A chart showing the number of customers per number of deposit dates.

which would be useful during training.

The final time-series dataset used for training and forecasting contained the below columns :

- *created_timesegments* - A date field is included for each customer, even for dates when no activity was made. This is done in order for the dataset to follow the time-series structure.
- *customerGuid* - The anonymized customer reference.
- *sum_Deposit_s* - This is the number of successful deposits for the current day. This is the value which will be forecast by the model.
- *sum_GOC-1* - This is the sum of GOC transactions for the previous day.
- *sum_O-1* - This is the sum of Other transactions for the previous day.
- *sum_SB-1* - This is the sum of SB transactions for the previous day.
- *sum_Deposit_s-1* - This is the sum of successful deposits for the previous day.
- *sum_Deposit_f-1* - This is the sum of failed deposits for the previous day.
- *sum_Withdrawal_s-1* - This is the sum of successful withdrawals for the previous day.

- *count_Deposit_s-1* - This is the count of successful deposits for the previous day.
- *count_Deposit_f-1* - This is the count of failed deposits for the previous day.

3.2.2 Data Pre-Processing

In order to prevent the model running into memory overflow errors due to large numbers, the final dataset was scaled using Min-Max Normalization between 0 and 1. This means that all the values in the dataset were scaled such that the minimum value is 0 and the maximum value is 1. This is achieved using Equation 3.1.

$$\frac{x - min(x)}{max(x) - min(x)} \tag{3.1}$$

3.3 Model Selection and Training

One of the models created with the aim to use as a final model was a simple 1 layer LSTM model. The model was created using the Keras [2] python library. This model consisted of a dense layer with a sigmoid activation function, an Adam optimizer and Mean Absolute Error (MAE) loss. The number of epochs, learning-rate and number of hidden nodes were determined through a grid-search of different parameter combinations.

The second LSTM model created was the same as the first LSTM model, however contained an additional LSTM layer. Adding layers to an LSTM model allows for greater model complexity. This means that if the problem is too complex for a one-layer model, the two-layer model will produce more accurate forecasts. In order to obtain the best LSTM for the created time-series, several hyper-parameter, data transformation and data segmentation combinations were explored. Other forecasting models which do not use deep learning were also created in order to evaluate their performance against that of the LSTM.

The models were trained using the data for one customer at a time. This means that a model per customer was being trained. The dataset size for each model was one row per day for thirteen months, amounting to a total of 397 rows per

[2] https://keras.io/

customer, making the dataset relatively small. The dataset was split into train-test sets by separating the latest date for each customer as part of the test set.

3.3.1 Model Configurations

In order to determine the best model configurations, several custom grid-searches of different models, datasets and parameters were run. These grid-searches were run on a sample of 100 customers. The results obtained through these experiments are presented in Section 4.1.1. In order to evaluate the models, the Root Mean Squared (RMSE) error function was used.

One-layer vs. Two-layer
One of the initial grid-searches performed was to determine whether a one-layer or two-layer LSTM model performed best on the dataset created.

One-day window vs. Two-day window
A time-series may have more than one time-step. In order to test out whether a larger time window would improve performance, a two-day time window was tested out.

Data Segments As seen in Figure 3.9, the number of dates a customer deposits varies greatly. This means that some datasets may be much more sparse than others. Due to this, the data was split up into different segments in order to determine whether any configurations might be affected by the different types of datasets. The data was split up as follows :

1. Customers depositing on between 0 and 100 different days.
2. Customers depositing on between 100 and 200 different days.
3. Customers depositing on between 200 and 300 different days.
4. Customers depositing on more than 300 different days.

Learning Rate

The learning rate of a model signifies the step size of each training cycle used in order to move towards the minimum loss function. Grid-searches testing out different learning-rates on different data segments were run.

Epochs

Epochs represent the number of training cycles a model performs. Finding the right number of epochs is very important in order to avoid underfitting (the model not learning enough) and overfitting (when the model fits too closely to the dataset and is not able to predict unseen data well). A grid-search was run the get an idea of the values producing the lowest error. Early stopping was also implemented. This allowed the model to stop training whenever the loss did not decrease after three epochs.

Hidden Nodes

Increasing the number of hidden nodes in an LSTM makes the model deeper and more complex. Having too few hidden nodes might prevent convergence. Therefore it is very important to find the right number of hidden nodes for a model. Using the parameters selected from the earlier grid-searches, another grid-search was run in order to select the optimal number of hidden nodes for the LSTM.

Prediction Selection

Dropout is a regularization technique often used to prevent model overfitting. This is done by dropping out neurons during training in order to prevent them from depending on each other too much and allowing them to learn individually, thus preventing overfitting.

However, the main purpose of implementing dropout in this study is to obtain different predictions from the model. This is done by also turning on dropout at prediction. This will also drop out some neurons when predicting and in turn, the model will be able to return a different prediction upon each call. The process of making multiple predictions using Dropout in order to obtain a pre-

diction distribution is called *Monte Carlo Dropout*.
Implementing this in Keras is usually very simple and done by setting the training parameter to *True*, however the Keras LSTM model does not take this configuration. As a solution to this problem, the Keras dropout layers class is updated to a custom class which allows the use of dropout on an LSTM.
The prediction is called ten times and all the values obtained are stored in an array. As will be described in the following section, an explanation will be extracted for the prediction. This means that we cannot simply take the mean of predictions as the final prediction, since this value will not have an explanation directly associated to it. Because of this, an analysis was run in order to determine whether taking the median prediction or the prediction closest to the mean made a difference.

3.3.2 Benchmark Model Implementation

In order to be able to evaluate the results obtained through the main model, statistical benchmark models were implemented. As discussed in the previous chapter, such models are less complex than machine learning models, however still outperform them in certain cases [6] [8]. The results obtained from such models are compared to those obtained by the machine learning model in Section 4.1.2.

Naive Forecasting

Naive forecasting makes the assumption that the forecast of the upcoming value is equal to the value of the previous observation. This is one of the simplest models one may use to make forecasts, however it has still proven to be successful in certain scenarios [28].

Auto-ARIMA Forecasting

ARIMA is one of the most popular forecasting methods and considered to be a good baseline for comparing the performance of other models [28]. For the purpose of this study, an auto-ARIMA model using the python statsmodels [3] library

[3]https://www.statsmodels.org/stable/index.html

was implemented. As stated in the previous chapter, the auto-arima model takes a univariate time-series, therefore only the sum of successful deposits feature was used in order to make these predictions.

3.4 Explainable AI

One of the challenges being explored in this study is providing a layer of interpretebility to model predictions. This section discusses different explainability methods implemented with the aim of increasing a user's trust in the model's predictions.

3.4.1 LIME

The first explainability method implemented was LIME. This was done using the python LIME library [4]. Since LIME is model agnostic, this library contains different functions for different models and explanations. For the purpose of this study, the explainer used was the LIME Recurrent Tabular Explainer. This particular explainer suits the LSTM model as it is a recurrent neural network type model and the data being used is in a tabular form.
The explainer is first initialized using the trained model. This explainer is then used to generate ten different predictions in order to obtain ten different explanations for the top five features. These values are later used in order to help the user of the model interpret the prediction as well as to measure the level of confidence of the explanations when taken as a distribution.

3.4.2 SHAP

Another explainability method which was implemented for this study was SHAP. This was done using the SHAP python library [5]. Similar to LIME, SHAP is also model agnostic and this library contains functions for different models. Since the implemented LSTM is a deep neural network, the SHAP DeepExplainer is used, which utilizes the DeepLIFT algorithm in order to approximate shapely

[4]https://github.com/marcotcr/lime
[5]https://shap.readthedocs.io/en/latest/index.html

values.
Similar to the LIME implementation, the explainer is first initialized using the trained model and then used multiple times for different predictions in order to get different explanations.

3.4.3 Model Confidence

As explained previously, the implementation is generating a distribution of predictions using the Monte Carlo Dropout technique. This allows one to determine how robust a prediction may be. Metrics extracted from the prediction distribution include the minimum prediction, maximum prediction, mean prediction, median prediction, standard deviation and coefficient of variation. Ideally, for a robust prediction, all predictions are the same and the coefficient of variation is 0.

3.4.4 Explainability Confidence

Since explanations for different predictions were extracted, these may also form a distribution. In order to evaluate the robustness of the explanations, an index similar to those proposed in [11] and [31] was implemented. This index is ultimately the coefficient of variance of all the explanations. This is found by computing the mean of the coefficients of variances for each feature. This process is depicted in Figure 3.10 and a pseudo-code implementation is given in Algorithm 1. The higher the index, the less stable an explanation is. This index may be used for both LIME and SHAP explanations and serves as a measure of how stable the explanations provided are across different predictions. Using this index, one can also analyze whether there is any relation between stable explanations and the accuracy of a prediction.

3.4.5 Hybrid Predictions

It was noted that the LSTM often mispredicted values which were actually 0 as high values whilst the ARIMA model predicted these values correctly. In order to tackle this issue, the ARIMA model was also run. When the prediction

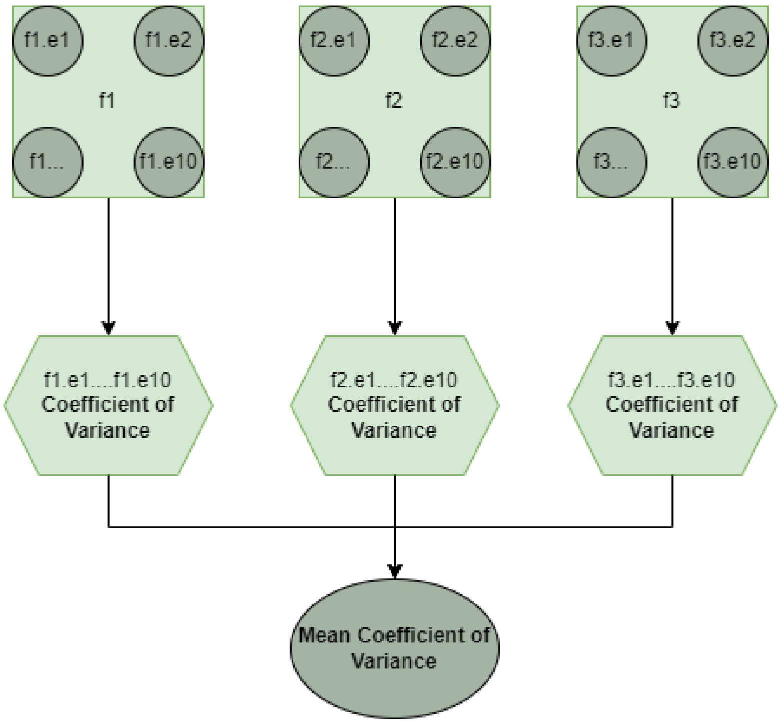

f1.e1
f1.e2
f1
f1...
f1.e10
f2.e1
f2.e2
f2
f2...
f2.e10
f3.e1
f3.e2
f3
f3...
f3.e10
f1.e1....f1.e10 Coefficient of Variance
f2.e1....f2.e10 Coefficient of Variance
f3.e1....f3.e10 Coefficient of Variance
Mean Coefficient of Variance

Figure 3.10: Process used in order to calculate the stability/confidence of explanations for predictions.

Algorithm 1 Explanation Stability/Confidence Index Calculation Algorithm

```
$var\_list \leftarrow []$
**for all** $int : feature \in features$ **do**
    $f_{std} \leftarrow standard\_deviation(feature\_explainability)$
    $f_{mean} \leftarrow mean(feature\_explainability)$
    $f_{var} \leftarrow \frac{f_{std}}{f_{mean}} * 100$
    $var_list \leftarrow var\_list.append(abs(f_{var}))$
**end for**
$var\_index \leftarrow mean(var\_list)$
```

generated by the ARIMA model was less than one, the minimum prediction was taken, whilst if it was more than one, the median was taken. This was done with the aim of unskewing the high predictions being produced by the LSTM whilst still obtaining explanations.

3.5 User Interface Prototype

In order to evaluate the explainability part of this study, a very basic User Interface (UI) was created as a prototype of a system which users would use to view predictions and explanations. This prototype was created in the form of a web application and was developed using the python Flask [6] library. Figure 3.11 shows the basic design of the web application, which describes the user inserting a customer reference, the data for that customer being extracted from a csv file containing all the raw data, the creation of the time-series, training of a model for that customer, the prediction and explanation cycles and finally, the output of the results to the web application for the user to analyze.

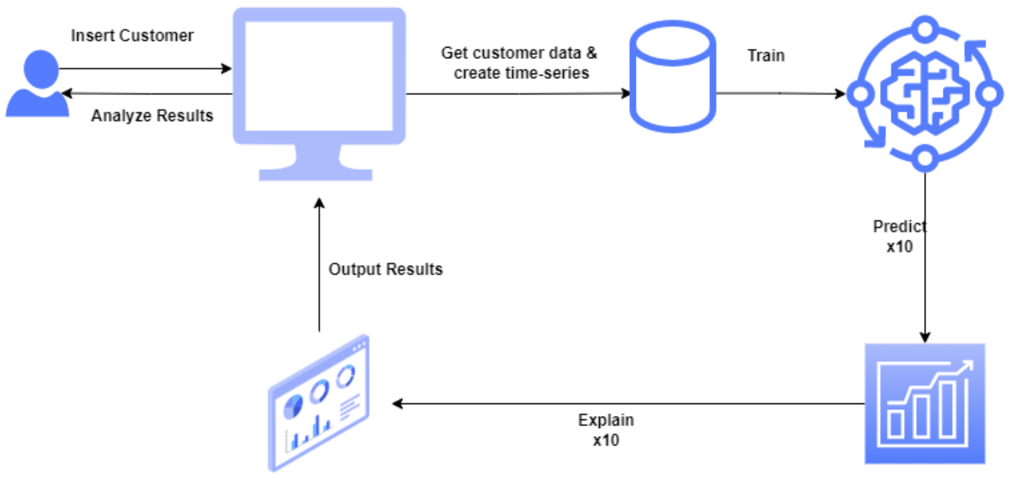


Figure 3.11: Web Application prototype design.

3.5.1 Other Experiments

The initial plan was to implement the model using AWS' Amazon Forecast [7]. This is a service offered by Amazon Web Services which delivers very accurate time-series forecasts. The plan was to train the Amazon Forecast model on the whole dataset for all customers. The following three datasets were created:

1. *Target time-series data* - contained the successful deposits amount column.
2. *Item metadata data* - contained the data corresponding to the customers' countries.
3. *Related time-series data* - contained all the other numerical columns in the time-series.

[6]https://flask.palletsprojects.com/en/2.1.x/
[7]https://docs.aws.amazon.com/forecast/latest/dg/what-is-forecast.html

A sample of the data was used to train the predictor and generate forecasts, however this implementation was deemed too costly, as the AWS free tier was exceeded even when using only a small sample of customer data.

3.6 Conclusion

To summarize, a solution was created by extracting and analyzing data, creating a time-series dataset, fitting a model to this dataset and generating explanations for the model predictions. The following chapter discusses the results produced from the experiments conducted.

Chapter 4 - Results and Discussion

This chapter analyses the results obtained through the experiments conducted, described in the previous chapter. The results obtained include results from experiments related to the machine learning model, others from experiments on their explainability and finally the user interface prototype.

4.1 Model Results

In this section, results related to the configuration and training of all the models will be discussed.

4.1.1 Model Selection Experiment Results

In order to determine what data and model configurations produce the best results, several tests were performed, the results of which will be presented in this subsection.

One-layer vs. Two-layer

The test was run using different model parameters and for the large majority of times, the one-layer model performed the same as the two-layer model. This was most likely due to the dataset being relatively small. Figure 4.1 shows a summary of the RMSE results of some of the grid-searches performed. The labels on top of each chart state whether the model used was one-layer or two-layer. Since the two models produced very similar results, the one-layer model was preferred since it was less complex than the two-layer one and required less computations, therefore less time to train.

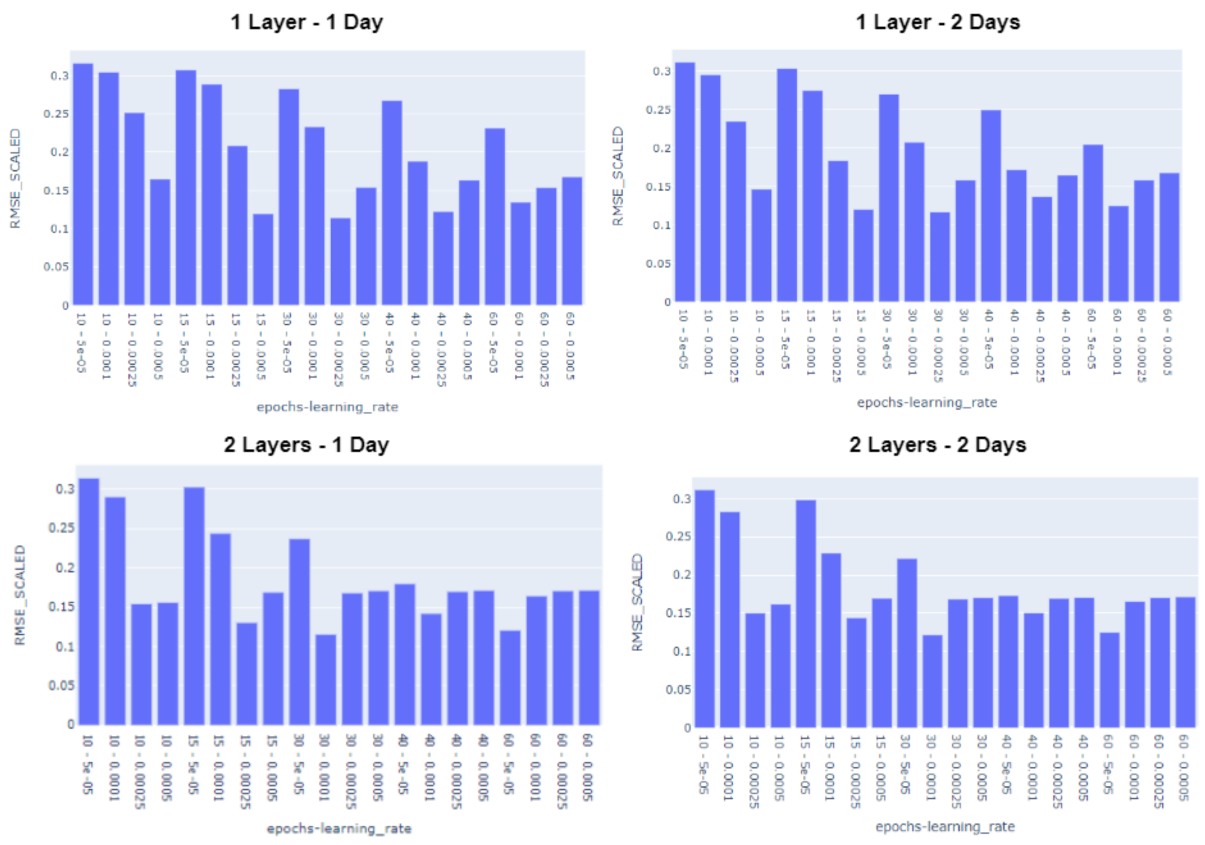


Figure 4.1: Charts showing the RMSE values of the different gridsearch results.

One-day window vs. Two-day window

As may be seen in Figure 4.1, the models trained using the two-day window did not provide any significant improvement upon the one-day time window. The models trained using one-day time window dataset outperformed the models using the two-day time window dataset or performed just as well. Due to this, the one-day time window dataset was chosen as the two-day time window dataset amounts to almost double the size of this dataset.

Data Segmentation

Figure 4.2 shows the results of the models trained separately on the datasets of customers depositing on different buckets of dates. As expected, models for customers depositing on more days have a lower error since the model essentially has more data to train on.

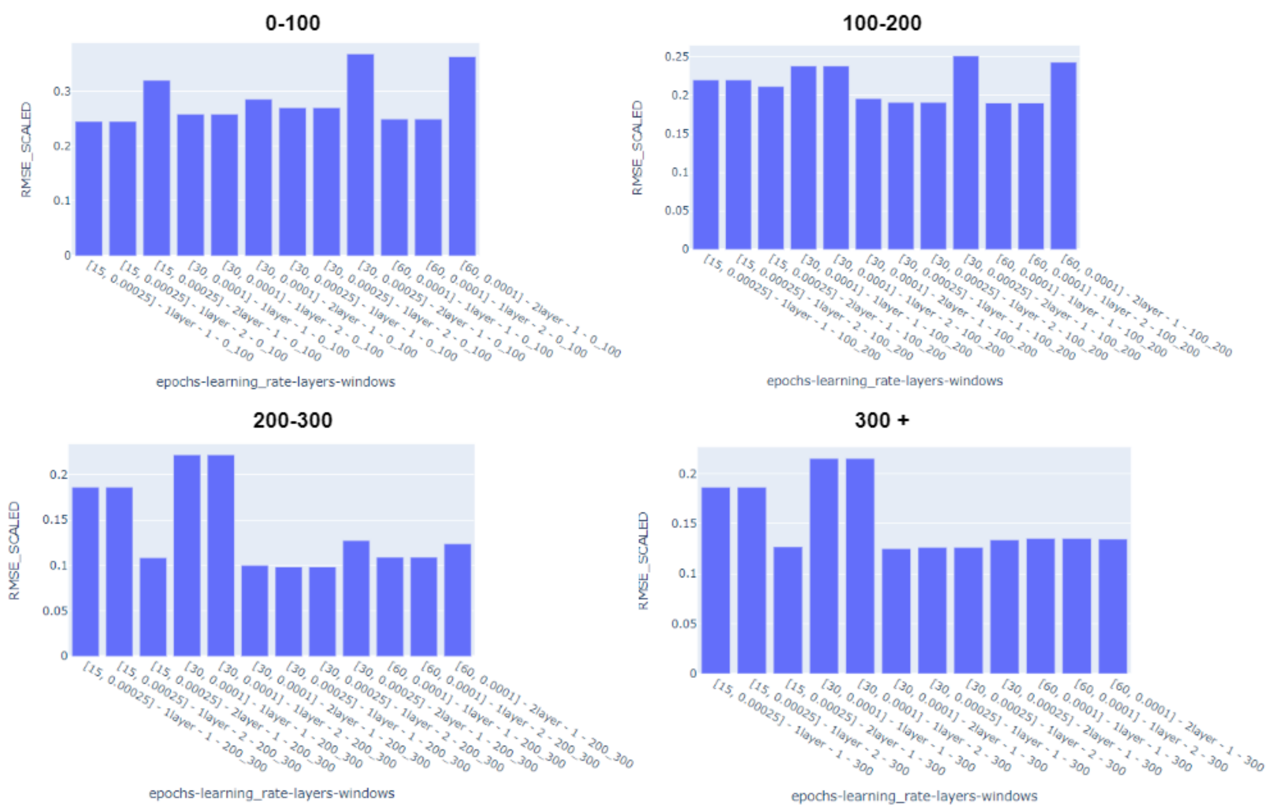


Figure 4.2: Charts showing the different RMSE values for the different segmentation datasets. RMSE is displayed on the y-axis whilst the epoch, learning rate, layers and window values are displayed on the x-axis.

Learning Rate and Epochs

Following the results shown in 4.2, a learning rate of 0.00025 and a total of 15 epochs proved to be the best choice for customers who made deposits on less than 100 days. On the other hand, 30 epochs were the best selection overall for the customers in the other segments.

Hidden Nodes

The number of hidden nodes tested out on each data segment were 64, 128 and 256. Figure 4.3 shows the results of these tests. One can note that for all segments besides the 0-100 deposit dates segment, the model with 128 hidden nodes performed best. For the 0-100 deposit dates segment, the model with 64 hidden nodes produced the best results. The reason for this is likely due to there being less data for such customers, which would require a less deep and complex model.

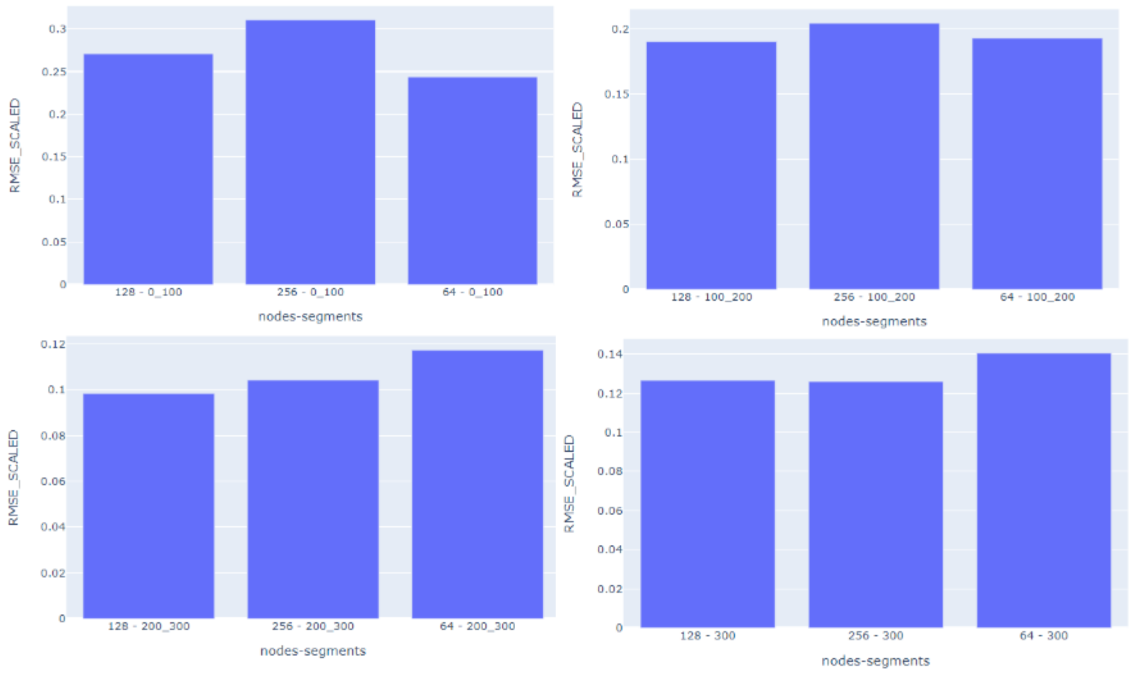


Figure 4.3: Charts showing the different RMSE values for different numbers of hidden nodes on the different data segment datasets. The RMSE is displayed on the y-axis whilst the number of nodes of each segment is displayed on the x-axis. (Top left - 0-100 deposit dates, Top right - 100-200 deposit dates, Bottom left - 200-300 deposit dates, Bottom right - 300+ deposit dates).

Prediction Selection

In order to determine whether the median or the prediction nearest to the mean is best to be taken as the prediction value, the RMSE of both values was calculated against the actual values on a sample of the data. Figure 4.4 shows the median and nearest mean RMSE values of the data in this sample. One can note that almost all values are equal, meaning that the predictions are also equal. Therefore the mean was chosen as the prediction value since it requires less computation to extract.

4.1.2 Comparison against benchmark model implementations

The performance of the created model was compared to the performance of the Naive and ARIMA models on the same predictions. This way, one may deter-

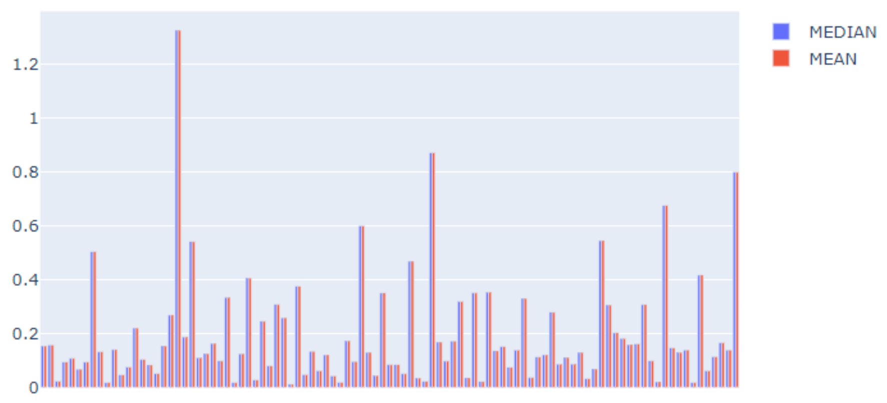


Figure 4.4: Chart showing the median and nearest mean values of each prediction.

mine whether the created model is able to outperform simpler models.

0-100 deposit dates segment

Since this data segment is the one which contains the most sparse data, it is the one on which the model is expected to perform the worst out of all the customer segments. Table 4.1 shows the RMSE results for the different predictions. Across all values, the Naive model has the least mean RMSE value. This is most likely caused because most correctly predicted values are zero. These customers are ones who deposited less than a hundred days a year, meaning that the target value is more often zero than not. Figure 4.5 shows the actual and different predicted values. The black line is the x=y line, meaning that the closer the points are to this line, the closer the predicted and actual values are. From these plots, one can confirm that most actual values are 0 or close to 0 and that most Naive predictions are 0, even when the actual value is not 0.
On the other hand, when excluding actual values equal to 0, one can note that the Naive model's RMSE increases drastically and the RMSE of the mean prediction of the LSTM is now the least of all the values. For actual values greater than 100, it is clear that the LSTM outperforms the statistical methods.

Another evaluation test performed on the data was to see how many times each prediction scored the least RMSE of all predictions. Figure 4.6 shows the results of these tests as barcharts. Similar to what can be seen in the previous results, the LSTM performs better for higher deposit amounts.

	All values	Values >0	Values >=100
Naive RMSE	0.09	0.42	0.61
ARIMA RMSE	0.10	0.36	0.51
Final LSTM RMSE	0.38	0.39	0.37
LSTM Median RMSE	0.48	0.34	0.37
LSTM Mean RMSE	0.40	0.30	0.35
LSTM Min RMSE	0.13	0.42	0.58
LSTM Max RMSE	0.49	0.34	0.36

Table 4.1: RMSE results for the 0-100 deposit dates customer segment.

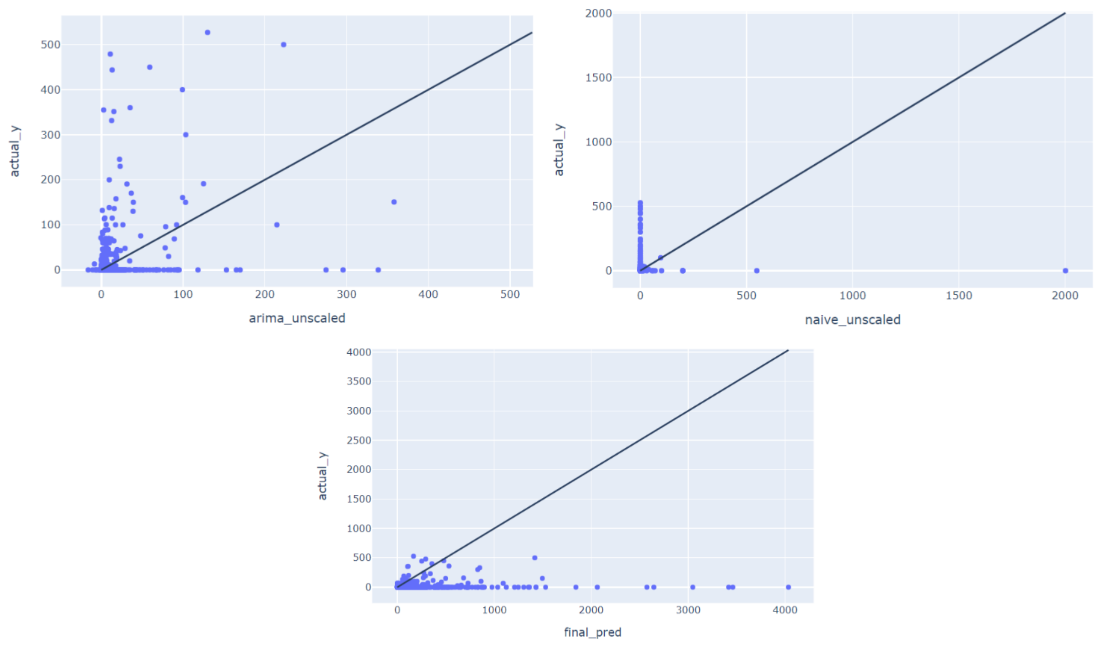


Figure 4.5: The actual (y-axis) vs predicted (x-axis) values for the 0-100 dataset from the ARIMA model (left), Naive model (right) and final LSTM prediction values (bottom)

100-200 deposit dates segment

For the data segment of customers depositing between 100 and 200 days, we can now note the the differences between the RMSE of the statistical methods and the LSTM are less than those observed for the 0-100 deposit dates customer segment. These results are shown in Table 4.2.

For all amounts, the ARIMA model outperforms the other models, however the difference between the RMSEs of the ARIMA prediction and the final LSTM prediction gets less as the amounts get higher. This means that the LSTM is better at predicting higher deposit amounts than lower ones. When compared to the values corresponding to the 0-100 segment dataset, the Final LSTM RMSE

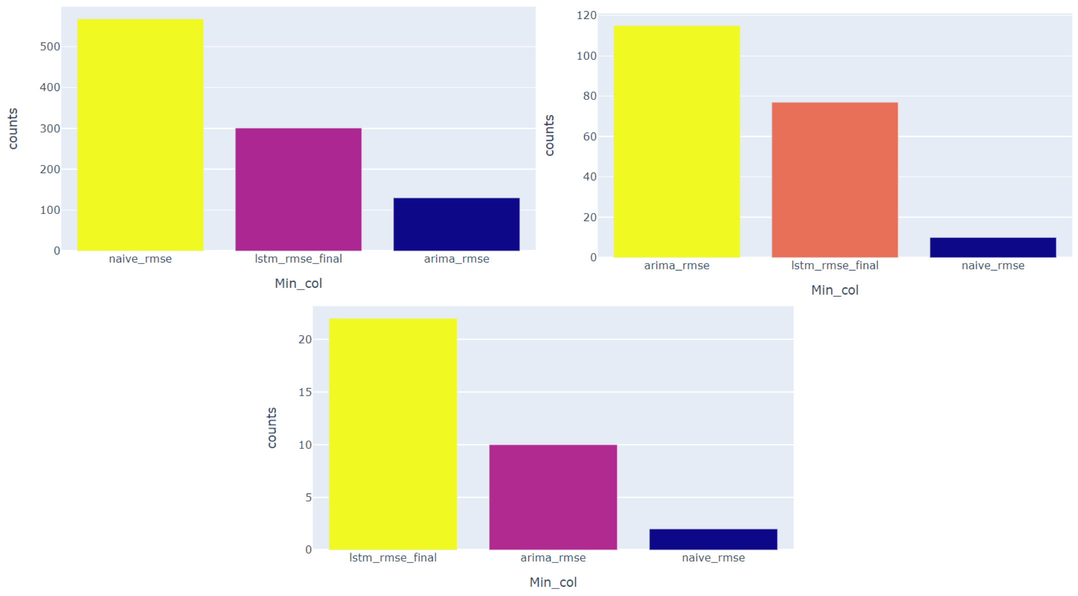


Figure 4.6: The counts of predictions with the minimum RMSE for the 0-100 dataset for : 1. all the data (left), 2. all data where the actual target value is not 0 (right) 3. values where the actual target values is greater or equal to 100 (middle). The y-axis shows the counts whilst the x-axis shows the model.

values are lower, except for when values are above 100. Figure 4.7 shows the

	All values	**Values >0**	**Values >=100**
Naive RMSE	0.14	0.27	0.47
ARIMA RMSE	0.13	0.19	0.38
Final LSTM RMSE	0.19	0.23	0.38
LSTM Median RMSE	0.20	0.23	0.38
LSTM Mean RMSE	0.18	0.22	0.39
LSTM Min RMSE	0.13	0.26	0.46
LSTM Max RMSE	0.20	0.23	0.38

Table 4.2: RMSE results for the 100-200 deposit dates customer segment.

prediction versus actual values for the different models. When compared to the scatter plots in Figure 4.5, we can see that the deposit amount values are higher. One can note that the Naive model returns a lot of 0 predictions. This is due to the fact that it doesn't take anything into consideration besides the previous day's deposits. Another observation made was that the LSTM makes predictions which are too high more often that the ARIMA model does. Figure 4.8 depicts the count of times each model produced the minimum RMSE value. One may note that over all the data, the Naive model still produced forecasts with the lowest RMSE more times than the other models. Since the mean RMSE of the

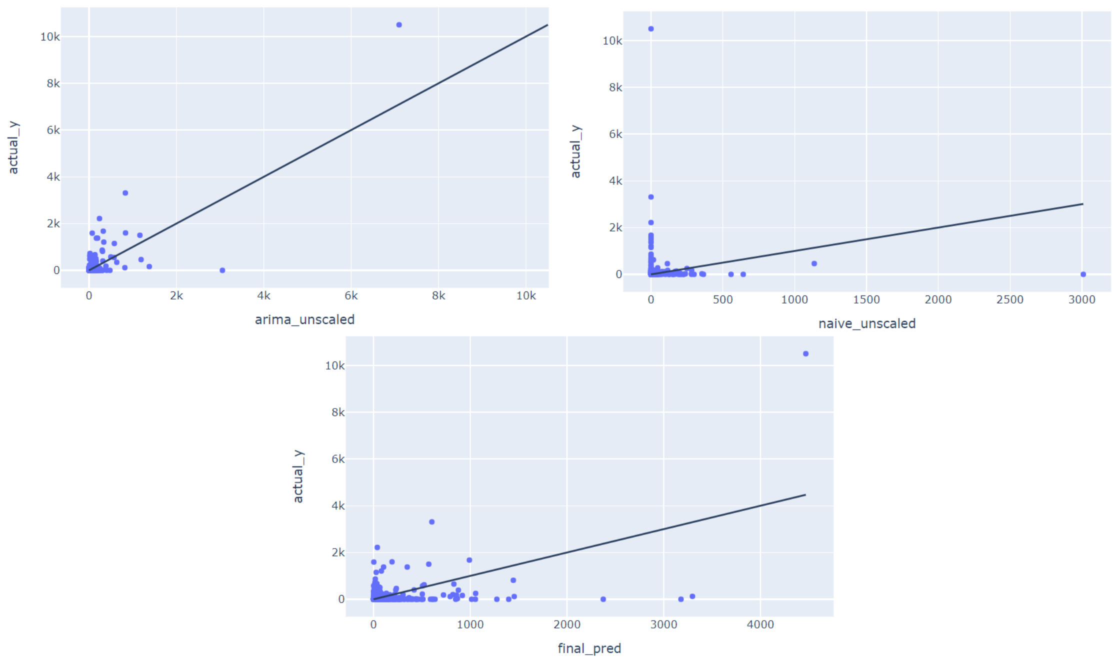


Figure 4.7: The actual (y-axis) vs predicted (x-axis) values for the 100-200 dataset from the ARIMA model (left), Naive model (right) and final LSTM prediction values (bottom)

ARIMA model is actually higher, this implies that the differences between the predicted and actual values where the Naive model does not produce the lowest RMSE are higher. When taking only deposits above 0 as well as when only taking deposits above 100, the ARIMA model generates the best results, with the LSTM coming in second and the Naive model showing very poor performance, implying that most correctly predicted values were due to amounts that were 0.

200-300 deposit dates segment

For predictions for customers who deposited 200 to 300 dates in a year, it is noted in Table 4.3 that the ARIMA model outperforms the other models for all predictions being taken into consideration. For such customers, one can also see that the Naive model performs very poorly. Once again, when compared to the previously analyzed segments, the LSTM RMSE is lower for this segment.

As seen in Figure 4.9, the LSTM once again predicts values which are too high when compared to the values predicted by the ARIMA model and the Naive

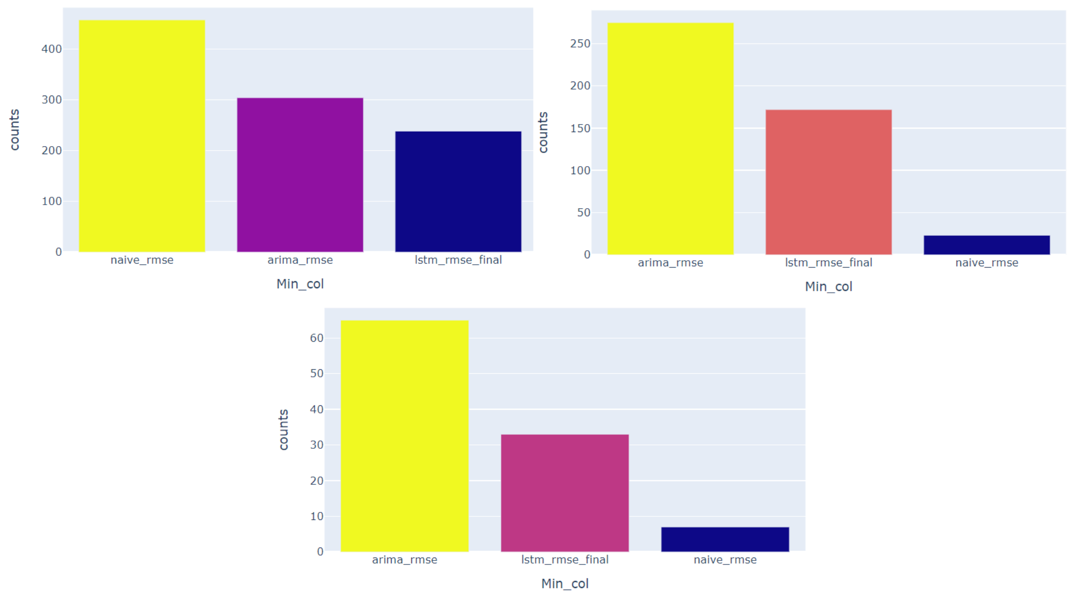


Figure 4.8: The counts of predictions with the minimum RMSE for the 100-200 dataset for : 1. all the data (left), 2. all data where the actual target value is not 0 (middle) 3. values where the actual target values is greater or equal to 100 (bottom). The y-axis shows the counts whilst the x-axis shows the model.

	All values	**Values >0**	**Values >=100**
Naive RMSE	0.17	0.24	0.33
ARIMA RMSE	0.12	0.14	0.22
Final LSTM RMSE	0.19	0.19	0.30
LSTM Median RMSE	0.19	0.19	0.30
LSTM Mean RMSE	0.17	0.19	0.30
LSTM Min RMSE	0.16	0.23	0.34
LSTM Max RMSE	0.19	0.19	0.30

Table 4.3: RMSE results for the 200-300 deposit dates customer segment.

model predicts too many zeros. When analyzing the count of times the minimum RMSE was produced by each model, as shown in Figure 4.10, the ARIMA model once again produces the best results. When excluding zero values, the LSTM produced better results than the Naive model. For zero values, the results obtained by the Naive model and the LSTM were also very close.

300+ deposit dates segment

Table 4.4 shows the results for customers depositing on more than 300 days a year. This customer segment is the one which contains most data. Although the best results are once again produced by the ARIMA model, we can note another

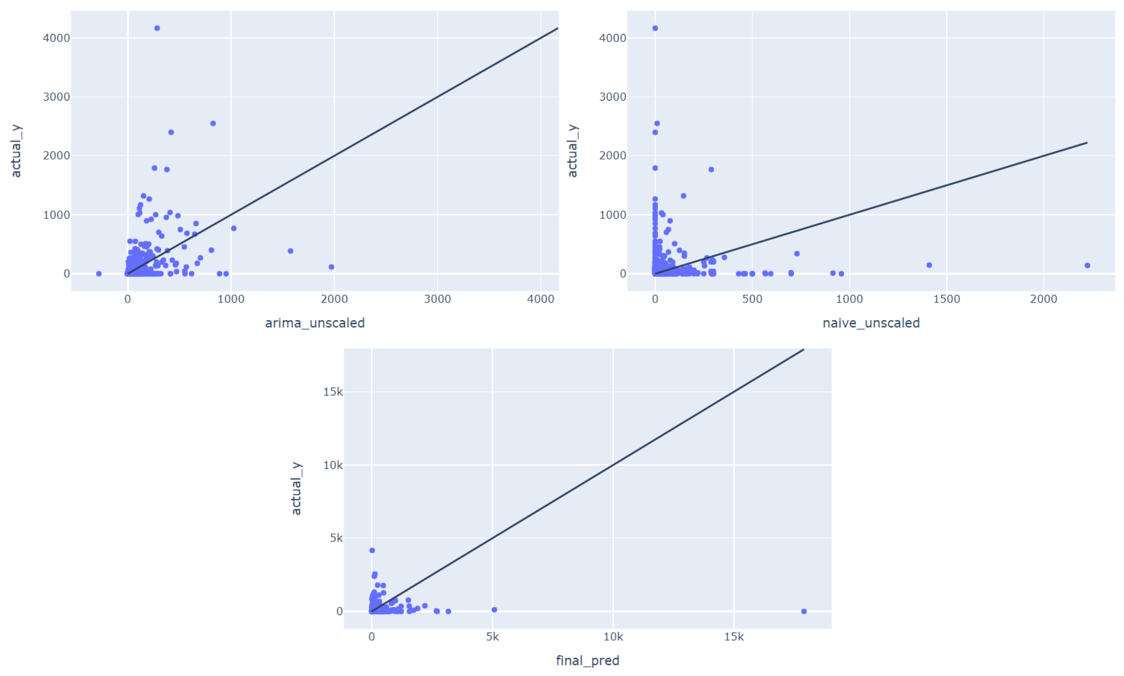


Figure 4.9: The actual (y-axis) vs predicted (x-axis) values for the 200-300 dataset from the ARIMA model (left), Naive model (right) and final LSTM prediction values (bottom)

drop in the LSTM RMSE when compared to the other segments. This drop is not as drastic for the ARIMA model, implying that the amount of data has a larger impact on the LSTM model than on the ARIMA model. The scatter plots

	All values	**Values >0**	**Values >=100**
Naive RMSE	0.21	0.24	0.29
ARIMA RMSE	0.13	0.12	0.17
Final LSTM RMSE	0.17	0.16	0.21
LSTM Median RMSE	0.17	0.16	0.21
LSTM Mean RMSE	0.17	0.16	0.22
LSTM Min RMSE	0.20	0.24	0.31
LSTM Max RMSE	0.18	0.16	0.21

Table 4.4: RMSE results for the 300+ deposit dates customer segment.

shown in Figure 4.11, show that the models produce predictions similar to those produced for the other segments, with the naive model predicting too many zeros and the LSTM model predicting values which are too high.

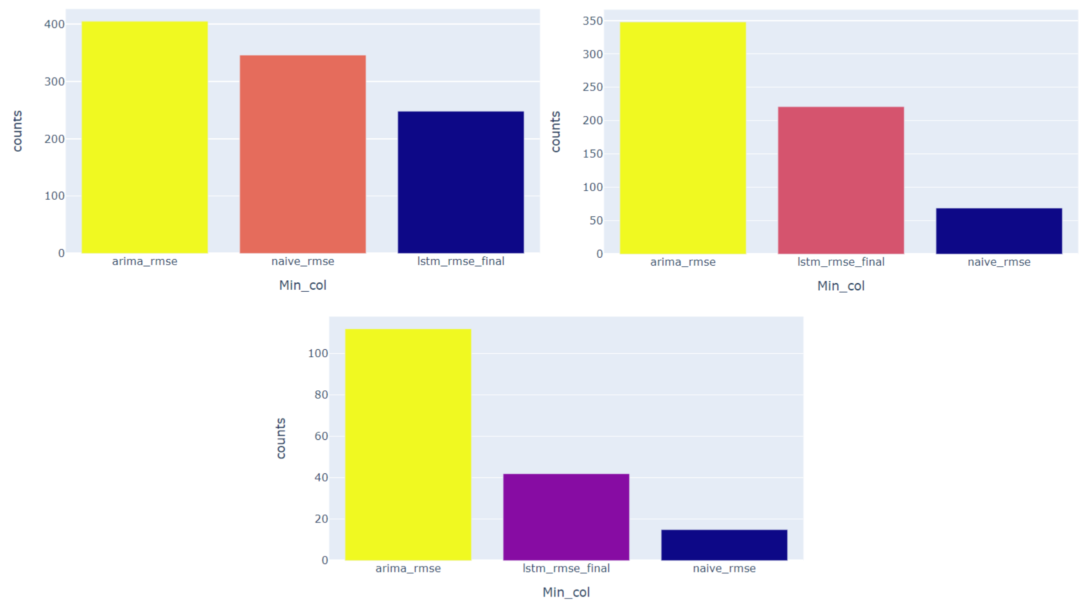


Figure 4.10: The counts of predictions with the minimum RMSE for the 200-300 dataset for : 1. all the data (left), 2. all data where the actual target value is not 0 (right) 3. values where the actual target values is greater or equal to 100 (bottom). The y-axis shows the counts whilst the x-axis shows the model.

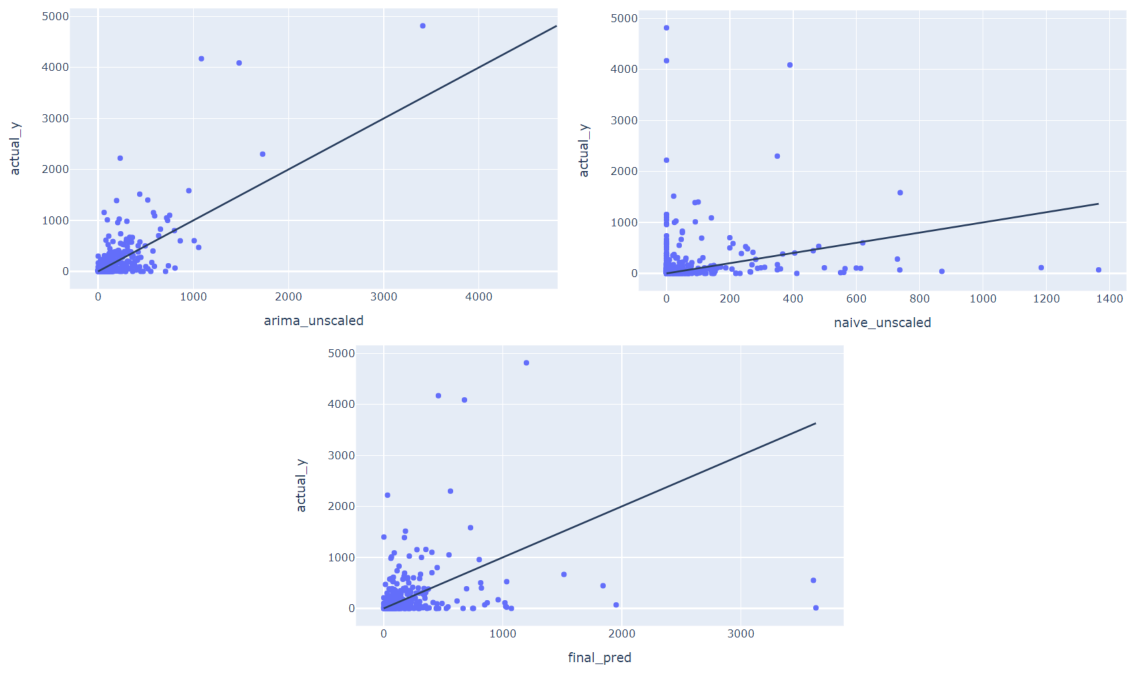


Figure 4.11: The actual (y-axis) vs predicted (x-axis) values for the 300+ dataset from the ARIMA model (left), Naive model (right) and final LSTM prediction values (bottom)

Similar results to the previous segments may also be seen for the lowest

RMSE counts in Figure 4.12, where the ARIMA model is superior to the rest and the LSTM and Naive results are almost equal across all data and the LSTM produces better results than the Naive model for values greater than 0.

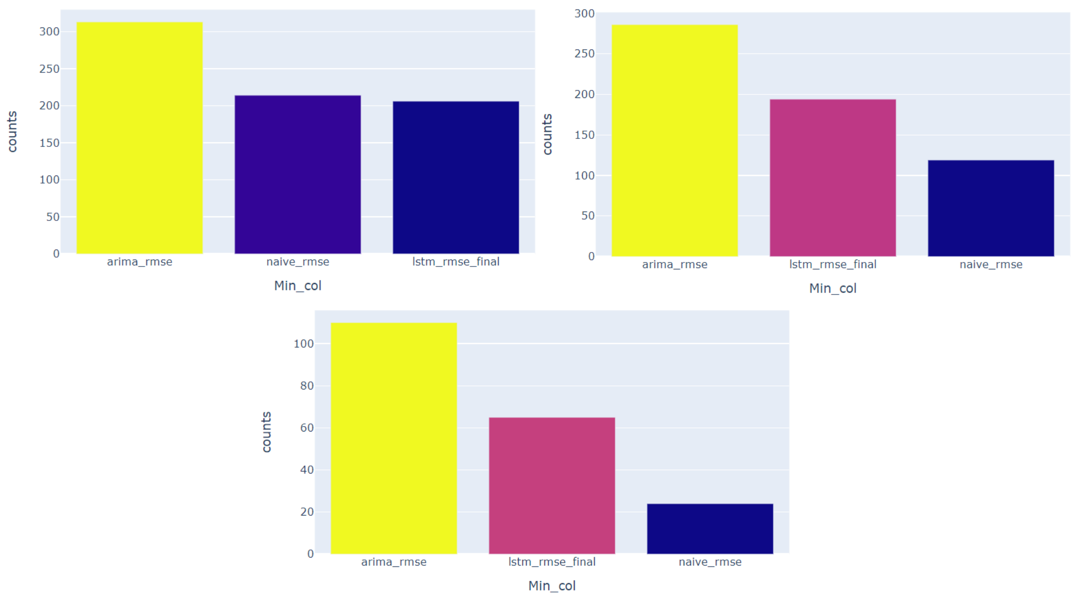


Figure 4.12: The counts of predictions with the minimum RMSE for the 300+ dataset for : 1. all the data (left), 2. all data where the actual target value is not 0 (middle) 3. values where the actual target values is greater or equal to 100 (right). The y-axis shows the counts whilst the x-axis shows the model.

4.2 Explainable AI Results

In this section, the results of the interpretability experiments conducted for this study will be discussed.

4.2.1 LIME

In order to visualize the results of the LIME explanations, the values are displayed as a plotly barchart, with negative values attributed to the features shown in red and positive values attributed to the features shown in green. An example of such an explanation is shown in Figure 4.13.

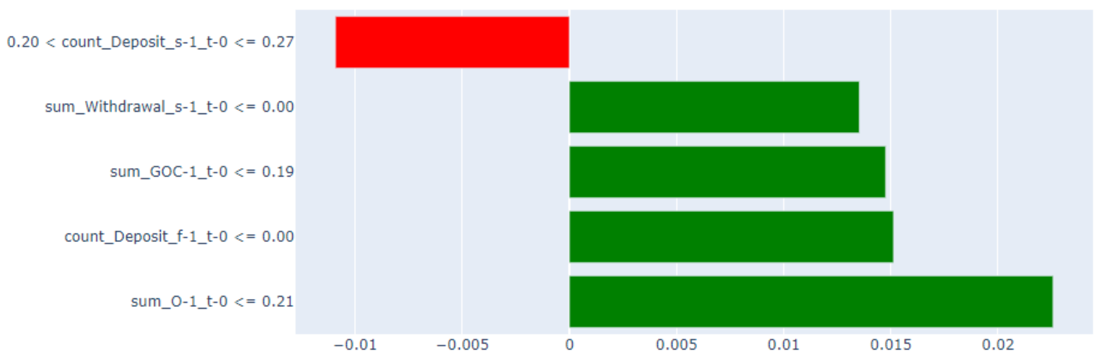


Figure 4.13: LIME results shown in the form of a plotly barchart.

4.2.2 SHAP

A similar chart output as that for LIME values is generated for SHAP values, shown in Figure 4.14. These visualizations were used rather than the ones provided in the LIME and SHAP libraries so that the charts for the two explanations are the same. This is required as during the evaluation, users will be asked which explanations they prefer, as will be detailed in the evaluation section. By using the same charts, the users may assess the quality of the values, rather than make decisions based on the visualizations of the values.

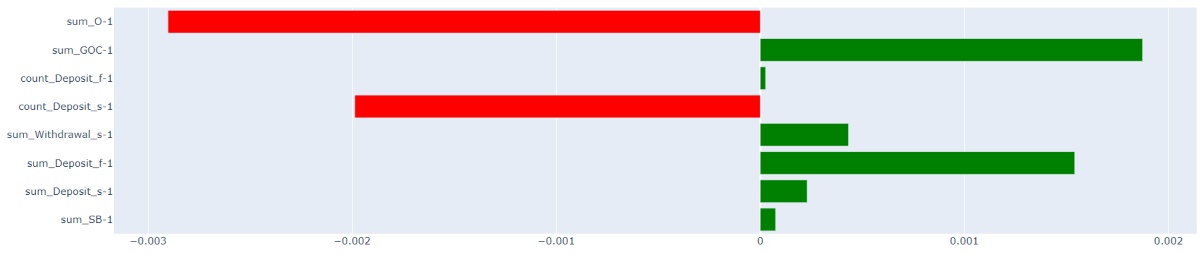


Figure 4.14: SHAP results shown in the form of a plotly barchart.

4.2.3 Model Confidence

The model confidence explainability method produces a number of values from the different predictions made in order to increase the transparency of the model. Table 4.5 shows these values and their description.

A scatter plot showing the distribution of the predictions was also created. An example of such a plot can be seen in Figure 4.15, which shows that a prediction

Name	Description
Minimum Prediction	Prediction having the lowest amount.
Maximum Prediction	Prediction having the highest amount.
Mean Prediction	Sum of all predictions over the count of predictions (10).
Median Prediction	Middle value of predictions.
Standard Deviation	Probability distribution measure of the predictions.
Coefficient of Variation	Ratio of standard deviation to the mean.

Table 4.5: Table showing the model confidence metrics and their descriptions.

of 14.7 was made seven times and a prediction of 0 was made three times.

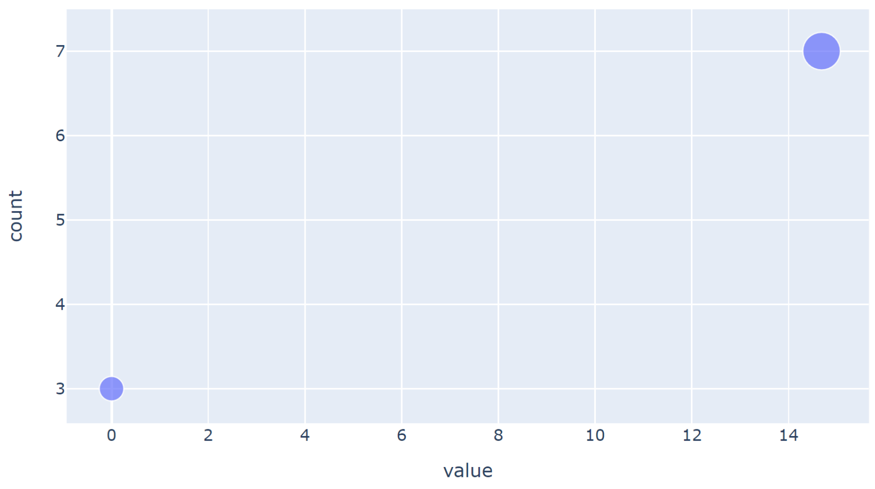


Figure 4.15: Scatter plot showing the distribution of predictions.

4.2.4 Explainability Confidence

In order to determine how much a user should trust an explanation, the explainability confidence index was created. Table 4.6 shows the mean index for LIME and SHAP for each data segment. The SHAP index is constantly less than the LIME index, meaning that SHAP explanations are more stable. This behaviour is expected, since SHAP uses Shapely values, which produce a global interpretation of the model, thus making these explanations more stable.
Another interesting observation is that the LIME index decreases as the number of depositing days increases. Having more depositing days means that most likely, more data is available. This highlights the effect that the amount of non-

zero data available has on the LIME explanations. Another experiment held

	LIME Index	SHAP Index
0-100	428.15	37.54
100-200	141.98	16.61
200-300	119.42	72.03
300+	66.73	17.21

Table 4.6: Table showing the explainability confidence indices for the different customer segment predictions.

using this index was to determine whether any correlation exists between the value of this index and the accuracy of a prediction. Table 4.7 shows the correlation results between the LSTM RMSE and explainability confidence index for each data segment. The correlation values are all relatively small, however one may note that for LIME, the correlation increases as the number of depositing days of a segment increases and all correlation values are positive. On the other hand, the correlation values for the SHAP explanations contain a mixed polarity and do not show a consistent ascending or descending pattern with the number of depositing days.

	LIME Corr	SHAP Corr
0-100	0.03	0.07
100-200	0.05	0.04
200-300	0.07	-0.06
300+	0.10	-0.04

Table 4.7: Table showing correlation between the different explainability confidence indices and the LSTM RMSE for the different customer segment predictions.

The indices were also plotted along with the predicted vs actual values, as shown in Figures 4.16 and 4.17. The value of the indices is represented by the size of the marker. These plots confirm what was indicated by the correlation values since in most cases, there does not seem to be any strong relationship with how close the actual and predicted values are to the size of the marker. The strongest correlation is seen for the LIME values of the 300+ data segment, where the plot shows relatively small dots when the predicted value is far off from the actual value, however a strong correlation was not found since smaller markers are also shown for forecasts which are close to the actual values.

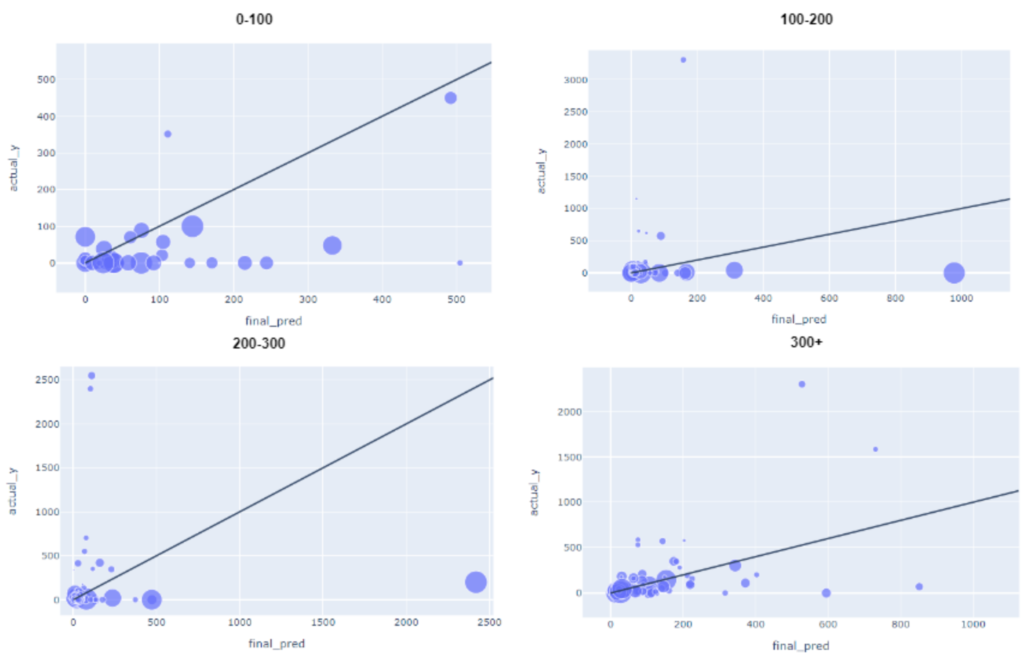


Figure 4.16: Plot of the actual values (y-axis) against the LSTM predictions (x-axis), with the size of the marker indicating the LIME explanation confidence index values.

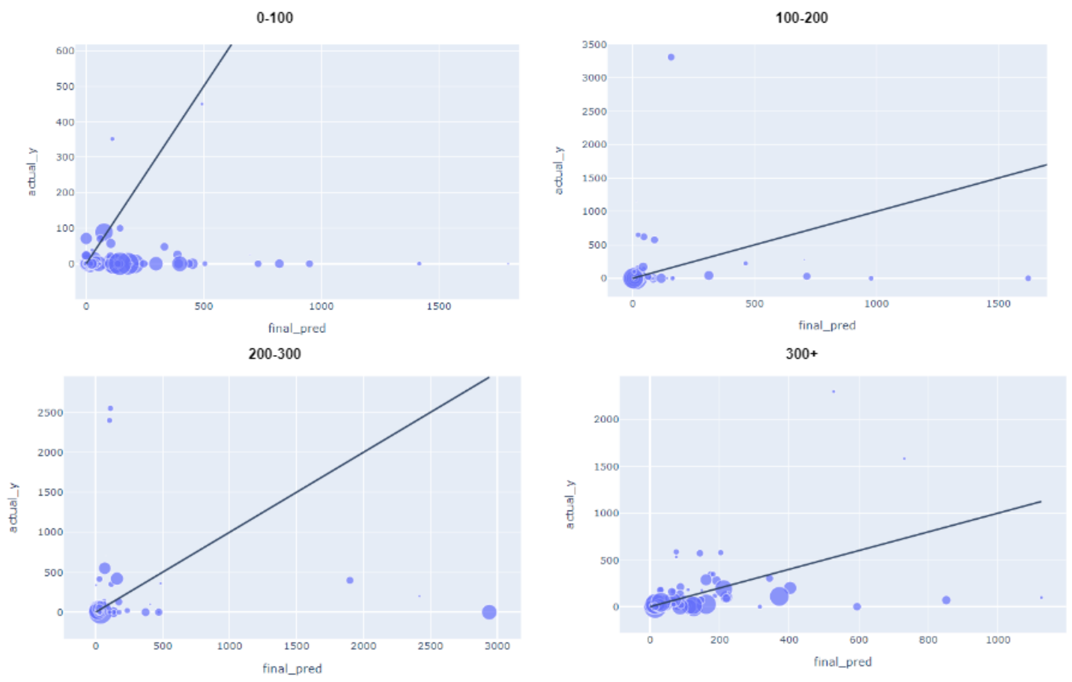


Figure 4.17: Plot of the actual values (y-axis) against the LSTM predictions (x-axis), with the size of the marker indicating the SHAP explanation confidence index values.

Figure 4.18 shows an example of some of the different explanations generated where the LIME Explanation Confidence Index value is at a low value of 7.2. Figure 4.19 shows some of the SHAP explanations for the same customer. The SHAP value is also low, having a value of 3.5.

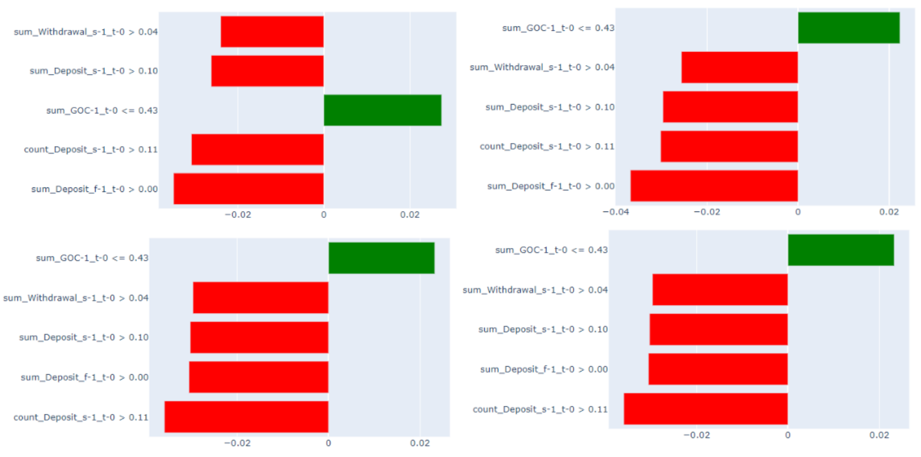


Figure 4.18: Different LIME explanations for a prediction with a good explanation confidence index.

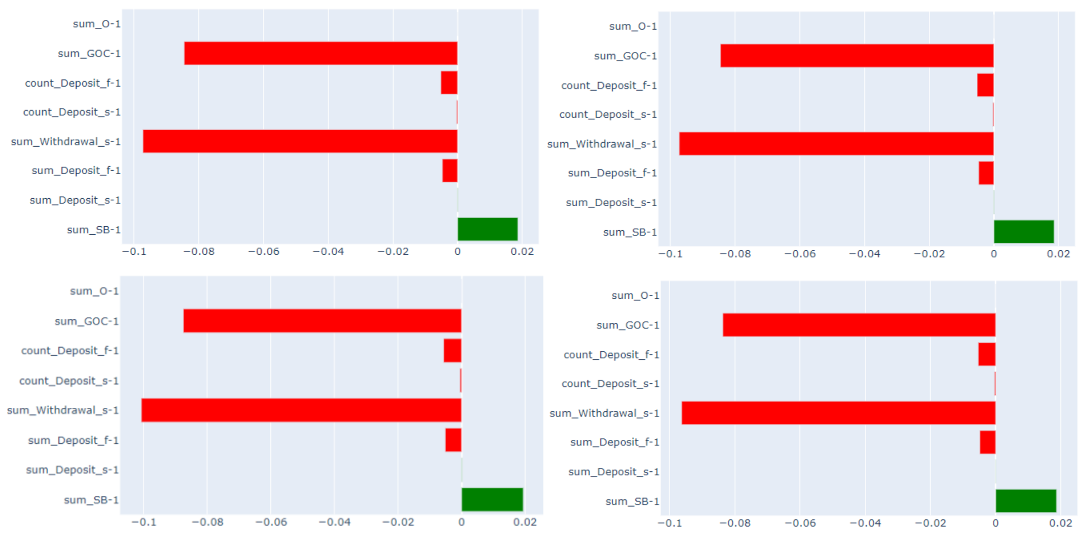


Figure 4.19: Different SHAP explanations for a prediction with a good explanation confidence index.

In contrast, Figure 4.20 shows an example of some of the different explanations generated where the LIME Explanation Confidence Index value is at a high value of 140.2. Figure 4.21 shows some of the SHAP explanations for the same customer. The SHAP value is also higher, having a value of 30.7.

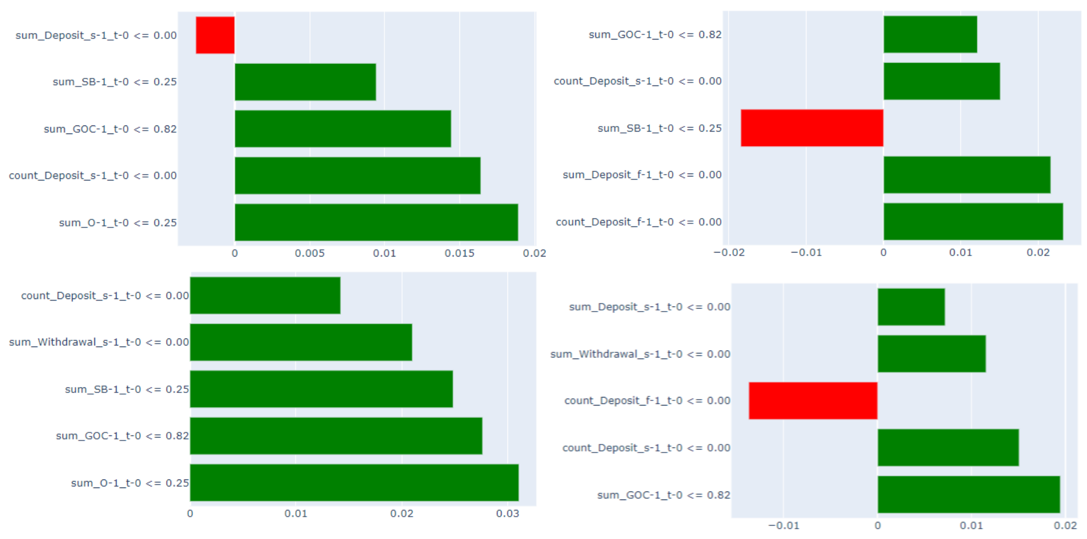


Figure 4.20: Different LIME explanations for a prediction with a bad explanation confidence index.

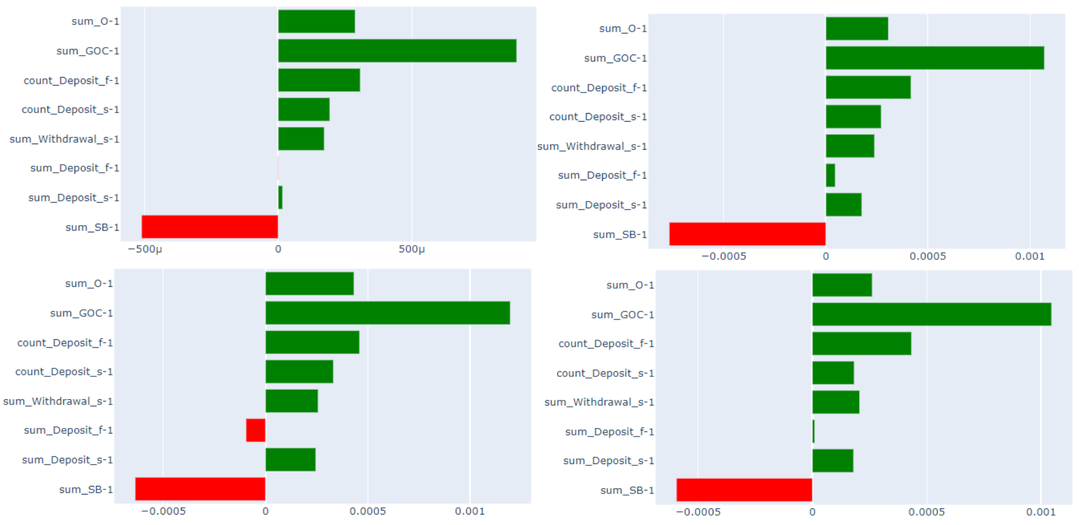
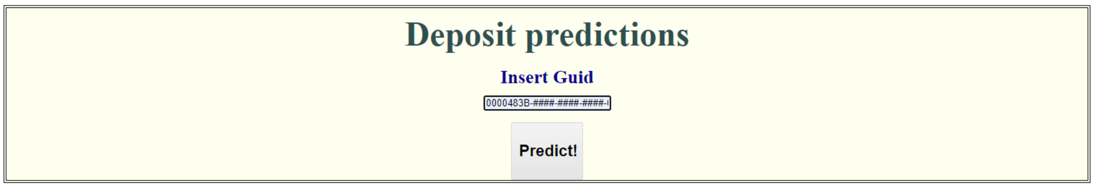


Figure 4.21: Different SHAP explanations for a prediction with an average explanation confidence index.

4.3 User Interface Prototype

In this section, a few snippets of the web application created for the purpose of evaluating this study will be shown.

Figure 4.22 shows the page in which a user may enter a customer reference for which they would like to generate predictions and explanations.

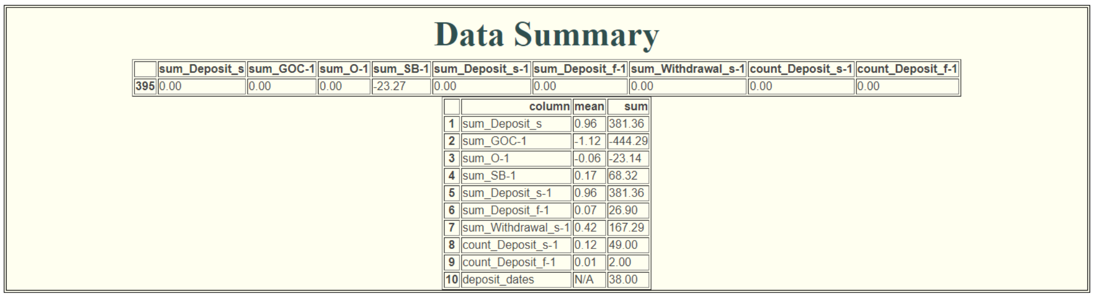


Deposit predictions

Insert Guid

Figure 4.22: Customer Entry Page

Figure 4.23 shows a tabular summary of the data available for the input customer.



Data Summary

	sum_Deposit_s	sum_GOC-1	sum_O-1	sum_SB-1	sum_Deposit_s-1	sum_Deposit_f-1	sum_Withdrawal_s-1	count_Deposit_s-1	count_Deposit_f-1
395	0.00	0.00	0.00	-23.27	0.00	0.00	0.00	0.00	0.00

	column	mean	sum
1	sum_Deposit_s	0.96	381.36
2	sum_GOC-1	-1.12	-444.29
3	sum_O-1	-0.06	-23.14
4	sum_SB-1	0.17	68.32
5	sum_Deposit_s-1	0.96	381.36
6	sum_Deposit_f-1	0.07	26.90
7	sum_Withdrawal_s-1	0.42	167.29
8	count_Deposit_s-1	0.12	49.00
9	count_Deposit_f-1	0.01	2.00
10	deposit_dates	N/A	38.00

Figure 4.23: Tabular Summary Section

Figure 4.24 shows a line graph in which the customer may select to view the different features or combination of features which are used by the model for training over time. Different features may be chosen by simply clicking on the variable from the legend on the right hand side of the plot.

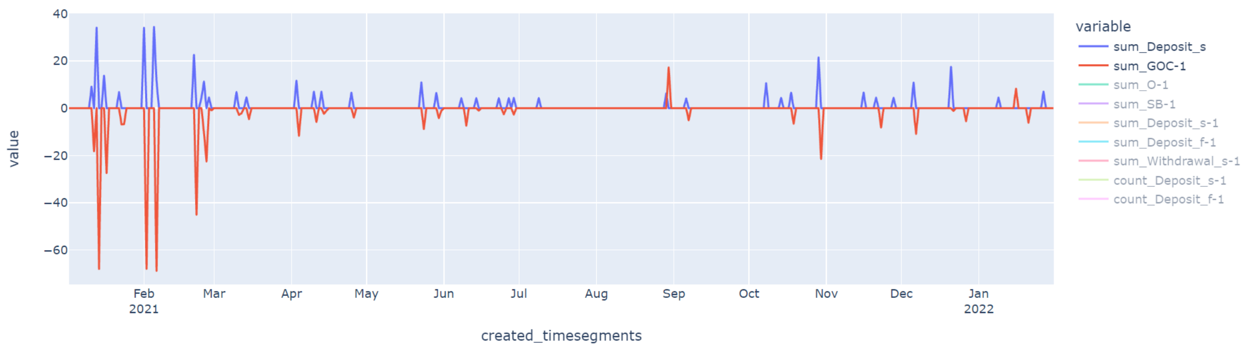
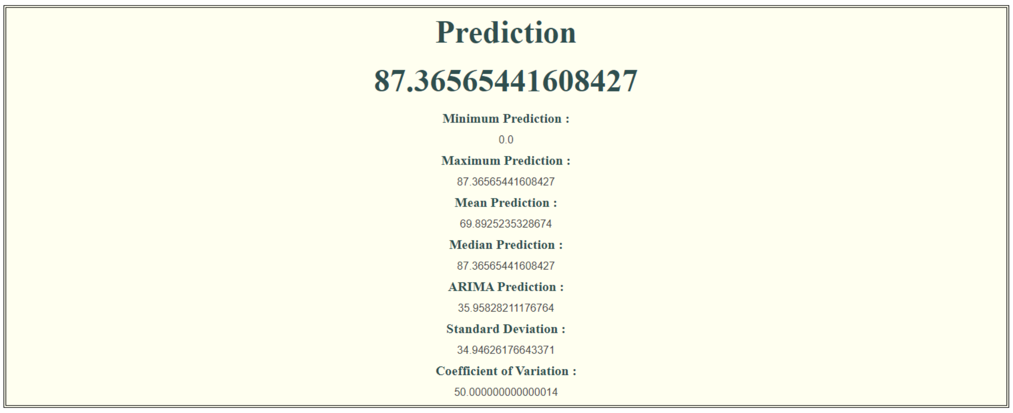


Figure 4.24: Line graph of feature values (y-axis) against time (x-axis).

Figure 4.25 shows how the values related to the model confidence are displayed. Along with this summary, the scatter plot depicted in Figure 4.15 is also output.



Prediction

87.36565441608427

Minimum Prediction :

0.0

Maximum Prediction :

87.36565441608427

Mean Prediction :

69.8925235328674

Median Prediction :

87.36565441608427

ARIMA Prediction :

35.95828211176764

Standard Deviation :

34.94626176643371

Coefficient of Variation :

50.000000000000014

Figure 4.25: Model confidence values.

Figure 4.26 shows the bar charts generated for the LIME and SHAP explanation values.

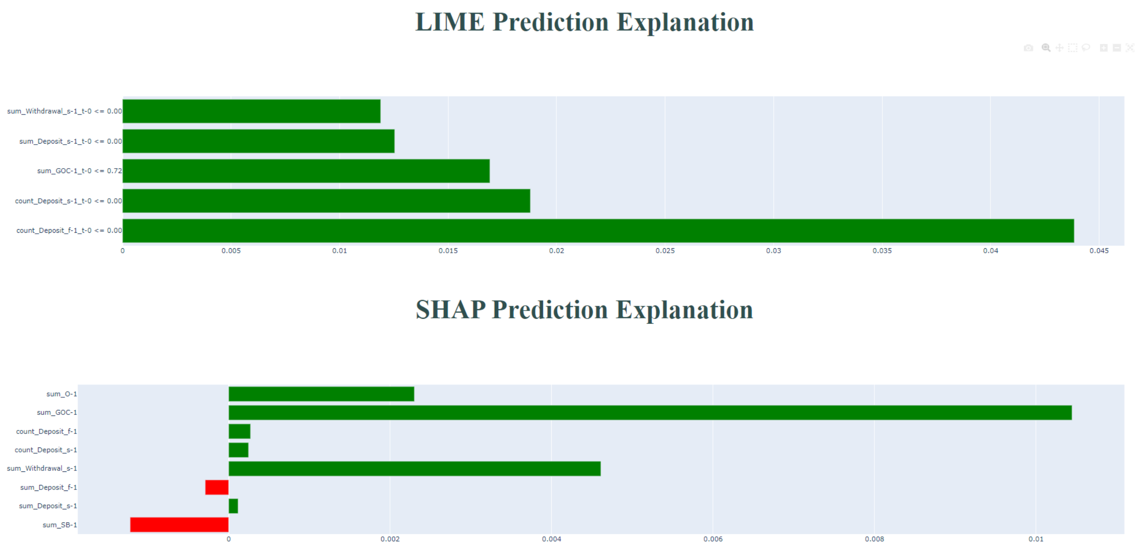
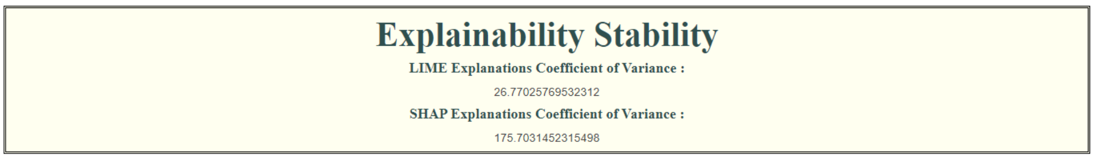


Figure 4.26: LIME and SHAP bar charts.

Figure 4.27 shows the output of the explanation stability indexes (coefficient of variance) for LIME and SHAP.



Explainability Stability

LIME Explanations Coefficient of Variance :

26.77025769532312

SHAP Explanations Coefficient of Variance :

175.7031452315498

Figure 4.27: Explanation stability indexes.

Figure 4.28 shows the box plots showing the distributions for the LIME and SHAP explanation values for the different features.

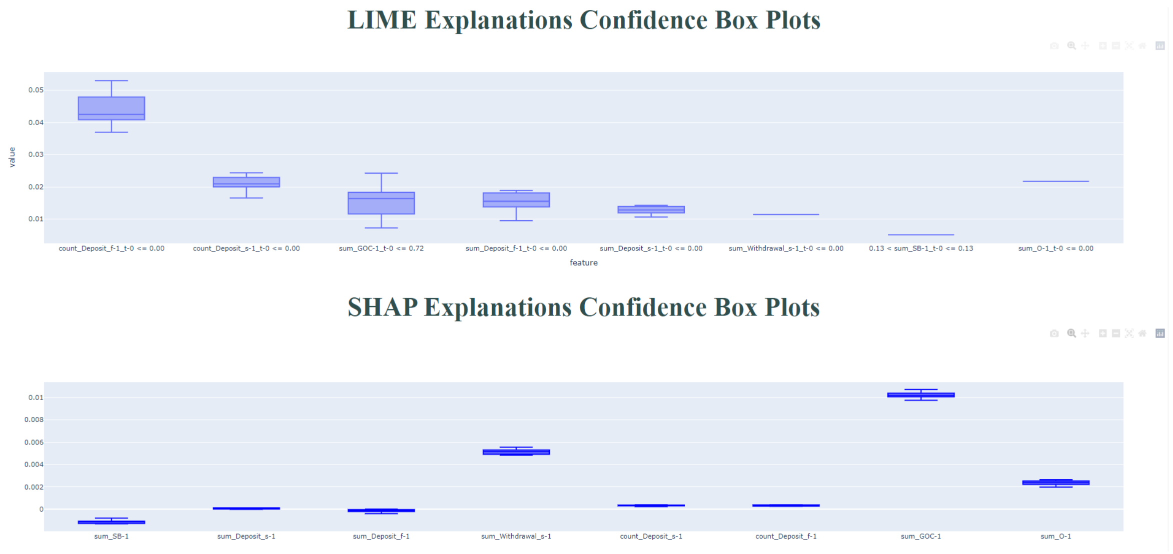


Figure 4.28: Box plots of explainability values' distributions.

4.4 Conclusion

This chapter presented the results produced by the experiments discussed in the previous chapter. Next, these results will be discussed and evaluated in order to reach a conclusion to this study.

Chapter 5 - Evaluation

In order to evaluate the results obtained through this study, both quantitative and qualitative evaluations were held. The quantitative evaluation consisted of analysing the results obtained from the model and explanation confidence indices and comparing them to results obtained from other studies. The qualitative evaluation was an expert evaluation of the system created showing the prediction and explanation results through a web application. These evaluations and their outcomes will be discussed in this chapter.

5.1 Quantitative Evaluation

5.1.1 Model Quantitative Evaluation

As described in the previous chapter, the predictions coming from the LSTM model were compared to predictions from Naive and ARIMA models. Table 5.1 shows a mean of the RMSE values obtained by each model across the different data segments.

	All values	**Values >0**	**Values >=100**
Naive RMSE	0.15	0.29	0.43
ARIMA RMSE	0.12	0.20	0.32
Final LSTM RMSE	0.23	0.24	0.32

Table 5.1: Mean RMSE results for all data segments.

The RMSE represents the standard deviation of prediction errors, signifying a normalized distance between the actual and predicted value. These values were calculated using the prediction values which were based on the dataset values scaled from zero to one. The RMSE has the same scale as these predicted

values. Since the models were run on a dataset created for the purpose of this study, comparing the results to those obtained from other studies is not very straightforward.
Yamak et al. [8] also evaluated the performance of ARIMA and LSTM models using an RMSE values, as previously shown in Figure 2.10. Although the RMSE values are not using the same scale, one can still compare the differences between the LSTM and ARIMA RMSEs in [8] and this study. Yamak et al. [8] use a univariate unscaled dataset on all models and one can note that the RMSE of the LSTM model is almost double that of the ARIMA model. The RMSE values produced by the LSTM model in this study were never as high as double the RMSE values produced by the ARIMA model. For actual values over 100, the models even produce the same RMSE.
This confirms the hypothesis made by Yamak et al. [8] that adding more data and features to the dataset improves the performance of an LSTM model, making it a better contender to the ARIMA model for time-series forecasting.

5.1.2 Explainability Quantitative Evaluation

Using the Explainability Confidence Index created in this study in order to evaluate the different explainability methods, we can conclude that LIME explanations are less stable than SHAP explanations. This result is expected since SHAP also performs global model interpretation, which would be the same for each explanation.
In [31], Man and Chan found that using their instability index, LIME was as stable as SHAP. The main difference between the index in [31] and the one introduced in this study is that whilst Man and Chan only use the ranking of features to compute the index, in this study the actual explainability values are used to compute the index.
Visani et al. [11] use indices created specifically for the Weighted Ridge Regression used by LIME, therefore the obtained indices cannot be generated for other explainability methods, meaning that this comparison cannot happen.

5.2 Qualitative Evaluation

Since most explanations are aimed at finally being used by humans, most studies on Explainable AI could be evaluated via human evaluation, more specifically domain expert assessments [1]. Several studies use this method of evaluation for Explainable AI techniques [34] [35], however qualitative explanations have limited potential in the field of time-series, due to the unintuitive nature of this data [1]. Due to this reason, both qualitative and quantitative evaluations were held to assess the results of this study.

For the qualitative evaluation, a group of eight people who work with the data used in this research on a daily basis were chosen. Their roles and years of experience in the field were recorded in order to be able to determine their level of expertise. This evaluation consisted of these participants answering some questions through an online form and a demonstration of predictions for two customers using the User Interface Prototype created.

5.2.1 Summary of Evaluation Questions

Before starting the evaluation process, participants were asked to give their consent for participating in this study. The following questions required participants to state their role at work, how they use data related to payments and how many years of experience they have working in the field of payments.

The final two questions before the demonstration asked how much participants trust a predictive model and whether anything would help them trust the predictive model more. At this point, they were not yet introduced to any explainability methods, therefore the main aim of this question was to gather domain experts' ideas of model explainability.

Next, examples for two customers were demonstrated through the user interface. First, participants were shown aggregated data, yesterday's data and a time-series plot of the features related to a customer. After analysing this data for some time, they were asked what they would think the total amount of deposits for the following day would be and what the thought process behind the value was. Using this question, one could analyze whether the explanations

produced used the same intuition as any of the participants.
Next, the prediction values, model confidence explanations, LIME explanations and SHAP explanations were presented. Participants were asked to score how much they trusted each explanation on a scale from one to five (one giving the least trust and five giving the most trust).
Finally, the LIME SHAP explanation confidence indices were shown along with the boxplots for the different features of the explanations. Once again, participants were asked how much these explanations increased their trust in the model on a scale from one to five.
Following the demonstration of two example customers, participants had to choose which explanation technique they found most clear from LIME, SHAP and Model Confidence. They were also asked whether they were willing to trade off a model's accuracy for more transparency. In the final question, participants were asked whether they had any proposals on what could make the predictions and explanations more clear.

5.2.2 Results

Introductory Questions

All eight participants gave their consent to participating in this study. The participants had different roles and experiences. Roles included various types of data analysts, people working on payment operations as well as the Head of Payments. Their daily jobs involve using payments data for fraud mitigation, reporting, analysing KPIs and task management amongst other things. Experience of the participants ranged from one to ten years, with the average number of experience years amounting to around 5 years.
Figure 5.1 shows the participant answers to the question on how much they trust a predictive model. The average trust is 3.38/5, which is a relatively high value. One can also note that none of the participants would trust the model completely.
The majority of users stated that knowing that a model has a high performance accuracy would help them trust its forecasts more. Another interesting idea

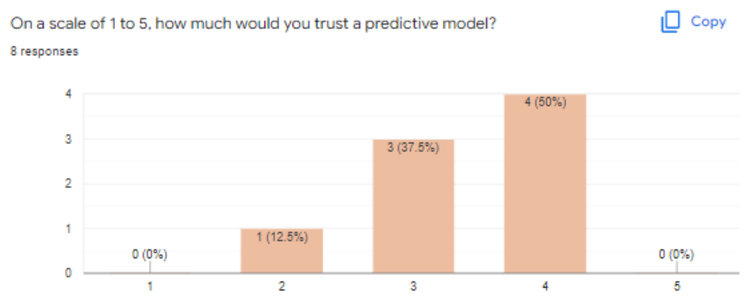
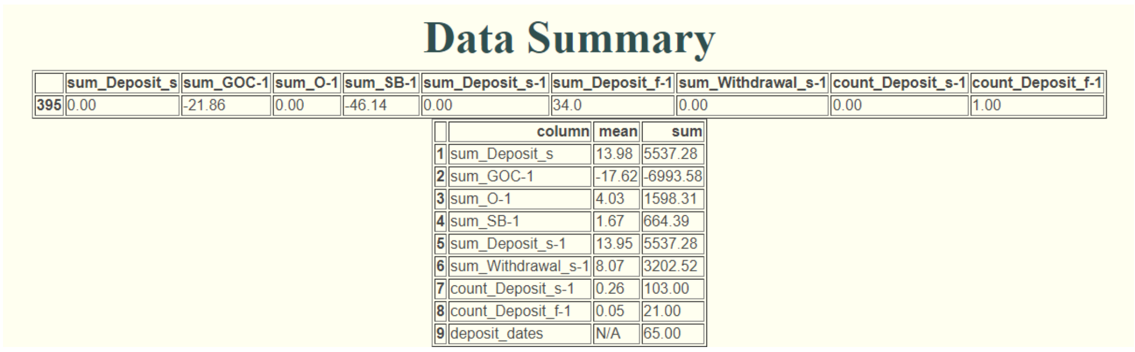


Figure 5.1: Participant answers to the question on how much they would trust a predictive model.

raised by one of the participants was that knowing a model is regularly updated to keep up with trends would help improve their trust in its predictions. Other answers included understanding the model and its intuition.

Demonstration Example 1

Figure 5.2 shows a summary of the data for the customer used in the first example. Based on this data and a line graph visualization of the features over time, participants had to give a value for the amount they thought the customer would deposit on the next day. The average of the values provided was 88. Reasons for the predictions were mostly based on the customer's depositing pattern. One feature which was mentioned several times as a reason for these predictions was the failed deposit amount for the previous day. Some participants noted that since the customer had a failed deposit on the previous day and no successful one, they would try depositing again.



Data Summary

	sum_Deposit_s	sum_GOC-1	sum_O-1	sum_SB-1	sum_Deposit_s-1	sum_Deposit_f-1	sum_Withdrawal_s-1	count_Deposit_s-1	count_Deposit_f-1
395	0.00	-21.86	0.00	-46.14	0.00	34.0	0.00	0.00	1.00

	column	mean	sum
1	sum_Deposit_s	13.98	5537.28
2	sum_GOC-1	-17.62	-6993.58
3	sum_O-1	4.03	1598.31
4	sum_SB-1	1.67	664.39
5	sum_Deposit_s-1	13.95	5537.28
6	sum_Withdrawal_s-1	8.07	3202.52
7	count_Deposit_s-1	0.26	103.00
8	count_Deposit_f-1	0.05	21.00
9	deposit_dates	N/A	65.00

Figure 5.2: Data summary shown for the customer in Demonstration Example 1.

The prediction made by the model was that of 223.92. This value was within the range of the values predicted by the participants, as their maximum prediction was 250. It is important to note that for this particular case, all prediction calls made by the model produced the same value.
Figure 5.3 shows the LIME explanation produced for the prediction of the first demo example. One can note that the failed and successful deposit features attributed to the prediction made. These explanations seem to be in line with the intuition of some of the participants.

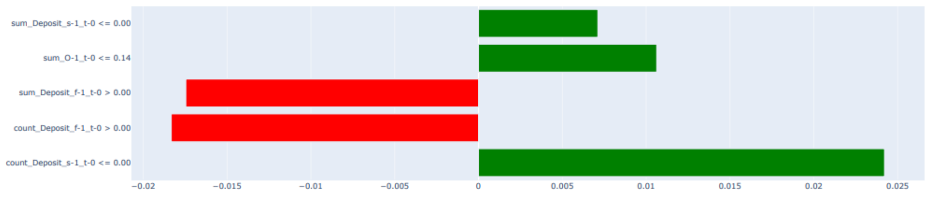


Figure 5.3: LIME explanation shown for the prediction for the customer in Demonstration Example 1.

Figure 5.4 shows the response of the participants for how much they trust the LIME explanation shown in Figure 5.3. The average trust value was 2.75. A higher trust score could not be observed for participants who stated that the failed deposit for the previous day was behind their prediction value. This might be since for them, the failed deposit had a positive affect on the prediction value, not a negative one as shown in the LIME explanation.

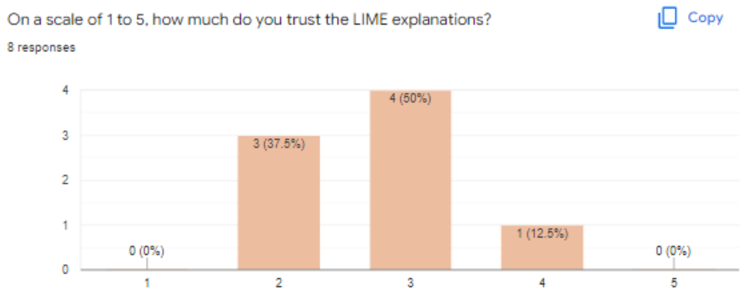


Figure 5.4: Participant responses on how much they trust the LIME explanations.

Next, the SHAP explanation shown in Figure 5.5 was presented. One can note that the fact the customer has no transactions related to *Other* products reduces has the lowest negative effect on the prediction whilst the fact that the customer lost on Games of Chance has the highest positive effect on the prediction. None of the participants had mentioned the *other* transactions feature as a reason for their prediction, however losses were mentioned, meaning that the customer losses on Games of Chance also affected their prediction values.

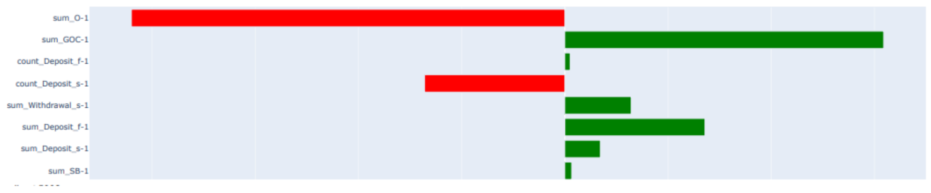


Figure 5.5: SHAP explanation shown for the prediction for the customer in Demonstration Example 1.

Figure 5.6 shows the participants' trust in the SHAP explanations, amounting to an average trust value of 3.5, almost a point higher than the average trust value for the LIME explanations. In this case, no pattern between the customers who stated that customer losses were part of the reason for their prediction and the SHAP trust scores was found.

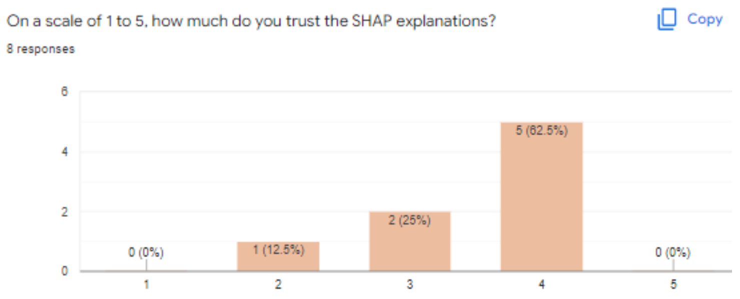


Figure 5.6: Participant responses on how much they trust the SHAP explanations.

Figure 5.7 shows how much participants trusted the model confidence explanation. This explanation scored an average trust value of 3.5, outscoring average trust for the LIME explanations and obtaining the same trust value as the SHAP explanations.

The confidence index for the LIME explanations was 18.41, whilst the confidence index for the SHAP explanations was 129.4. Since SHAP values have a lower mean than the LIME values, and the values for different features are further apart from each other, the box-plot might be misleading. One can see the

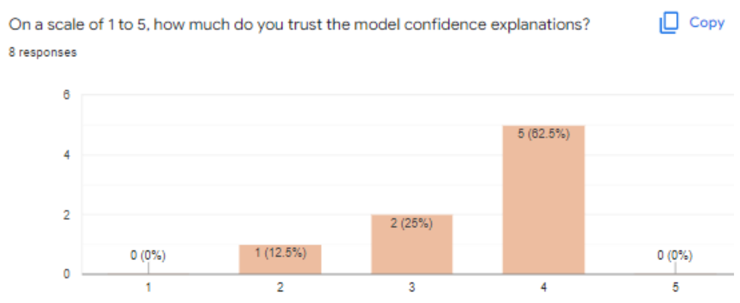


Figure 5.7: Participant responses on how much they trust the model confidence explanations.

boxplots for the LIME and SHAP features in Figure 5.8 and Figure 5.9 respectively. One would think that the SHAP explanations are more stable, however the index shows that this is in fact not the case.

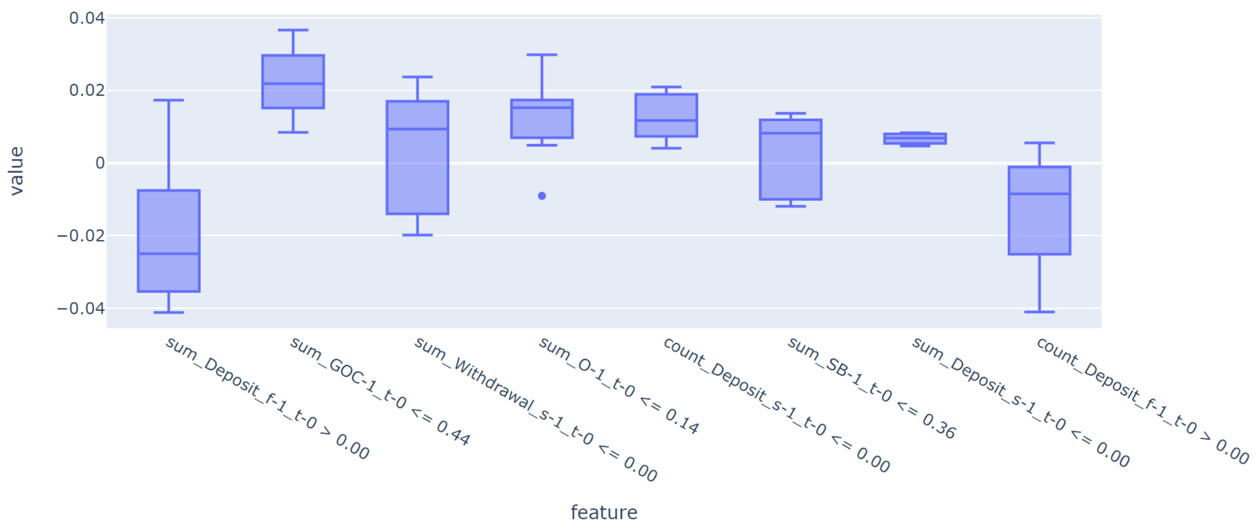


Figure 5.8: Box plot showing the distribution for the LIME explanations for the predictions for the customer in Demonstration Example 1.

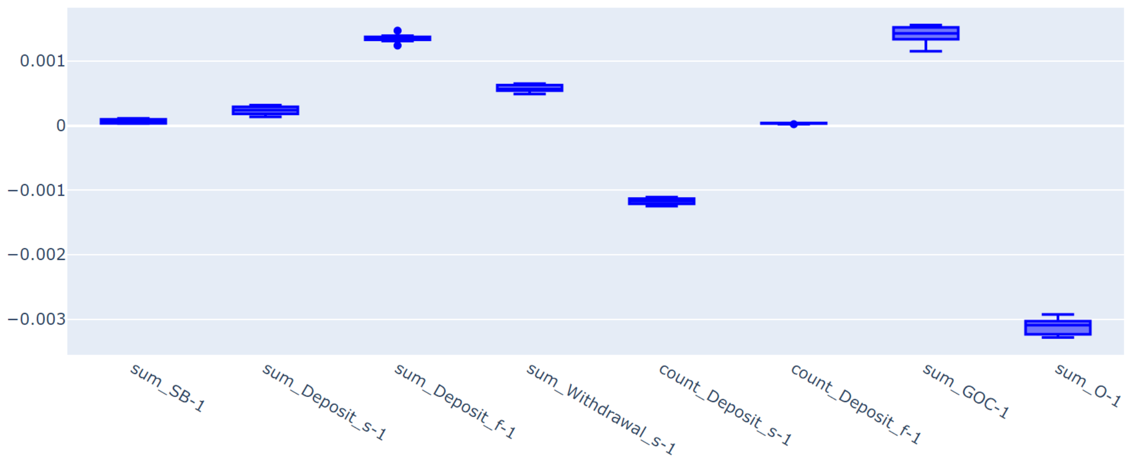
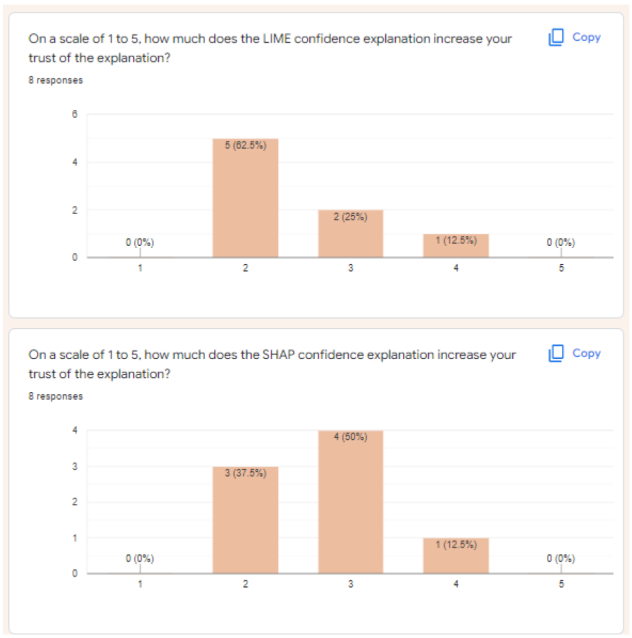


Figure 5.9: Box plot showing the distribution for the SHAP explanations for the predictions for the customer in Demonstration Example 1.

Figure 5.10 shows how much the explanation confidence increases the participants' trust in the model explanations. For LIME, the average trust score was 2.5 whilst for SHAP it was 2.75. From Figure 5.8, one can note that in the LIME explanations, the attribution value for the failed deposit features varies by quite a bit, which might have decreased the users' trust in the explanation. For SHAP, the attribution value for the count of failed deposits varied very slightly, which might be the reason that the participants leaned towards trusting the SHAP explanations more.


On a scale of 1 to 5, how much does the LIME confidence explanation increase your trust of the explanation?

8 responses

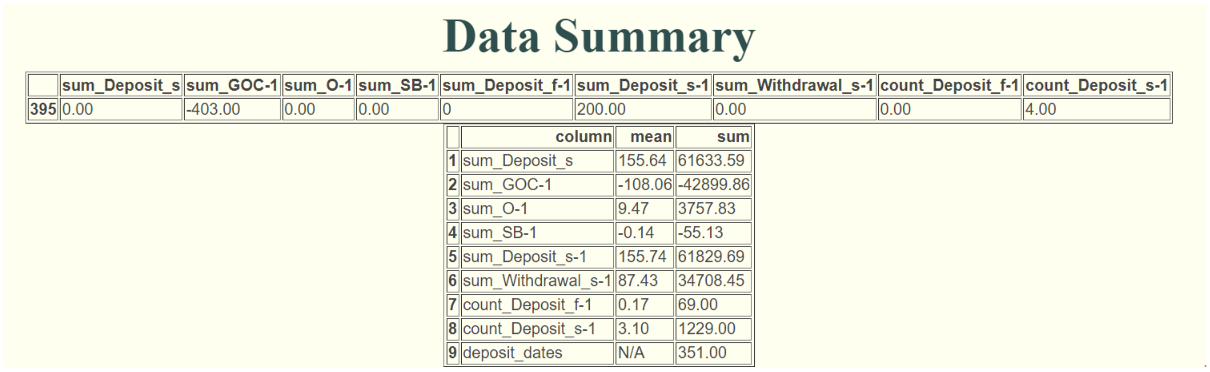
On a scale of 1 to 5, how much does the SHAP confidence explanation increase your trust of the explanation?

8 responses

Figure 5.10: Trust levels for the LIME confidence explanations (top) and SHAP confidence explanations (bottom).

Demonstration Example 2

Figure 5.11 shows a summary of the data for the customer used in the second example. The average value for human predictions for this customer was 134. Once again, these predictions were mostly based on the customer's depositing pattern. Some predictions also made mention of the customer's losses.

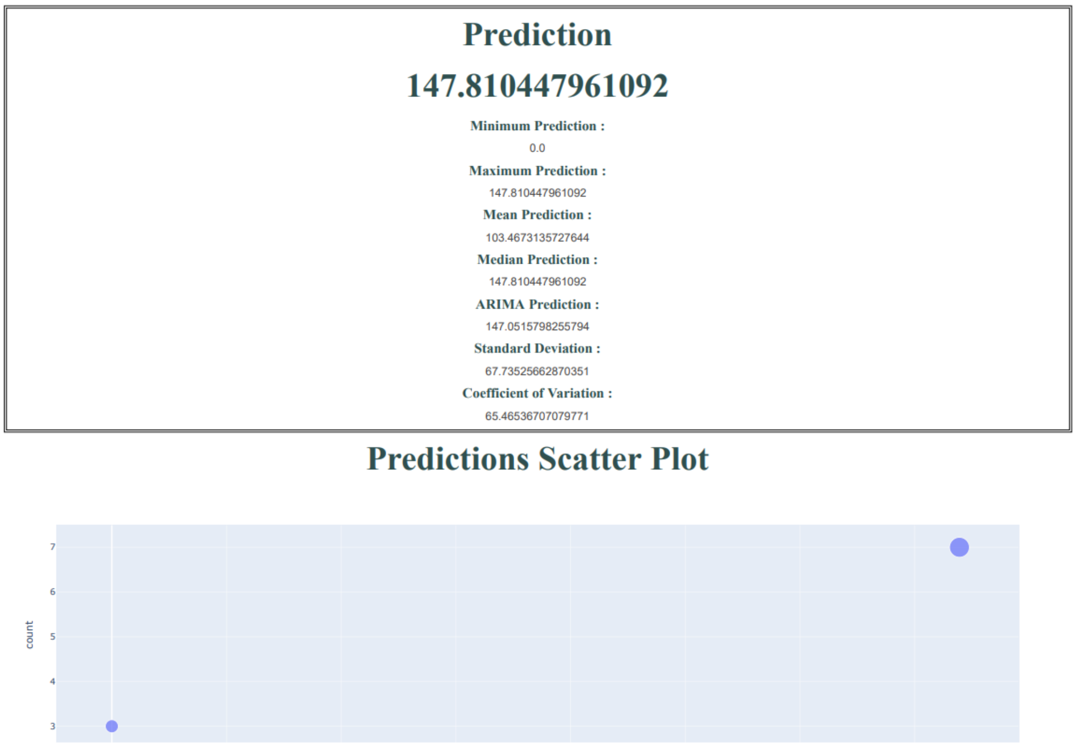

Data Summary

	sum_Deposit_s	sum_GOC-1	sum_O-1	sum_SB-1	sum_Deposit_f-1	sum_Deposit_s-1	sum_Withdrawal_s-1	count_Deposit_f-1	count_Deposit_s-1
395	0.00	-403.00	0.00	0.00	0	200.00	0.00	0.00	4.00

	column	mean	sum
1	sum_Deposit_s	155.64	61633.59
2	sum_GOC-1	-108.06	-42899.86
3	sum_O-1	9.47	3757.83
4	sum_SB-1	-0.14	-55.13
5	sum_Deposit_s-1	155.74	61829.69
6	sum_Withdrawal_s-1	87.43	34708.45
7	count_Deposit_f-1	0.17	69.00
8	count_Deposit_s-1	3.10	1229.00
9	deposit_dates	N/A	351.00

Figure 5.11: Data summary shown for the customer in Demonstration Example 2.

The prediction made by the model was that of 147.8. This value is very close to the average of the values that were predicted by the participants. This time, the predictions were not all the same, as they were in the previous example. Figure 5.12 shows the information on the different predictions as well as the scatter plot showing the values of these predictions. One can note that the model predicted a value of zero in three of the ten prediction calls.



Prediction

147.810447961092

Minimum Prediction :
0.0
Maximum Prediction :
147.810447961092
Mean Prediction :
103.4673135727644
Median Prediction :
147.810447961092
ARIMA Prediction :
147.0515798255794
Standard Deviation :
67.73525662870351
Coefficient of Variation :
65.46536707079771

Predictions Scatter Plot

Figure 5.12: Prediction output for Demonstration Example 2.

Figure 5.13 shows the LIME explanation produced for the prediction of the second demo example. It can be observed that the feature with the largest contribution is the sum of successful withdrawals for the previous day, which is zero. The losses from Games of Chance and the deposits data for the previous day are also amongst the top five contributing features.

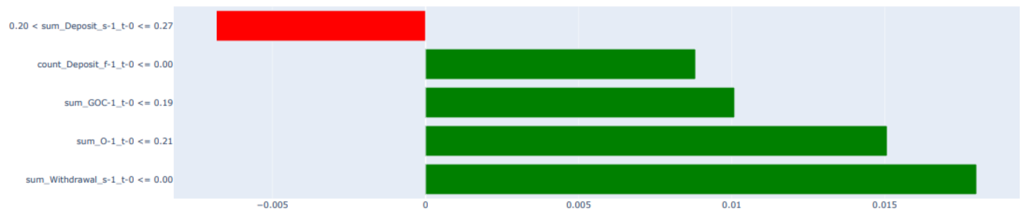


Figure 5.13: LIME explanation shown for the prediction for the customer in Demonstration Example 2.

Figure 5.14 shows the response of the participants for how much they trust the LIME explanation shown in Figure 5.13. The average trust value was 3.25. From verbal feedback collected during the evaluation session, the participants agreed that the previous day's deposits should lower the prediction whilst the losses from Games of Chance should increase it.

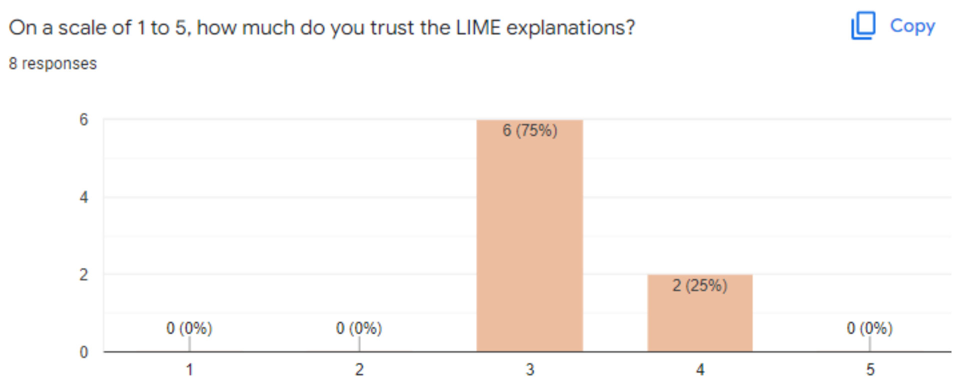


Figure 5.14: Participant responses on how much they trust the LIME explanations.

Next, the SHAP explanation shown in Figure 5.15 was presented. One can note that the feature with the highest SHAP value is the amount of Sportsbook transactions that the customer made on the previous day. However, this customer had not played Sportsbook within the timeframe of the dataset. In contrast to the LIME explanation, the SHAP explanations showed that the fact the customer didn't have any transactions classified as *Other*, actually attributed negatively towards the prediction.

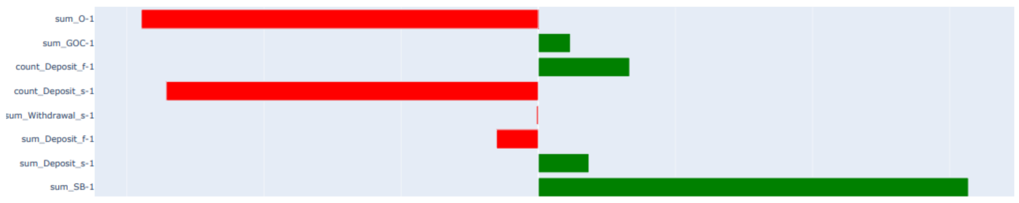


Figure 5.15: SHAP explanation shown for the prediction for the customer in Demonstration Example 2.

Figure 5.16 shows the participants' trust in the SHAP explanations, amounting to an average trust value of 3.25, equal to the mean of the LIME trust values.

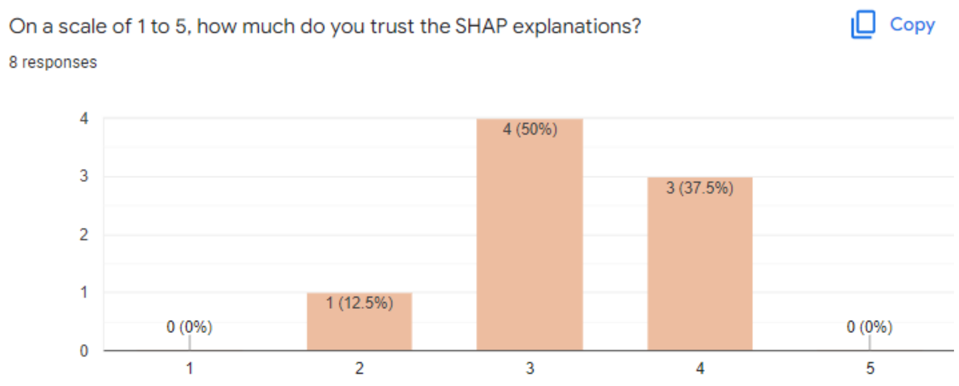


Figure 5.16: Participant responses on how much they trust the SHAP explanations.

Figure 5.17 shows how much participants trusted the model confidence explanation. This explanation scored an average trust value of 3.5, this time outscoring the trust values for both LIME and SHAP. Even though for this particular prediction, the stability of predictions was not as high as in the first example, the average trust remained the same.

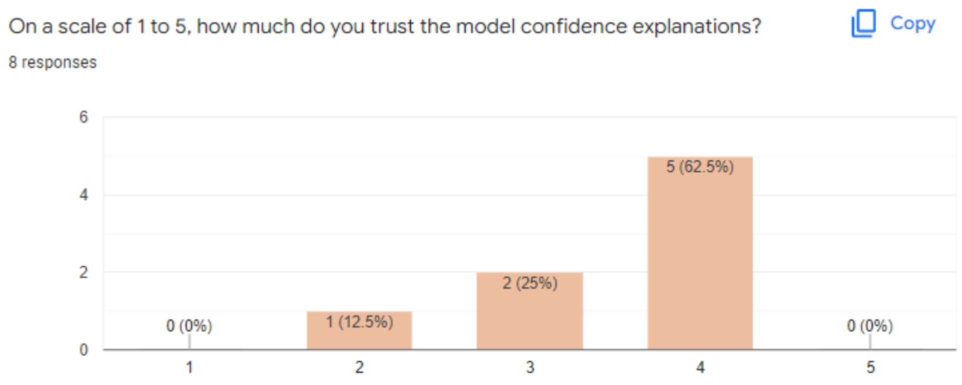


Figure 5.17: Participant responses on how much they trust the model confidence explanations.

The confidence index for the LIME explanations was 4.85, whilst the confidence index for the SHAP explanations was 9.75. The box-plots for the LIME and SHAP features are shown in in Figure 5.18 and Figure 5.19 respectively. The explanation values for this example are very stable.

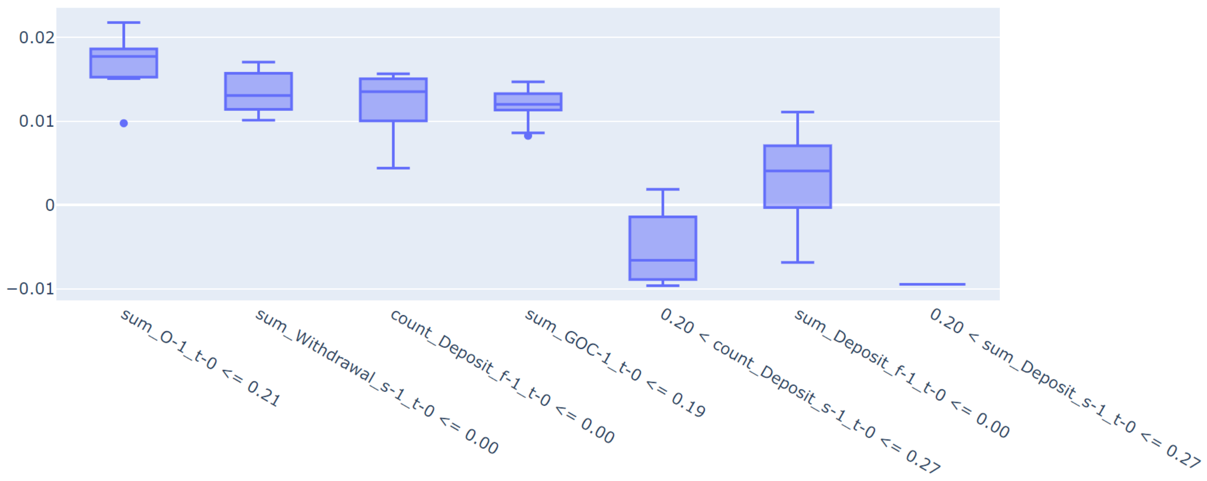


Figure 5.18: Box plot showing the distribution for the LIME explanations for the predictions for the customer in Demonstration Example 2.

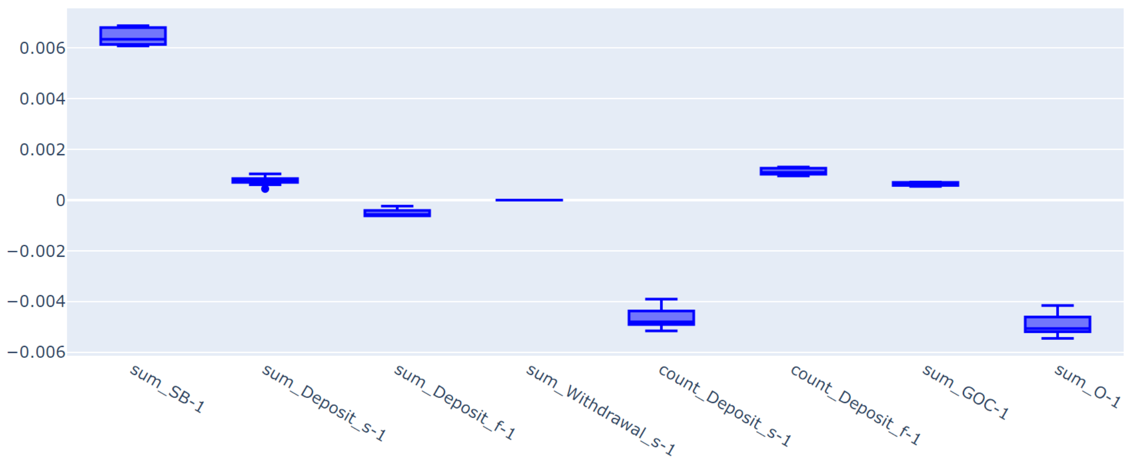


Figure 5.19: Box plot showing the distribution for the SHAP explanations for the predictions for the customer in Demonstration Example 2.

Figure 5.20 shows how much the explanation confidence increases the participants' trust in the model explanations. For LIME, the average trust score was 3.5 whilst for SHAP it was 3.25. In 5.18, one can note that the in the LIME explanations, the sum of deposits on the previous day's attribution value was very stable. This lead to the participants trusting these explanations since they matched their intuition. Figure 5.19 shows that the most stable SHAP values were the lower values.

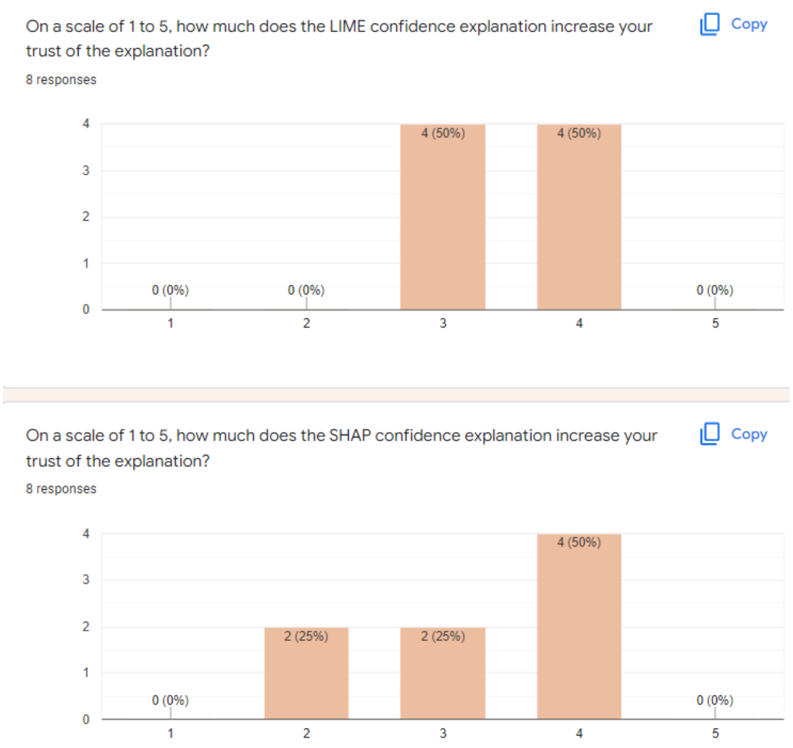


Figure 5.20: Trust levels for the LIME confidence explanations (top) and SHAP confidence explanations (bottom).

Concluding Questions

Figure 5.21 shows the participants' preferred explanation techniques, showing that the Confidence explanations were the most preferred, while no participants preferred the LIME explanations.

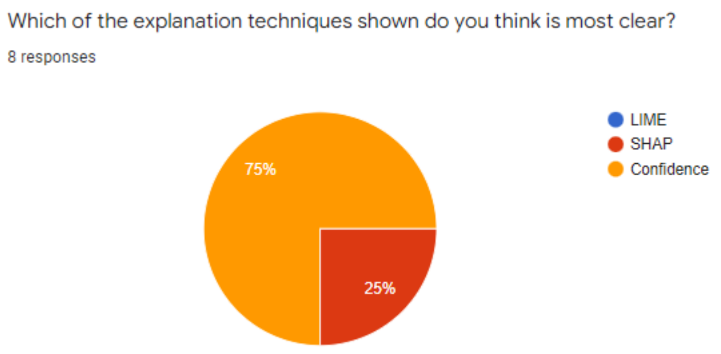


Figure 5.21: The participants' preferred explanation methods.

Most participants were not willing to trade accuracy for more transparency, with 5 answering that they would not make this trade-off, two answering that they are willing to trade accuracy for transparency and one participant answering that this would depend on how much accuracy were to be traded off.
In the final questions, participants had to give ideas on how the presented explanations could be made clearer. Answers to this question included :

- Using the full historical data of a customer.
- Giving a better description of the features before showing the explanation.
- Showing more visualization.

5.3 Conclusion

Overall, the model results could not be deemed satisfactory. The fact that statistical models using a univariate dataset managed to constantly outperform the deep learning model using a multivariate dataset shows that there is room for a lot of improvement to be made in the dataset and the model. These improvements will be discussed in the following chapter.

5.3.1 Summary of Results

The Index created in order to perform quantitative evaluation of the explanations proved to be successful. Although no correlation was found between this index and the prediction's accuracy, the index managed to quantify the explanation uncertainty well, especially for LIME explanations.
Finally, the qualitative explanation showed that domain experts are in fact interested in using predictive models to help in their everyday tasks. The level of trust they have in such models is already quite high, however there is still room to increase this level of trust.
Overall, the participants all preferred the confidence explanations. This is likely due to the fact that this explanation was the easiest one for participants to understand. As stated by one of the participants in the final question, adding clearer explanations for each feature might help users understand the LIME and SHAP explanations better.

Another thing to note was that SHAP was preferred to LIME. From verbal feedback during the evaluation session, the reason that SHAP was preferred was that it showed all features. This means that this preference was not due to the actual values, but related to the number of features output by each method. If one were to show the participants all the features in LIME the results might have turned out differently.

It was also clear that most experts were not willing to trade accuracy for transparency. This means that models having transparency must be as accurate as models without transparency or that a new transparency technique needs to be developed which would encourage the participants to change their mind about this decision.

An encouraging result is that the explainability confidence scores and box-plot visualizations helped users increase their trust in the explanations and in turn the predictions produced by the model. In Demo example 1, the LIME and SHAP confidence indices were higher than the indices for the explanations in Example 2. In example 1, this increased the participants' trust in the model an average of 2.5 and 2.75 for LIME and SHAP respectively. In example 2, higher trust values averaging 3.5 and 3.25 for LIME and SHAP respectively were recorded. This implied that the higher stability indices helped increase the participants' trust in the explanations.

Chapter 6 - Conclusions

Throughout this study, data related to customer transactions was analysed and a time-series dataset was created and then used by different models to forecast a customer's deposits for the following day. Explainable predictions were output from the LSTM model using Monte-Carlo Dropout, SHAP and LIME. An index was created to evaluate the stability of the generated SHAP and LIME explanations. Finally, the work done for this project was presented to domain experts through a web application in order to conduct an expert evaluation of the explainability.

6.1 Revisiting the Aim and Objectives

The aim of this project, specified in the first chapter, was to obtain accurate and explainable predictions and to evaluate these explanations. The first objective (O1) set to reach this aim was to perform EDA and create a time-series dataset. This was successfully accomplished through the extensive data analysis performed, which in turn helped to create a time-series dataset.
The second objective (O2) set was to configure a machine learning model which used the created dataset to forecast future values. An LSTM was configured in order to meet this objective, however this model's predictions were less accurate than that of the statistical ARIMA model.
The third objective (O3) was to explore whether Explainable AI could be used to interpret the model's predictions. This objective was successfully met through the Monte-Carlo, SHAP and LIME implementations on the model's predictions. Plots were also created in order to visualize these explanations.
The final objective (04) for this study was to evaluate and present the explainability results. The explainability results were evaluated quantitatively using

a stability index created in this research based on past studies [11] [31]. This index may be used on any explainability method for feature importance, therefore values were generated for both the LIME and SHAP explanations. SHAP explanations were found to be more stable than LIME explanations. A web application was then built to present all the results and an expert evaluation was held through which it was found that the explainability techniques used did in fact help increase the domain-experts' trust in the predictions, thus concluding the final objective.

6.2 Limitations

The main limitation of this study is the LSTM model. The created model does not produce satisfactory results as the ARIMA model manages to outperform it. This is probably due to the datasets not being large enough for the LSTM to gain an advantage over the statistical method. This is more concerning given that experts are not willing to trade off the accuracy of a model for transparency, meaning that the model must be improved in order for domain experts to be willing to use it. The following section will discuss some possible ways that the model forecasts could be improved.

6.3 Future Works and Improvements

Upon evaluating the results obtained through this study, some possible improvements were noted. These include :

1. *Using a customer's full historical data.* In this study, only one year of data was extracted, meaning that more data was available for most customers which was not utilized by the model. Such data could help improve the LSTM model's performance.
2. *Adding more features to the model.* This might also help the LSTM in making better predictions. One important feature that could be added is a customer's balance. A customer's balance would definitely effect how likely they would be to make a deposit. This was not used in this study since

it would make the data extraction process more complex, since an additional data source would be required. Data on promotions being offered and events taking place could also be introduced to the model.

3. *Using transfer learning.* In this study, a model was created for each customer. This means that the model was not aware of trends followed by other customers. Having such a visibility might improve the model's performance as the model would ultimately be much more knowledgeable. This could be done by training a model on the dataset for all customers and then using the weights produced by this pre-trained model as the starting weights for the individual customer's model.
4. *Outputting more visualizations related to SHAP and LIME.* The expert evaluation proved that the SHAP and LIME explanations were not very straightforward for people not intrinsic to machine learning and predictive models. With the help of these experts, one could look into creating explanations which are more user friendly and targeted directly to be used by such experts.

By taking into consideration these improvements, future work could include creating a larger, more refined time-series dataset, improving model performance by utilizing techniques such as transfer learning and creating more user-friendly explanation visualizations, with the target users being domain experts.

6.4 Concluding Remarks

In this study a deep-learning model as well as statistical models were applied to a created time-series and explanations on predictions were produced. Monte-Carlo dropout was used to add a layer of transparency to the model results as well as to utilize multiple predictions along with the ARIMA prediction in order to improve the model's performance. Multiple LIME and SHAP explanations were also produced and an index was developed to measure the stability of these explanations, thus helping a user identify how much an explanation should be trusted. All these results were output in a web application which was used to perform an expert evaluation of the explainability of the model.

This study opens up more opportunities for research on methods where explainability techniques may be evaluated as well as improved in such a way that they are more understandable by the end-users. It also established that domain experts do trust predictive models and are open to learning new ways that can help them better understand model predictions. The index developed in this study was received very well by these experts as it helped them trust the explanations and in turn, the predictions produced by the model even more.

References

[1] T. Rojat, R. Puget, D. Filliat, J. Del Ser, R. Gelin, and N. Díaz-Rodríguez, "Explainable artificial intelligence (xai) on timeseries data: A survey," *arXiv preprint arXiv:2104.00950*, 2021.

[2] A. B. Arrieta, N. Díaz-Rodríguez, J. Del Ser, A. Bennetot, S. Tabik, A. Barbado, S. García, S. Gil-López, D. Molina, R. Benjamins *et al.*, "Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai," *Information fusion*, vol. 58, pp. 82–115, 2020.

[3] C. Molnar, *Interpretable machine learning*. Lulu. com, 2020.

[4] A. Nielsen, *Practical time series analysis: prediction with statistics and machine learning*. " O'Reilly Media, Inc.", 2019.

[5] J. Hu and W. Zheng, "Transformation-gated lstm: Efficient capture of short-term mutation dependencies for multivariate time series prediction tasks," in *2019 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2019, pp. 1–8.

[6] S. Makridakis, E. Spiliotis, and V. Assimakopoulos, "Statistical and machine learning forecasting methods: Concerns and ways forward," *PloS one*, vol. 13, no. 3, p. e0194889, 2018.

[7] A. R. S. Parmezan, V. M. Souza, and G. E. Batista, "Evaluation of statistical and machine learning models for time series prediction: Identifying the state-of-the-art and the best conditions for the use of each model," *Information sciences*, vol. 484, pp. 302–337, 2019.

[8] P. T. Yamak, L. Yujian, and P. K. Gadosey, "A comparison between arima, lstm, and gru for time series forecasting," in *Proceedings of the 2019 2nd Inter-*

national Conference on Algorithms, Computing and Artificial Intelligence, 2019, pp. 49–55.

[9] K. Fauvel, T. Lin, V. Masson, É. Fromont, and A. Termier, "Xcm: An explainable convolutional neural network for multivariate time series classification," *arXiv preprint arXiv:2009.04796*, 2020.

[10] J. Souza and C. K. Leung, "Explainable artificial intelligence for predictive analytics on customer turnover: A user-friendly interface for non-expert users," in *Explainable AI Within the Digital Transformation and Cyber Physical Systems*. Springer, 2021, pp. 47–67.

[11] G. Visani, E. Bagli, F. Chesani, A. Poluzzi, and D. Capuzzo, "Statistical stability indices for lime: Obtaining reliable explanations for machine learning models," *Journal of the Operational Research Society*, vol. 73, no. 1, pp. 91–101, 2022.

[12] D. M. Nelson, A. C. Pereira, and R. A. De Oliveira, "Stock market's price movement prediction with lstm neural networks," in *2017 International joint conference on neural networks (IJCNN)*. IEEE, 2017, pp. 1419–1426.

[13] R. Goebel, A. Chander, K. Holzinger, F. Lecue, Z. Akata, S. Stumpf, P. Kieseberg, and A. Holzinger, "Explainable ai: the new 42?" in *International cross-domain conference for machine learning and knowledge extraction*. Springer, 2018, pp. 295–303.

[14] K. P. Kalyanathaya *et al.*, "A literature review and research agenda on explainable artificial intelligence (xai)," 2022.

[15] 2018 reform of eu data protection rules. European Commission. [Online]. Available: https://ec.europa.eu/commission/sites/beta-political/files/data-protection-factsheet-changes_en.pdf

[16] F. Xu, H. Uszkoreit, Y. Du, W. Fan, D. Zhao, and J. Zhu, *Explainable AI: A Brief Survey on History, Research Areas, Approaches and Challenges*, 09 2019, pp. 563–574.

[17] M. T. Ribeiro, S. Singh, and C. Guestrin, “" why should i trust you?" explaining the predictions of any classifier,” in *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, 2016, pp. 1135–1144.

[18] S. M. Lundberg and S.-I. Lee, “A unified approach to interpreting model predictions,” *Advances in neural information processing systems*, vol. 30, 2017.

[19] A. R. S. Parmezan, V. M. Souza, and G. E. Batista, “Evaluation of statistical and machine learning models for time series prediction: Identifying the state-of-the-art and the best conditions for the use of each model,” *Information sciences*, vol. 484, pp. 302–337, 2019.

[20] Q. Ma, “Comparison of arima, ann and lstm for stock price prediction,” in *E3S Web of Conferences*, vol. 218. EDP Sciences, 2020.

[21] G. Bontempi, S. B. Taieb, and Y.-A. Le Borgne, “Machine learning strategies for time series forecasting,” in *European business intelligence summer school*. Springer, 2012, pp. 62–77.

[22] N. K. Ahmed, A. F. Atiya, N. E. Gayar, and H. El-Shishiny, “An empirical comparison of machine learning models for time series forecasting,” *Econometric reviews*, vol. 29, no. 5-6, pp. 594–621, 2010.

[23] G. O. Roberts and J. S. Rosenthal, “General state space markov chains and mcmc algorithms,” *Probability surveys*, vol. 1, pp. 20–71, 2004.

[24] S. Siami-Namini, N. Tavakoli, and A. S. Namin, “A comparison of arima and lstm in forecasting time series,” in *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*. IEEE, 2018, pp. 1394–1401.

[25] V. Cerqueira, L. Torgo, and C. Soares, “Machine learning vs statistical methods for time series forecasting: Size matters,” *arXiv preprint arXiv:1909.13316*, 2019.

[26] N. K. Ahmed, A. F. Atiya, N. E. Gayar, and H. El-Shishiny, “An empirical comparison of machine learning models for time series forecasting,” *Econometric reviews*, vol. 29, no. 5-6, pp. 594–621, 2010.

[27] S. Siami-Namini, N. Tavakoli, and A. S. Namin, “A comparison of arima and lstm in forecasting time series,” in *2018 17th IEEE international conference on machine learning and applications (ICMLA)*. IEEE, 2018, pp. 1394–1401.

[28] D. Kiefer, F. Grimm, M. Bauer, D. Van *et al.*, “Demand forecasting intermittent and lumpy time series: Comparing statistical, machine learning and deep learning methods,” in *Proceedings of the 54th Hawaii International Conference on System Sciences*, 2021, p. 1425.

[29] G. P. Zhang, “Time series forecasting using a hybrid arima and neural network model,” *Neurocomputing*, vol. 50, pp. 159–175, 2003.

[30] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, “Grad-cam: Visual explanations from deep networks via gradient-based localization,” in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 618–626.

[31] X. Man and E. P. Chan, “The best way to select features? comparing mda, lime, and shap,” *The Journal of Financial Data Science*, vol. 3, no. 1, pp. 127–139, 2021.

[32] C. Hsiao, *Analysis of panel data*. Cambridge university press, 2022.

[33] T.-T. Nguyen and S. Yoon, “A novel approach to short-term stock price movement prediction using transfer learning,” *Applied Sciences*, vol. 9, no. 22, p. 4745, 2019.

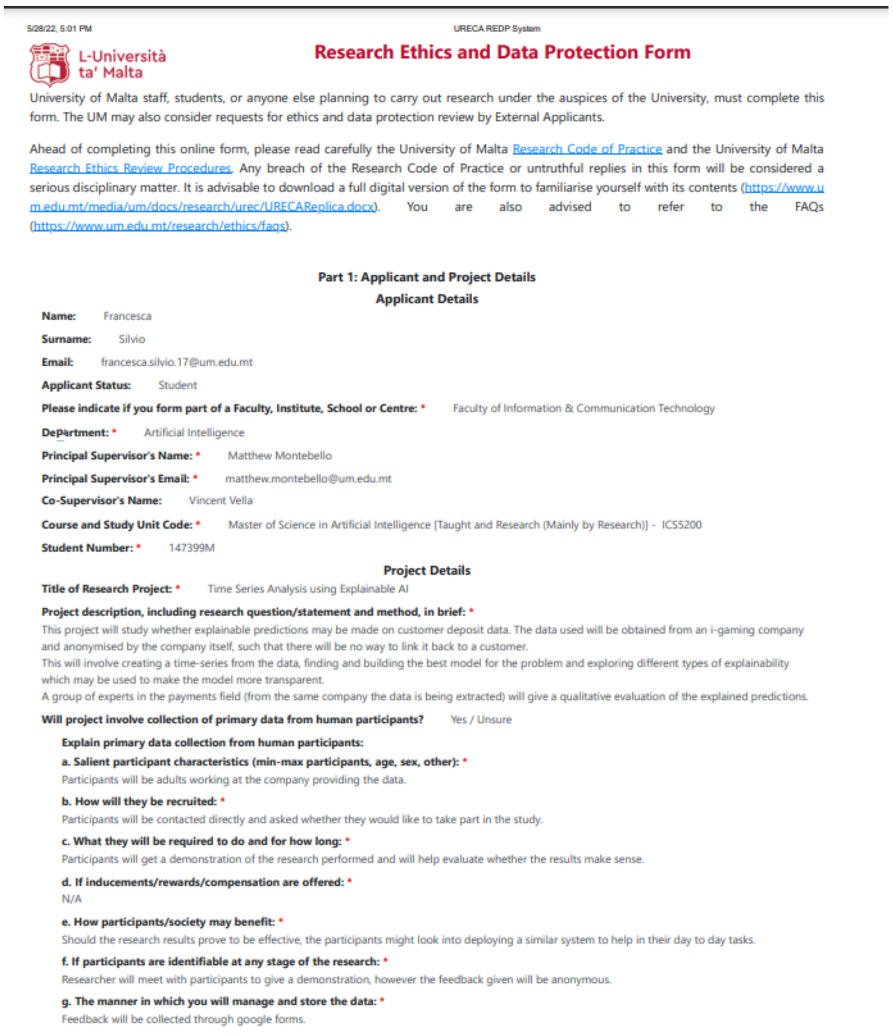
[34] H. Ismail Fawaz, G. Forestier, J. Weber, L. Idoumghar, and P.-A. Muller, “Accurate and interpretable evaluation of surgical skills from kinematic data using fully convolutional neural networks,” *International journal of computer assisted radiology and surgery*, vol. 14, no. 9, pp. 1611–1617, 2019.

[35] S. M. Muddamsetty, M. N. Jahromi, and T. B. Moeslund, “Expert level evaluations for explainable ai (xai) methods in the medical domain,” in *International Conference on Pattern Recognition*. Springer, 2021, pp. 35–46.

Chapter 7 - Appendix

7.1 Research Approvals

7.1.1 UREC Form



5/28/22, 5:01 PM URECA REDP System

L-Università ta' Malta

Research Ethics and Data Protection Form

University of Malta staff, students, or anyone else planning to carry out research under the auspices of the University, must complete this form. The UM may also consider requests for ethics and data protection review by External Applicants.

Ahead of completing this online form, please read carefully the University of Malta Research Code of Practice and the University of Malta Research Ethics Review Procedures. Any breach of the Research Code of Practice or untruthful replies in this form will be considered a serious disciplinary matter. It is advisable to download a full digital version of the form to familiarise yourself with its contents (https://www.um.edu.mt/media/um/docs/research/urec/URECAReplica.docx). You are also advised to refer to the FAQs (https://www.um.edu.mt/research/ethics/faqs).

Part 1: Applicant and Project Details

Applicant Details

Name: Francesca

Surname: Silvio

Email: francesca.silvio.17@um.edu.mt

Applicant Status: Student

Please indicate if you form part of a Faculty, Institute, School or Centre: * Faculty of Information & Communication Technology

Department: * Artificial Intelligence

Principal Supervisor's Name: * Matthew Montebello

Principal Supervisor's Email: * matthew.montebello@um.edu.mt

Co-Supervisor's Name: Vincent Vella

Course and Study Unit Code: * Master of Science in Artificial Intelligence [Taught and Research (Mainly by Research)] - ICS5200

Student Number: * 147399M

Project Details

Title of Research Project: * Time Series Analysis using Explainable AI

Project description, including research question/statement and method, in brief: *

This project will study whether explainable predictions may be made on customer deposit data. The data used will be obtained from an i-gaming company and anonymised by the company itself, such that there will be no way to link it back to a customer.
This will involve creating a time-series from the data, finding and building the best model for the problem and exploring different types of explainability which may be used to make the model more transparent.
A group of experts in the payments field (from the same company the data is being extracted) will give a qualitative evaluation of the explained predictions.

Will project involve collection of primary data from human participants? Yes / Unsure

Explain primary data collection from human participants:

a. Salient participant characteristics (min-max participants, age, sex, other): *
Participants will be adults working at the company providing the data.

b. How will they be recruited: *
Participants will be contacted directly and asked whether they would like to take part in the study.

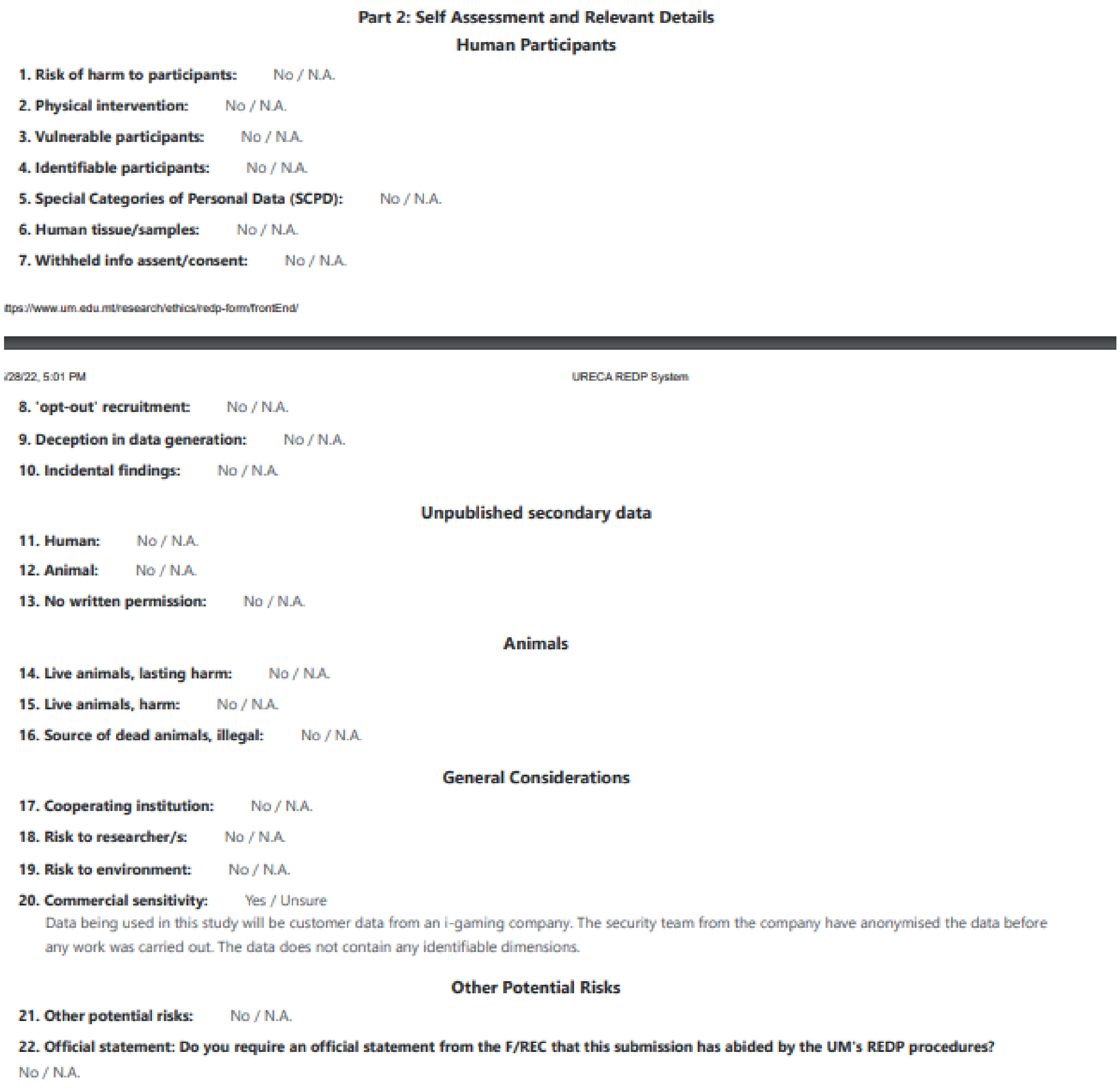
c. What they will be required to do and for how long: *
Participants will get a demonstration of the research performed and will help evaluate whether the results make sense.

d. If inducements/rewards/compensation are offered: *
N/A

e. How participants/society may benefit: *
Should the research results prove to be effective, the participants might look into deploying a similar system to help in their day to day tasks.

f. If participants are identifiable at any stage of the research: *
Researcher will meet with participants to give a demonstration, however the feedback given will be anonymous.

g. The manner in which you will manage and store the data: *
Feedback will be collected through google forms.



Part 2: Self Assessment and Relevant Details

Human Participants

1. Risk of harm to participants: No / N.A.

2. Physical intervention: No / N.A.

3. Vulnerable participants: No / N.A.

4. Identifiable participants: No / N.A.

5. Special Categories of Personal Data (SCPD): No / N.A.

6. Human tissue/samples: No / N.A.

7. Withheld info assent/consent: No / N.A.

ttps://www.um.edu.mt/research/ethics/redp-form/frontEnd/

/28/22, 5:01 PM URECA REDP System

8. 'opt-out' recruitment: No / N.A.

9. Deception in data generation: No / N.A.

10. Incidental findings: No / N.A.

Unpublished secondary data

11. Human: No / N.A.

12. Animal: No / N.A.

13. No written permission: No / N.A.

Animals

14. Live animals, lasting harm: No / N.A.

15. Live animals, harm: No / N.A.

16. Source of dead animals, illegal: No / N.A.

General Considerations

17. Cooperating institution: No / N.A.

18. Risk to researcher/s: No / N.A.

19. Risk to environment: No / N.A.

20. Commercial sensitivity: Yes / Unsure

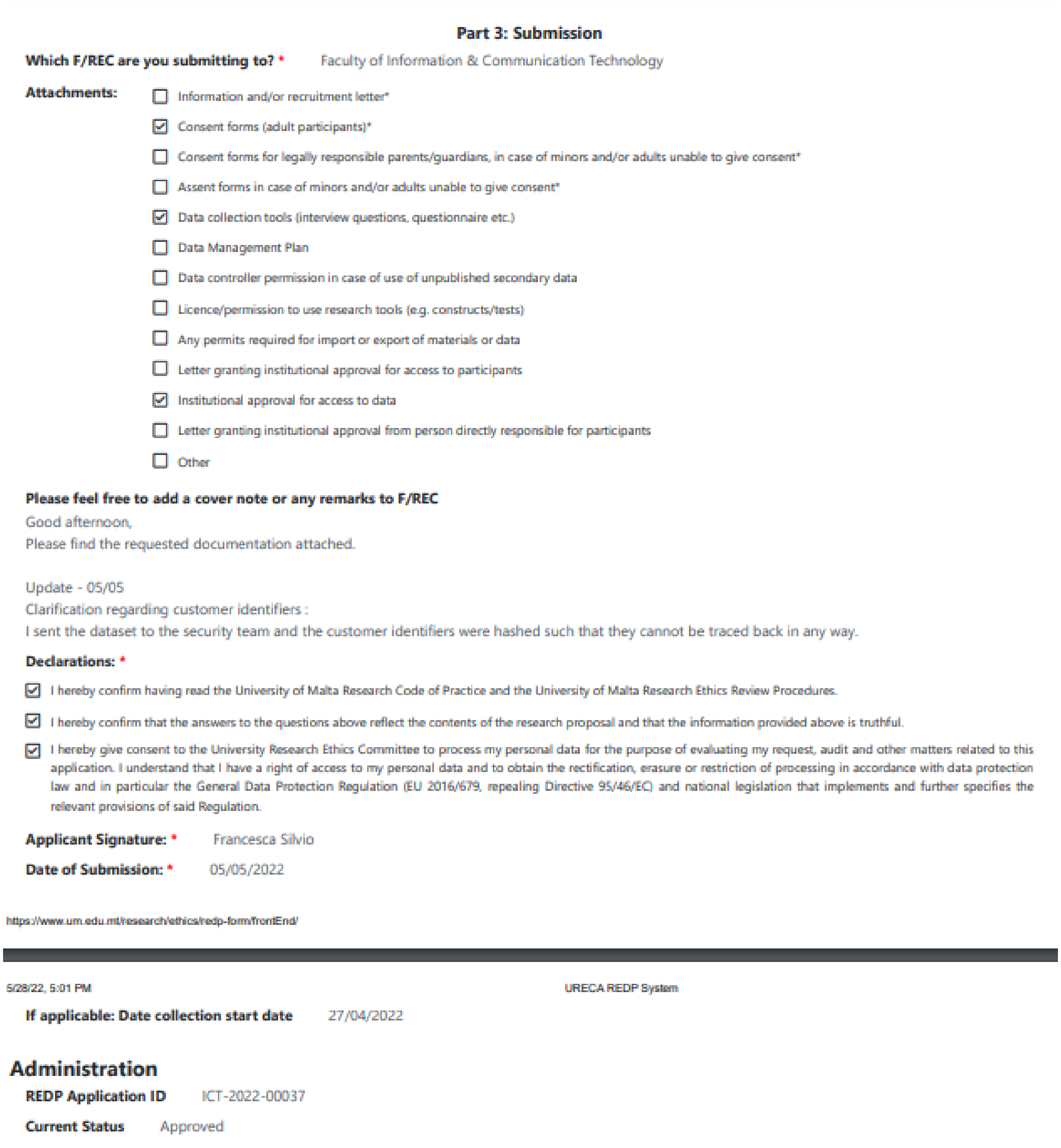
Data being used in this study will be customer data from an i-gaming company. The security team from the company have anonymised the data before any work was carried out. The data does not contain any identifiable dimensions.

Other Potential Risks

21. Other potential risks: No / N.A.

22. Official statement: Do you require an official statement from the F/REC that this submission has abided by the UM's REDP procedures?

No / N.A.



Part 3: Submission

Which F/REC are you submitting to? * Faculty of Information & Communication Technology

Attachments:

- ☐ Information and/or recruitment letter*
- ☑ Consent forms (adult participants)*
- ☐ Consent forms for legally responsible parents/guardians, in case of minors and/or adults unable to give consent*
- ☐ Assent forms in case of minors and/or adults unable to give consent*
- ☑ Data collection tools (interview questions, questionnaire etc.)
- ☐ Data Management Plan
- ☐ Data controller permission in case of use of unpublished secondary data
- ☐ Licence/permission to use research tools (e.g. constructs/tests)
- ☐ Any permits required for import or export of materials or data
- ☐ Letter granting institutional approval for access to participants
- ☑ Institutional approval for access to data
- ☐ Letter granting institutional approval from person directly responsible for participants
- ☐ Other

Please feel free to add a cover note or any remarks to F/REC

Good afternoon,
Please find the requested documentation attached.

Update - 05/05
Clarification regarding customer identifiers :
I sent the dataset to the security team and the customer identifiers were hashed such that they cannot be traced back in any way.

Declarations: *

- ☑ I hereby confirm having read the University of Malta Research Code of Practice and the University of Malta Research Ethics Review Procedures.
- ☑ I hereby confirm that the answers to the questions above reflect the contents of the research proposal and that the information provided above is truthful.
- ☑ I hereby give consent to the University Research Ethics Committee to process my personal data for the purpose of evaluating my request, audit and other matters related to this application. I understand that I have a right of access to my personal data and to obtain the rectification, erasure or restriction of processing in accordance with data protection law and in particular the General Data Protection Regulation (EU 2016/679, repealing Directive 95/46/EC) and national legislation that implements and further specifies the relevant provisions of said Regulation.

Applicant Signature: * Francesca Silvio

Date of Submission: * 05/05/2022

https://www.um.edu.mt/research/ethics/redp-form/frontEnd/

5/28/22, 5:01 PM URECA REDP System

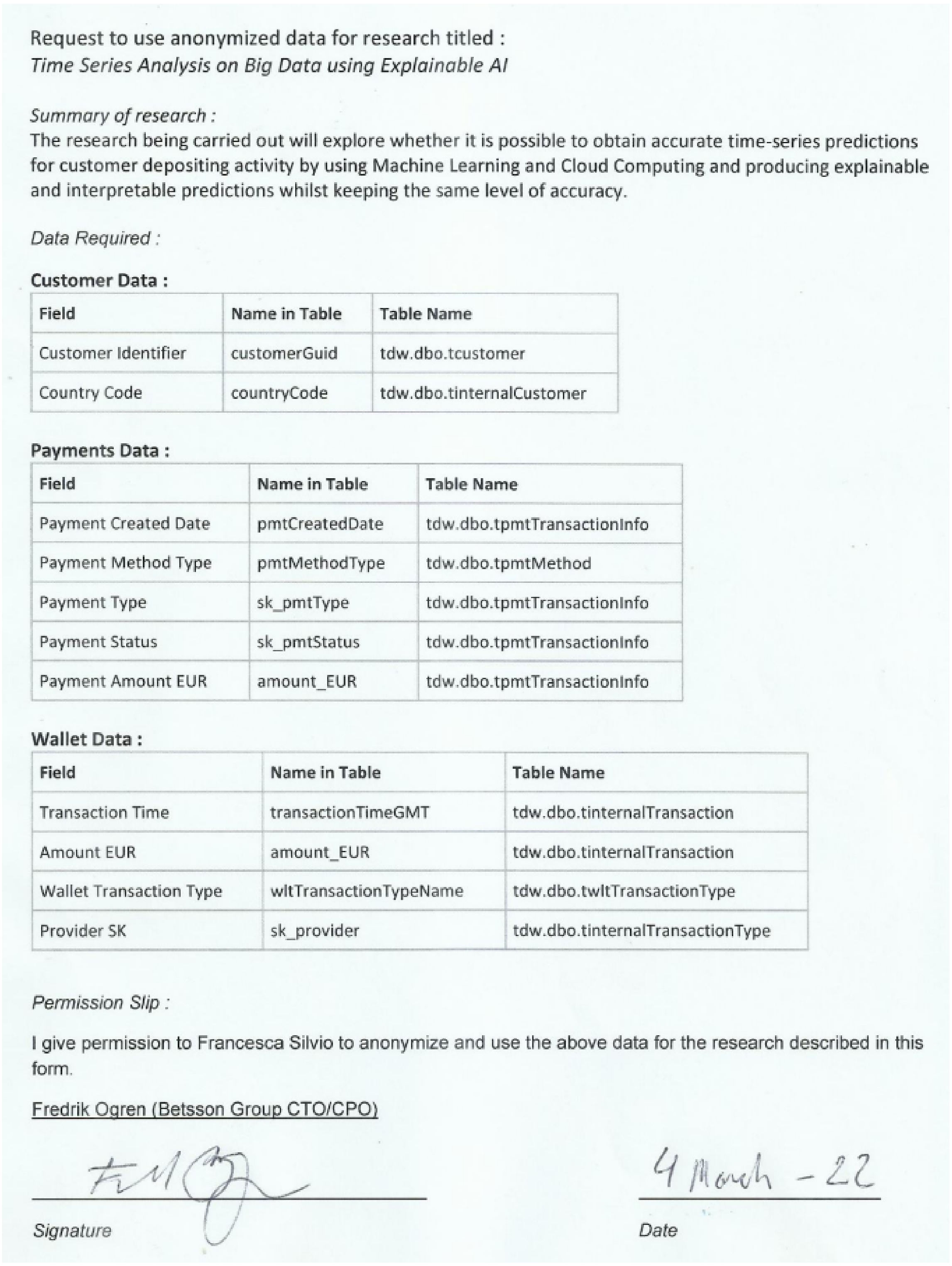
If applicable: Date collection start date 27/04/2022

Administration

REDP Application ID ICT-2022-00037

Current Status Approved

7.1.2 Data Permission Form



Request to use anonymized data for research titled :
Time Series Analysis on Big Data using Explainable AI

Summary of research :
The research being carried out will explore whether it is possible to obtain accurate time-series predictions for customer depositing activity by using Machine Learning and Cloud Computing and producing explainable and interpretable predictions whilst keeping the same level of accuracy.

Data Required :

Customer Data :

Field	Name in Table	Table Name
Customer Identifier	customerGuid	tdw.dbo.tcustomer
Country Code	countryCode	tdw.dbo.tinternalCustomer

Payments Data :

Field	Name in Table	Table Name
Payment Created Date	pmtCreatedDate	tdw.dbo.tpmtTransactionInfo
Payment Method Type	pmtMethodType	tdw.dbo.tpmtMethod
Payment Type	sk_pmtType	tdw.dbo.tpmtTransactionInfo
Payment Status	sk_pmtStatus	tdw.dbo.tpmtTransactionInfo
Payment Amount EUR	amount_EUR	tdw.dbo.tpmtTransactionInfo

Wallet Data :

Field	Name in Table	Table Name
Transaction Time	transactionTimeGMT	tdw.dbo.tinternalTransaction
Amount EUR	amount_EUR	tdw.dbo.tinternalTransaction
Wallet Transaction Type	wltTransactionTypeName	tdw.dbo.twltTransactionType
Provider SK	sk_provider	tdw.dbo.tinternalTransactionType

Permission Slip :

I give permission to Francesca Silvio to anonymize and use the above data for the research described in this form.

Fredrik Ogren (Betsson Group CTO/CPO)

[illegible]
Signature

4 March -22
Date

7.2 Submission Folder

The submission folder may be accessed through the below Google Drive link :
`https://drive.google.com/file/d/1_ZtZ8w8zEmPmt4Z8SA9eM47VDlWz4LI2/view?usp=sharing`
This submission folder is split into 13 parts :

1. **SQL Extractioin Queries** - contains the SQL queries used in order to extract the data from the database.
2. **EDA** - Folder containing scripts used to analyze the extracted data and process it to create a time-series.
3. **Training Experiment 1** - code files and data related to the first training experiment.
4. **Training Experiment 2** - code files and data related to the second training experiment.
5. **Training Experiment 3** - code files and data related to the third training experiment.
6. **Training Experiment 4** - code files and data related to the fourth training experiment.
7. **Training Experiment 5** - code files and data related to the fifth training experiment.
8. **Final Training** - code files, data and results related to the final training sessions.
9. **Evaluation** - files related to the training evaluation and quantitative explanation evaluation.
10. **Prototype** - source files and readMe related to the web application prototype of the developed work.
11. **Survey** - survey results from domain expert evaluation.
12. **Research Approvals** - Contains documents related to the approval of this research.
13. **Data** - other data related to this study.

7.3 Exploratory Data Analysis

Overview

In this notebook, I am performing an Exploratory Data Analysis on a sample of the data which will be used in this study.

The sample used contains 1 week of data for 1 brand on the platform.

In this notebook, reference is made to wallet and payments data. On the gaming platform, a customer has a wallet which is affected by transactions such as bets, wins, deposits and withdrawals. For the purpose of this study, data on deposits and withdrawals will be referred to as payments data and any other transactions affecting the wallet (which are not related to payments) will be referred to as wallet data. This means that transactions which are going from the customer's wallet to the gaming platform are wallet transactions whilst transactions going to/from the customer's payment account to their platform wallet will be referred to as payment transactions.

The unprocessed data was extracted for payments and wallet data separately.

Below, one can see an extract of the *payments* data and the number of rows before processing.

1
```
import pandas as pd
import numpy as np

payments_unproc = pd.read_csv('../Data/pmt_jan1_masked.csv', header=None,
                  names=['customerGuid','sk_pmtTransactionInfo','countryCode','pmtCreatedDate','pmtS
print('Number of rows: ', len(payments_unproc) )
payments_unproc.head()
```

```
Number of rows:  1048576
C:\Users\frans\.conda\envs\python_3_7\lib\site-packages\IPython\core\interactiveshell.py:3457: DtypeWa
  exec(code_obj, self.user_global_ns, self.user_ns)
```

1

	customerGuid	sk_pmtTransactionInfo	countryCode	pmtCreatedDate	pmtStatusChangeDate
0	FFB5661D-####-####-####-A9C0492CF659	543037780.0	FI	02:31.6	03:19.5
1	FFB5661D-####-####-####-A9C0492CF659	543138775.0	FI	59:11.6	00:01.6
2	FFB5661D-####-####-####-A9C0492CF659	543143726.0	FI	17:14.0	18:07.5

	customerGuid	sk_pmtTransactionInfo	countryCode	pmtCreatedDate	pmtStatusChangeDate
3	FFB5661D-####-####-####-A9C0492CF659	543163997.0	FI	08:29.1	09:21.0
4	FFB5661D-####-####-####-A9C0492CF659	542507798.0	FI	45:21.5	46:13.9

Below, one can see an extract of the *wallet* data and the number of rows before processing.

2
```
df1 = pd.read_csv('../Data/wlt_jan1_masked.csv', header=None, names=['customerGuid', 'transactionTime
df2 = pd.read_csv('../Data/wlt_jan2_masked.csv', header=None, names=['customerGuid', 'transactionTime

df12 = pd.concat([df1,df2],axis=0)
df3 = pd.read_csv('../Data/wlt_jan3_masked.csv', header=None, names=['customerGuid', 'transactionTime
wlt_unproc = pd.concat([df12,df2],axis=0)

print('Number of rows: ', len(wlt_unproc))
wlt_unproc.head()
```

```
C:\Users\frans\.conda\envs\python_3_7\lib\site-packages\IPython\core\interactiveshell.py:3457: DtypeWa
  exec(code_obj, self.user_global_ns, self.user_ns)

Number of rows:  48918504
```

2

	customerGuid	transactionTimeGMT	countryCode	amount_EUR	sk_provider	wltTransa
NaN	customerGuid	transactionTimeGMT	countryCode	amount_EUR	sk_provider	wltTransa
0.0	9459FCFD-####-####-####-7DE2	2022-01-01 19:33:13.550	NO	-0.9977	19	Bet
1.0	FA3A481C-####-####-####-4A22	2022-01-01 20:20:28.003	NO	-0.9977	122	Bet
2.0	756AB4FD-####-####-####-2207	2022-01-01 20:36:12.730	PE	-0.1323	19	Bet
3.0	9C745CD3-####-####-####-FB0C	2022-01-02 17:11:16.047	PE	0.0176	19	Win

Remarks :

One can immediately note the large difference in the volume of transactions extracted from wallet and payments.\ The payments sample contains 318,900 rows whilst the wallet sample contains 48,918,501 rows.\ It is also important to note that not all the rows in the payments sample are related to both successful and unsuccessful transactions.\

Load data

Start by loading the processed data.

The dataset that is being analyzed contains a total of 31 columns and 473,498 rows. This data has been pre-processed separately.\ The dataset contains is split by customer identifier and a set of time segments - each 4 hours apart (given that transactional activity was performed by the customer in that timeframe). The dataset also contains the country of each customer.\

The quantitive data in the dataset contains customer transactional data split by product and customer payments data (deposits and withdrawals) split by success/failure and payment method type.

3
```
data = pd.read_csv("../Data/processed_data.csv", index_col=0)
data = data.reset_index()
print('Number of rows: ', len(data) )
data.head()
```

```
Number of rows:  473498
```

3

	customerGuid	countryCode	created_timesegments	sum_n_SB	sum_p_SB	sum_n_GOC
0	00002912-####-####-####-B070	PE	2022-01-02 16:00:00	4.4103	0.0	0.0
1	00002912-####-####-####-B070	PE	2022-01-02 20:00:00	8.8188	0.0	0.0
2	00002912-####-####-####-B070	PE	2022-01-05 20:00:00	2.2324	0.0	0.0
3	00002912-####-####-####-B070	PE	2022-01-06 16:00:00	11.1653	0.0	0.0
4	000069A8-####-####-####-48FB	PE	2022-01-02 20:00:00	1.3228	0.0	0.0

5 rows × 31 columns

Data Distribution Analysis

In this section of the notebook, visualizations will be output in order to show how the sample data is distributed.

Function to create subplots.

4
```
from plotly.subplots import make_subplots
import plotly.graph_objects as go
import plotly.offline as pyo

# Set notebook mode to work in offline
pyo.init_notebook_mode()

def get_subplots(df,rows, cols, date, columns):
  fig = make_subplots(rows=rows, cols=cols, subplot_titles=(columns))

  row_cnt=0
  col_cnt=1

  for c in columns:
    if row_cnt>=rows:
      col_cnt+=1
      row_cnt=1
    else :
      row_cnt+=1
    fig.append_trace(go.Bar(x=df[date], y=df[c], name=c,  marker=dict(color="Black"), opacity=1), row
  return fig
```

In order to get a more generic view of the data, the customer reference and country code will be dropped and the quantitive values will be grouped by the time segments and summed up.

5
```
print(data)
```

```
                        customerGuid countryCode created_timesegments  \
0       00002912-####-####-####-B070          PE  2022-01-02 16:00:00
1       00002912-####-####-####-B070          PE  2022-01-02 20:00:00
2       00002912-####-####-####-B070          PE  2022-01-05 20:00:00
3       00002912-####-####-####-B070          PE  2022-01-06 16:00:00
4       000069A8-####-####-####-48FB          PE  2022-01-02 20:00:00
...                              ...         ...                  ...
473493  FFF15A57-####-####-####-A655          CL  2022-01-04 20:00:00
473494  FFF2EBC4-####-####-####-9576          PL  2022-01-03 08:00:00
473495  FFF4438B-####-####-####-4419          PE  2022-01-05 00:00:00
473496  FFF78042-####-####-####-F6E8          PE  2022-01-05 12:00:00
473497  FFF8A403-####-####-####-CE69          PE  2022-01-02 00:00:00

        sum_n_SB  sum_p_SB  sum_n_GOC  sum_p_GOC  sum_p_O  sum_n_O  \
0         4.4103       0.0        0.0        0.0      0.0      0.0
1         8.8188       0.0        0.0        0.0      0.0      0.0
2         2.2324       0.0        0.0        0.0      0.0      0.0
3        11.1653       0.0        0.0        0.0      0.0      0.0
4         1.3228       0.0        0.0        0.0      0.0      0.0
...          ...       ...        ...        ...      ...      ...
473493    0.0000       0.0        0.0        0.0      0.0      0.0
473494    0.0000       0.0        0.0        0.0      0.0      0.0
473495    0.0000       0.0        0.0        0.0      0.0      0.0
473496    0.0000       0.0        0.0        0.0      0.0      0.0
473497    0.0000       0.0        0.0        0.0      0.0      0.0

        sum_Deposit_Wallet_s  ...  count_Deposit_CreditCard_s  \
0                     0.0000  ...                         0.0
1                     0.0000  ...                         0.0
2                     0.0000  ...                         0.0
3                     0.0000  ...                         0.0
4                     4.4094  ...                         0.0
...                      ...  ...                         ...
473493                0.0000  ...                         0.0
473494                0.0000  ...                         0.0
473495                0.0000  ...                         0.0
473496                4.4649  ...                         0.0
```

```
473497              8.8227  ...                        0.0

        count_Deposit_CreditCard_f  count_Deposit_Wallet_f  \
0                              0.0                     0.0
1                              0.0                     0.0
2                              0.0                     0.0
3                              0.0                     0.0
4                              0.0                     0.0
...                            ...                     ...
473493                         1.0                     1.0
473494                         0.0                     1.0
473495                         1.0                     2.0
473496                         0.0                     0.0
473497                         0.0                     0.0

        count_Withdrawal_BankPayout_f  count_Withdrawal_BankPayout_s  \
0                                 0.0                            0.0
1                                 0.0                            0.0
2                                 0.0                            0.0
3                                 0.0                            0.0
4                                 0.0                            0.0
...                               ...                            ...
473493                            0.0                            0.0
473494                            0.0                            0.0
473495                            0.0                            0.0
473496                            0.0                            0.0
473497                            0.0                            0.0

        count_Withdrawal_Wallet_s  count_Withdrawal_Wallet_f  \
0                             0.0                        0.0
1                             0.0                        0.0
2                             0.0                        0.0
3                             0.0                        0.0
4                             0.0                        0.0
...                           ...                        ...
473493                        0.0                        0.0
473494                        0.0                        0.0
473495                        0.0                        0.0
473496                        0.0                        0.0
473497                        0.0                        0.0

        count_Withdrawal_CreditCard_s  count_Withdrawal_CreditCard_f  \
0                                 0.0                            0.0
1                                 0.0                            0.0
2                                 0.0                            0.0
3                                 0.0                            0.0
4                                 0.0                            0.0
...                               ...                            ...
473493                            0.0                            0.0
473494                            0.0                            0.0
473495                            0.0                            0.0
473496                            0.0                            0.0
473497                            0.0                            0.0

        count_Deposit_Bank_f
0                        0.0
1                        0.0
2                        0.0
3                        0.0
4                        0.0
...                      ...
473493                   0.0
473494                   0.0
473495                   0.0
473496                   0.0
473497                   0.0

[473498 rows x 31 columns]
```

6
```
grouped_data = data.drop(['customerGuid', 'countryCode'], axis=1)
grouped_data = grouped_data.groupby(["created_timesegments"]).agg('sum')
grouped_data = grouped_data.reset_index()
```

7
```
grouped_data['deposits_s'] = grouped_data['sum_Deposit_CreditCard_s']+grouped_data['sum_Deposit_Walle
grouped_data['withdrawals_s'] =grouped_data['sum_Withdrawal_BankPayout_s']+ grouped_data['sum_Withdraw
grouped_data.head()
```

7

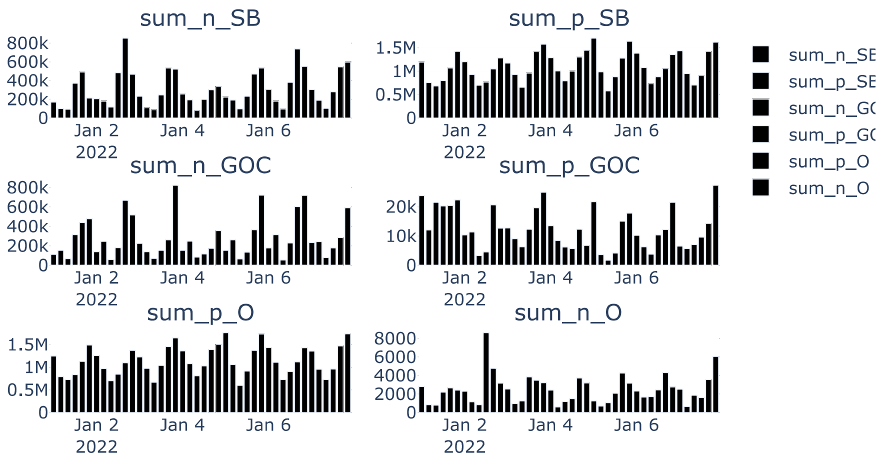
	created_timesegments	sum_n_SB	sum_p_SB	sum_n_GOC	sum_p_GOC	sum_p_O
0	2022-01-01 00:00:00	169475.8784	110640.7645	1.246075e+06	1.191828e+06	23825.2289
1	2022-01-01 04:00:00	99690.0067	154005.1852	7.917224e+05	7.549154e+05	12100.9162
2	2022-01-01 08:00:00	93875.2902	68082.0872	7.268918e+05	6.840198e+05	21467.8431
3	2022-01-01 12:00:00	370301.3039	315847.5170	8.356710e+05	7.983196e+05	20297.8505
4	2022-01-01 16:00:00	491990.9490	439738.0582	1.125939e+06	1.068906e+06	20405.2416

5 rows × 31 columns

Show Wallet Data

Showing the sums of the amounts for each coloumn related to wallet transactional data at each timestamp.

8
```
subplots = get_subplots(grouped_data,3,2,'created_timesegments', data.columns[3:9])
subplots.update_layout({
'plot_bgcolor': 'rgba(0, 0, 0, 0)',
'paper_bgcolor': 'rgba(0, 0, 0, 0)',
})
subplots.show()
```



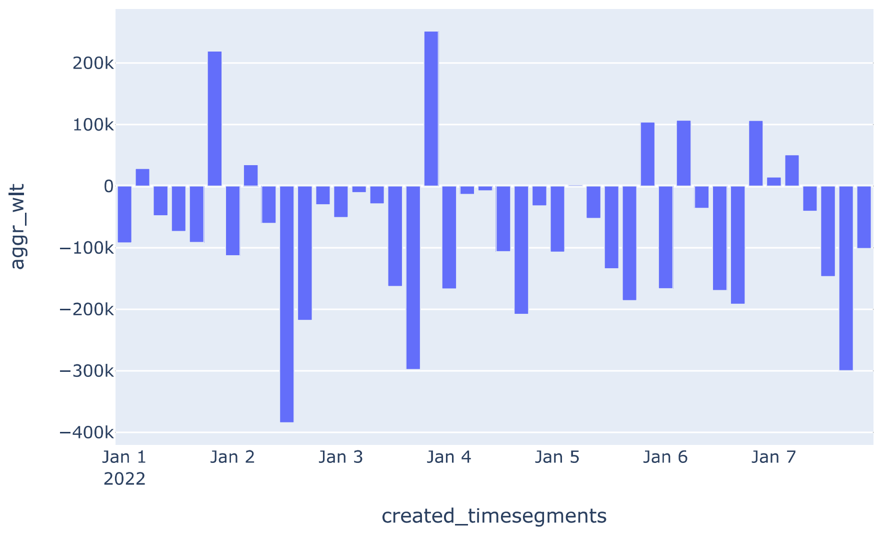
Remarks:

- From the above plots, one can see that in general, customers get more positive transactions when playing on the sportsbook product. This means that customers tend to win more from Sportsbook than from Games of Chance.
- One can also note that most transactions classified as *other* are positive. This is because most of these transactions are related to bonuses given to customers, therefore money is gained from such transactions.

Show aggregated wallet data

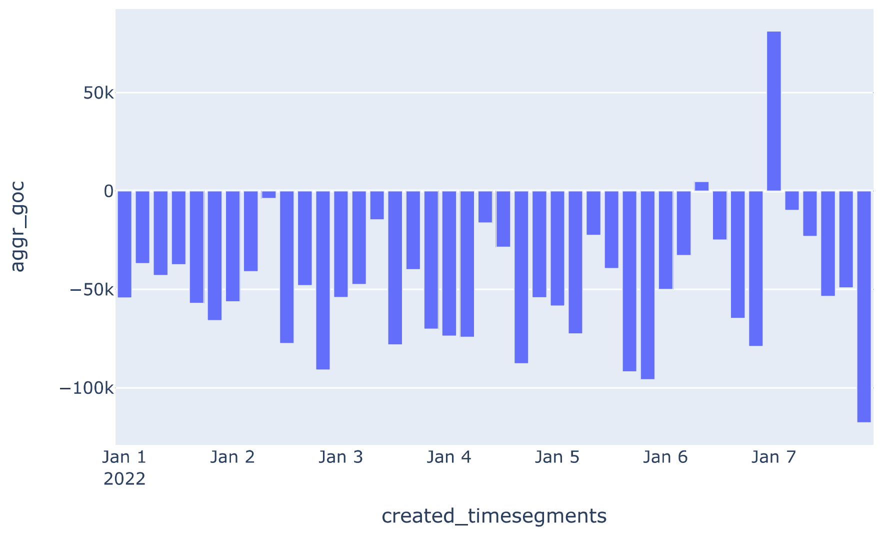
The below plot shows a sum of all the wallet transactions per timestamp. Here, we can see how customer's wallets were affected in the particular timestamp.

9
```
import plotly.express as px
grouped_data['aggr_wlt'] = grouped_data['sum_p_GOC'] + grouped_data['sum_p_O'] + grouped_data['sum_p_

px.bar(grouped_data, x='created_timesegments', y='aggr_wlt')
```

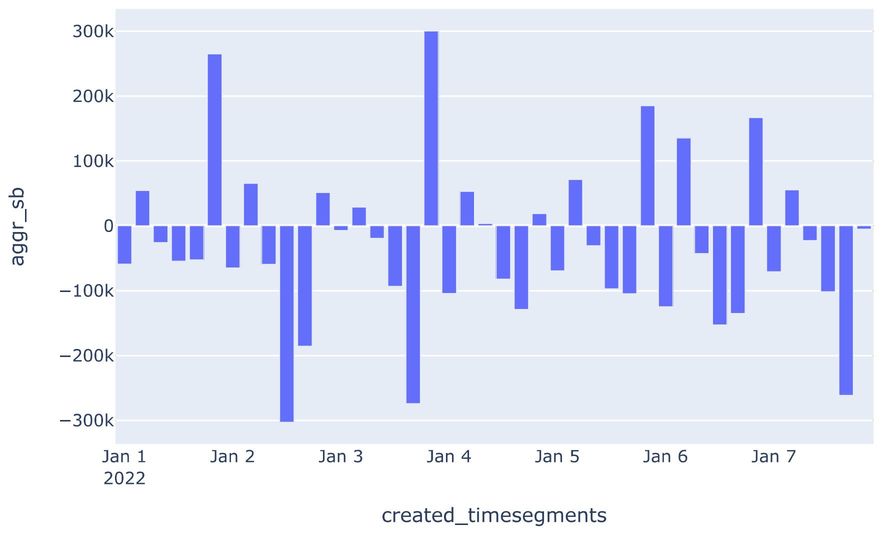


Remarks:\ One can note that in general, a customer's wallet is affected negatively in a timestamp more often than it is affected positively.

```
grouped_data['aggr_goc'] = grouped_data['sum_p_GOC']  - grouped_data['sum_n_GOC']

px.bar(grouped_data, x='created_timesegments', y='aggr_goc')
```



```
11  grouped_data['aggr_sb'] = grouped_data['sum_p_SB']  - grouped_data['sum_n_SB']

    px.bar(grouped_data, x='created_timesegments', y='aggr_sb')
```



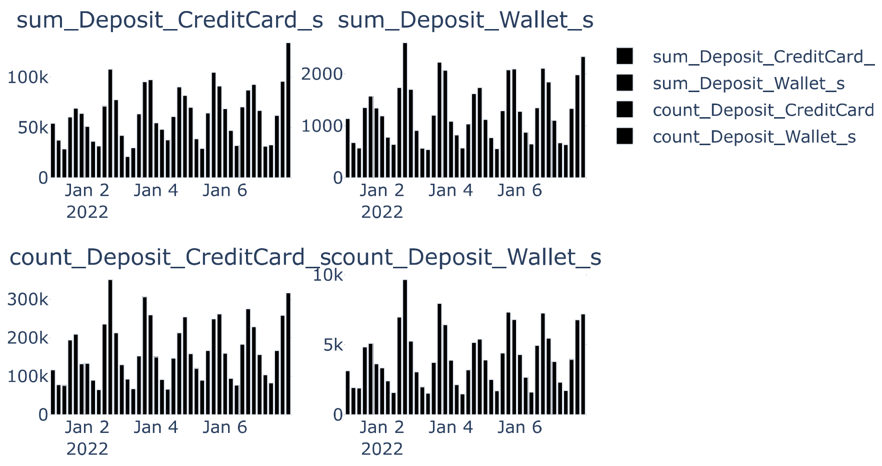
Remarks:\ As was also remarked previously, the graphs above also show that a customer is more likely to have positive transactions in their wallet when betting on SB.

Payments Data Analysis

When a customer attempts to make a payment transaction, it is not always successful.

The visualisations below show the distribution of **successful deposits** by payment method type across the different timestamps.

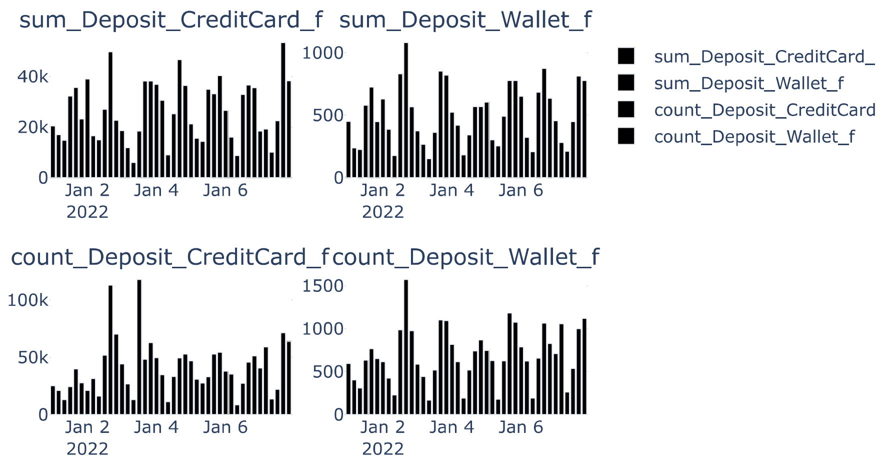
12
```
import plotly.io as pio
subplots = get_subplots(grouped_data,2,2,'created_timesegments', ['sum_Deposit_CreditCard_s', 'sum_De
subplots.update_layout({
'plot_bgcolor': 'rgba(0, 0, 0, 0)',
'paper_bgcolor': 'rgba(0, 0, 0, 0)',
})
subplots.show()
```



Remarks:\ Upon first glance, one can see a pattern across time for both the amounts and counts of successful deposit transactions.

The visualisations below show the distribution of **failed deposits** by payment method type across the different timestamps.

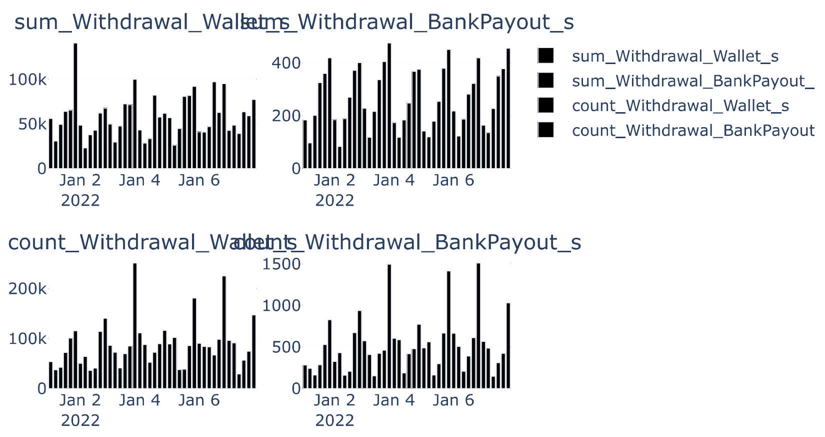
13
```
subplots = get_subplots(grouped_data,2,2,'created_timesegments', ['sum_Deposit_CreditCard_f', 'sum_De
subplots.update_layout({
'plot_bgcolor': 'rgba(0, 0, 0, 0)',
'paper_bgcolor': 'rgba(0, 0, 0, 0)',
})
subplots.show()
```



Remarks:\ A pattern across time may be seen for failed deposits, similar to that for successful ones.

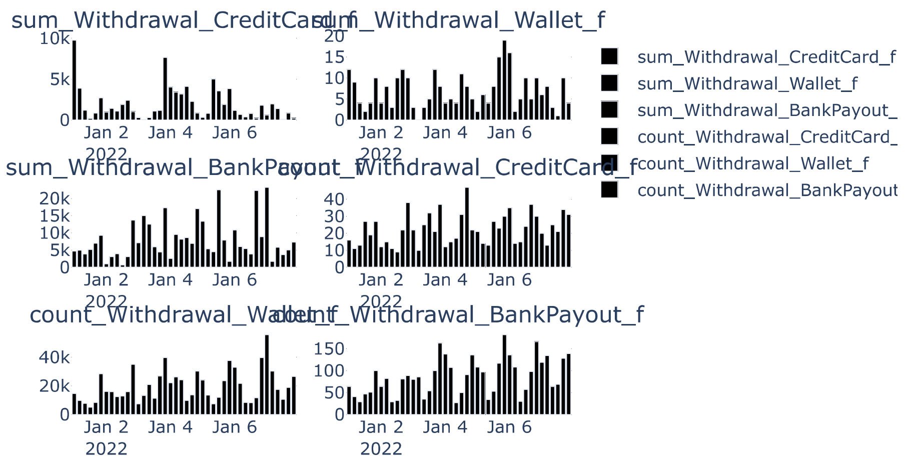
The visualisations below show the distribution of **successful withdrawals** by payment method type across the different timestamps.

14
```
subplots = get_subplots(grouped_data,2,2,'created_timesegments', ['sum_Withdrawal_Wallet_s', 'sum_Wit
subplots.update_layout({
'plot_bgcolor': 'rgba(0, 0, 0, 0)',
'paper_bgcolor': 'rgba(0, 0, 0, 0)',
})
subplots.show()
```



The visualisations below show the distribution of **failed withdrawals** by payment method type across the different timestamps.

```
subplots = get_subplots(grouped_data,3,2,'created_timesegments', ['sum_Withdrawal_CreditCard_f', 'sum_
subplots.update_layout({
'plot_bgcolor': 'rgba(0, 0, 0, 0)',
'paper_bgcolor': 'rgba(0, 0, 0, 0)',
})
subplots.show()
```



Following the above analysis, an analysis on the successful transactions based solely on **time** (rather than date-time) will be performed below :

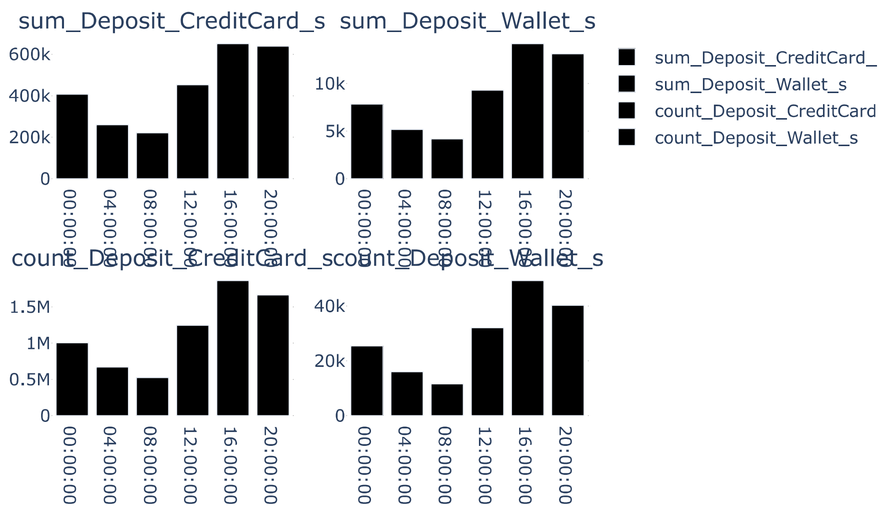
16
```
time_grouping = grouped_data
time_grouping['time'] = pd.to_datetime(time_grouping['created_timesegments']).dt.time
time_grouping = time_grouping.drop(['created_timesegments'], axis=1)
time_grouping = time_grouping.groupby(["time"]).agg('sum')
time_grouping = time_grouping.reset_index()
time_grouping.head()
```

16

	time	sum_n_SB	sum_p_SB	sum_n_GOC	sum_p_GOC	sum_p_O	sum_
0	00:00:00	1.693982e+06	1.194274e+06	9.640811e+06	9.375567e+06	99064.9541	16169.(
1	04:00:00	1.144580e+06	1.608466e+06	6.935791e+06	6.621108e+06	56709.0854	6566.7
2	08:00:00	6.758920e+05	4.791249e+05	4.963858e+06	4.845797e+06	49941.7401	8728.3(
3	12:00:00	2.194072e+06	1.310054e+06	6.528495e+06	6.189394e+06	67107.6680	22309.!
4	16:00:00	3.938720e+06	2.796595e+06	9.030355e+06	8.591944e+06	114540.0829	26760.(

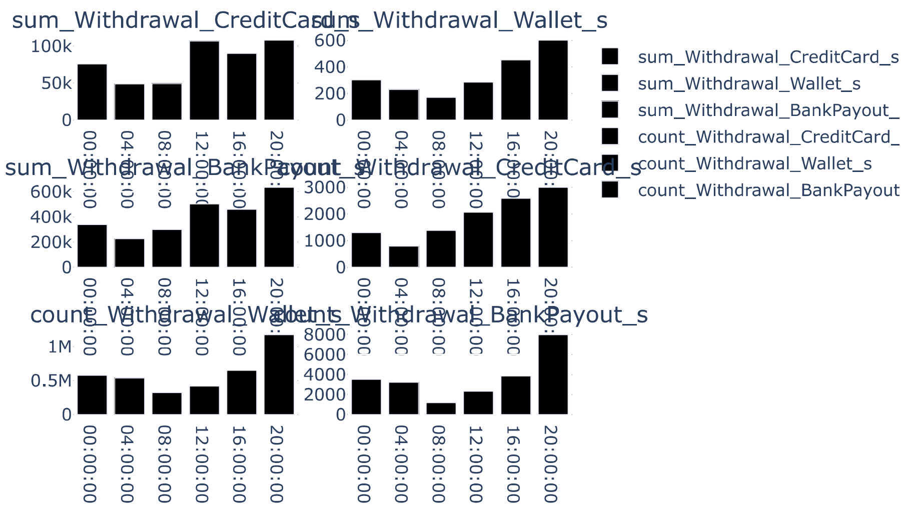
5 rows × 34 columns

17
```
subplots = get_subplots(time_grouping,2,2,'time', ['sum_Deposit_CreditCard_s', 'sum_Deposit_Wallet_s'
subplots.update_layout({
'plot_bgcolor': 'rgba(0, 0, 0, 0)',
'paper_bgcolor': 'rgba(0, 0, 0, 0)',
})
subplots.show()
```



Remarks:\ The above visualizations also show a pattern as seen in the previous ones. It may be seen clearer that from the 8am until the 4pm time segments, the deposit amounts and counts increase. After the 8pm segment, these start to decrease until the 8am segment.\ The time segment with the most deposits, both in count and amount is the 4pm segment for both method types.

18
```
subplots = get_subplots(time_grouping,3,2,'time', ['sum_Withdrawal_CreditCard_s', 'sum_Withdrawal_Wal
subplots.update_layout({
'plot_bgcolor': 'rgba(0, 0, 0, 0)',
'paper_bgcolor': 'rgba(0, 0, 0, 0)',
})
subplots.show()
```



Remarks:\ A pattern may also be seen for withdrawals, however it is different from that of deposits.\ In this case, the deposit amounts usually increase until the 8pm segment and decrease until the 8am segment.\ The time segment where most successful withdrawals are made is the 8pm segment.

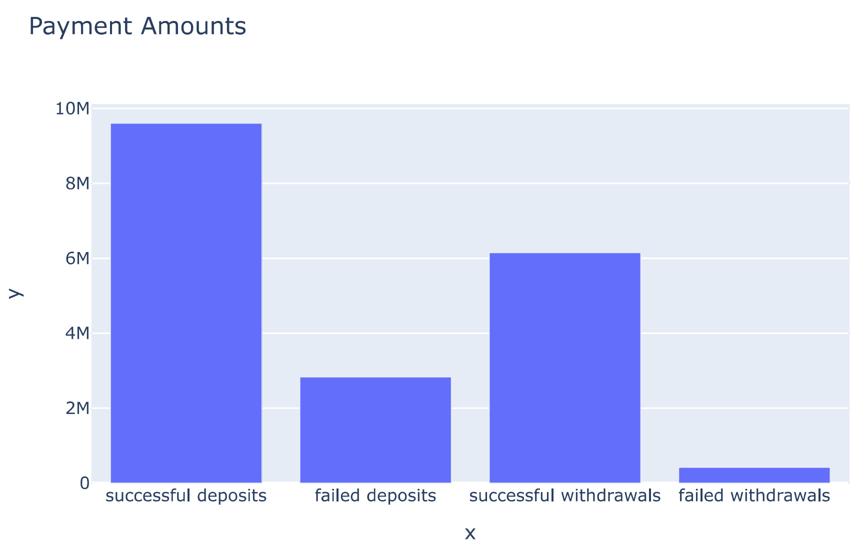
Payments Aggregate Data Analysis

In this section of the analysis, payments data will be grouped in order to conduct a more generic analysis.

```
19  total_deposits_s_sum = grouped_data['sum_Deposit_CreditCard_s']+grouped_data['sum_Deposit_Wallet_s']
    total_deposits_s_cnt = grouped_data['count_Deposit_CreditCard_s']+grouped_data['count_Deposit_Wallet_
    total_deposits_f_sum = grouped_data['sum_Deposit_CreditCard_f']+grouped_data['sum_Deposit_Wallet_f']+
    total_deposits_f_cnt = grouped_data['count_Deposit_CreditCard_f']+grouped_data['count_Deposit_Wallet_
```

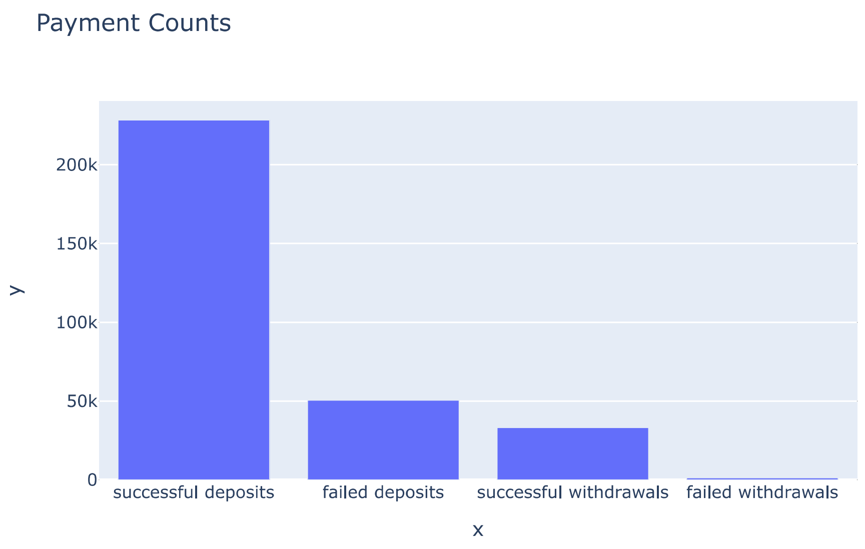
```
20  total_withdrawals_s_sum = grouped_data['sum_Withdrawal_BankPayout_s']+ grouped_data['sum_Withdrawal_W
    total_withdrawals_s_cnt = grouped_data['count_Withdrawal_BankPayout_s']+ grouped_data['count_Withdraw
    total_withdrawals_f_sum = grouped_data['sum_Withdrawal_CreditCard_f']+grouped_data['sum_Withdrawal_Wa
    total_withdrawals_f_cnt = grouped_data['count_Withdrawal_CreditCard_f']+grouped_data['count_Withdrawa
```

```
21  fig = px.bar(x=['successful deposits', 'failed deposits', 'successful withdrawals', 'failed withdrawa
    fig.show()
```



Remarks:\ The above visualization shows us that more deposits are made than withdrawals.

```
22 fig = px.bar(x=['successful deposits', 'failed deposits', 'successful withdrawals', 'failed withdrawa
   fig.show()
```



Remarks:\ From the above visualization, we can also see that the count of successful deposits is also greater than that of successful withdrawals. One may also note that whilst the difference in counts between deposits and withdrawals seems to be greater than the differences in the amounts of deposits and withdrawals. This shows that the average amount of withdrawals is most likely greater than the average amount of deposits.

Showing the payment amount **averages** in order to confirm the above remark.

```
23 print('Average amount of successful deposits: €', total_deposits_s_sum.sum()/total_deposits_s_cnt.sum
print('Average amount of failed deposits: €', total_deposits_f_sum.sum()/total_deposits_f_cnt.sum())
print('Average amount of successful withdrawals: €', total_withdrawals_s_sum.sum()/total_withdrawals_
print('Average amount of failed withdrawals: €', total_withdrawals_f_sum.sum()/total_withdrawals_f_cn
```

```
Average amount of successful deposits: € 42.05066788883146
Average amount of failed deposits: € 56.06199119354775
Average amount of successful withdrawals: € 185.08948746612873
Average amount of failed withdrawals: € 339.48453135179153
```

Remarks:\ As expected, the average amount of a withdrawal is considerabely larger than the average amount of a deposit.

The *unprocessed* payments dataset will be used for easier aggregations.

```
24 pmts_data = pd.read_csv('../Data/pmt_jan1_masked.csv', header=None,
                   names=['customerGuid','sk_pmtTransactionInfo','countryCode','pmtCreatedDate','pmtS


pmts_data['status'] = np.where(np.logical_or(pmts_data['sk_pmtStatus'] == 3, pmts_data['sk_pmtStatus'
```

```
C:\Users\frans\.conda\envs\python_3_7\lib\site-packages\IPython\core\interactiveshell.py:3457: DtypeWa

Columns (0,2,3,4,6,7) have mixed types.Specify dtype option on import or set low_memory=False.
```

Grouping the transactions by payment type, payment method type and payment status for a more general analysis.

```
25 group_transactions = pmts_data.groupby(["pmtTypeName","pmtMethodTypeName","status"]).pmtAmount_EUR.ag
group_transactions = group_transactions.reset_index()
print(group_transactions)
```

```
   pmtTypeName pmtMethodTypeName status           sum   count
0      Deposit              Bank      f  2.000000e+02       4
1      Deposit        CreditCard      f  1.107356e+06   21349
2      Deposit        CreditCard      s  2.630445e+06   53792
3      Deposit            Wallet      f  1.721500e+06   29110
4      Deposit            Wallet      s  6.971320e+06  174546
5   Withdrawal        BankPayout      f  8.304786e+05    3615
6   Withdrawal        BankPayout      s  3.682187e+06   22093
7   Withdrawal        CreditCard      f  8.140062e+04     290
8   Withdrawal        CreditCard      s  4.787010e+05    2042
9   Withdrawal            Wallet      f  3.354864e+05     938
10  Withdrawal            Wallet      s  2.465375e+06   11121
```

26
```
deposits_grouped = group_transactions[group_transactions.pmtTypeName == 'Deposit']
deposits_grouped.head()
```

26

	pmtTypeName	pmtMethodTypeName	status	sum	count
0	Deposit	Bank	f	2.000000e+02	4
1	Deposit	CreditCard	f	1.107356e+06	21349
2	Deposit	CreditCard	s	2.630445e+06	53792
3	Deposit	Wallet	f	1.721500e+06	29110
4	Deposit	Wallet	s	6.971320e+06	174546

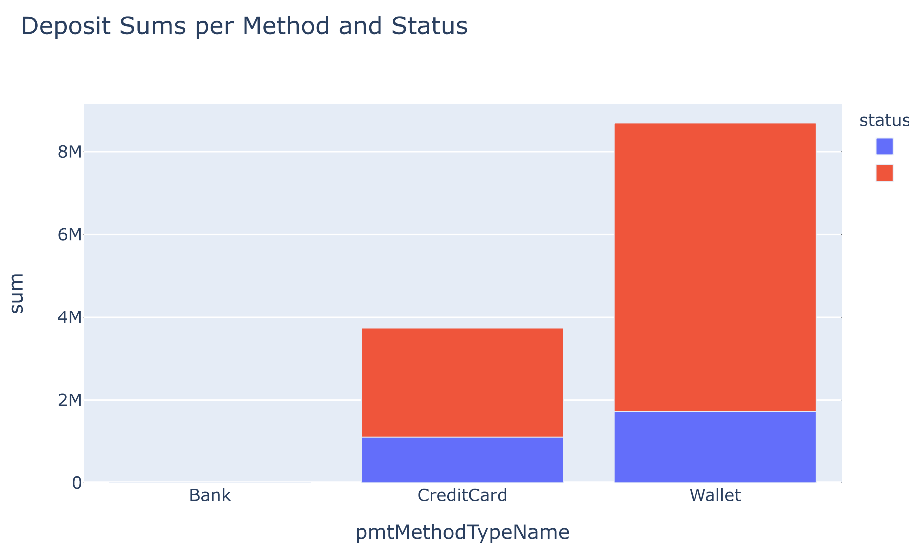
27
```
withdrawals_grouped = group_transactions[group_transactions.pmtTypeName == 'Withdrawal']
withdrawals_grouped.head()
```

27

	pmtTypeName	pmtMethodTypeName	status	sum	count
5	Withdrawal	BankPayout	f	8.304786e+05	3615
6	Withdrawal	BankPayout	s	3.682187e+06	22093
7	Withdrawal	CreditCard	f	8.140062e+04	290
8	Withdrawal	CreditCard	s	4.787010e+05	2042
9	Withdrawal	Wallet	f	3.354864e+05	938

The visualization below shows the share of successful and failed deposits by payment method type.

28
```
fig = px.bar(deposits_grouped,x="pmtMethodTypeName", y="sum", color='status', barmode = 'stack', titl
fig.show()
```



Remarks:\ One may note that for deposits using credit cards, the proportion of failures compared to successful transactions is much larger than that for wallet transactions.

Investigating whether there is any relationship between the product a customer plays on and their payment behaviour

Get payments of customers who have bet on GOC

29
```
goc_cust = data[data['sum_n_GOC']>0]
goc_cust = goc_cust['customerGuid'].drop_duplicates()
#goc_pmt = data.join(goc_cust, on=['customerGuid', 'customerGuid'], how='inner', lsuffix = '_pmt', rs
goc_pmt= pd.merge(goc_cust, data, how='inner', left_on=['customerGuid', 'customerGuid'], right_on=['c

goc_pmt_grouped = goc_pmt.drop(['customerGuid', 'countryCode'], axis=1)
goc_pmt_grouped = goc_pmt_grouped.groupby(["created_timesegments"]).agg('sum')
goc_pmt_grouped = goc_pmt_grouped.reset_index()

goc_pmt_grouped.head()
```

29

	created_timesegments	sum_n_SB	sum_p_SB	sum_n_GOC	sum_p_GOC	sum_p_O
0	2022-01-01 00:00:00	46793.3382	22659.7577	1.246075e+06	1.191823e+06	20343.4093
1	2022-01-01 04:00:00	27645.6796	55208.1531	7.917224e+05	7.549152e+05	10928.6703
2	2022-01-01 08:00:00	29392.9818	23228.4372	7.268918e+05	6.840198e+05	20790.9939
3	2022-01-01 12:00:00	84349.3066	66905.8381	8.356710e+05	7.983196e+05	18487.0921
4	2022-01-01 16:00:00	126874.6231	118340.8610	1.125939e+06	1.068906e+06	17501.7977

5 rows × 29 columns

Get payments of customers who bet on SB

30
```
sb_cust = data[data['sum_n_SB']>0]
sb_cust = sb_cust['customerGuid'].drop_duplicates()
sb_pmt= pd.merge(sb_cust, data, how='inner', left_on=['customerGuid', 'customerGuid'], right_on=['cus

sb_pmt_grouped = sb_pmt.drop(['customerGuid', 'countryCode'], axis=1)
sb_pmt_grouped = sb_pmt_grouped.groupby(["created_timesegments"]).agg('sum')
sb_pmt_grouped = sb_pmt_grouped.reset_index()

sb_pmt_grouped.head()
```

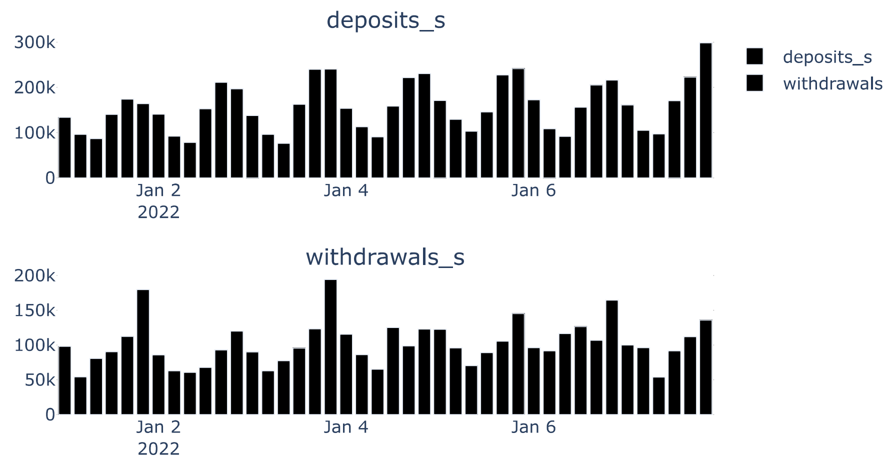
30

	created_timesegments	sum_n_SB	sum_p_SB	sum_n_GOC	sum_p_GOC	sum_p_O
0	2022-01-01 00:00:00	169475.8784	110355.1984	292300.1613	282205.4893	11343.3273
1	2022-01-01 04:00:00	99690.0067	153720.0352	188354.3685	178274.1563	4121.9978
2	2022-01-01 08:00:00	93875.2902	68078.8775	103983.8195	95721.9478	6092.2109
3	2022-01-01 12:00:00	370301.3039	315699.3705	142068.9598	128987.9168	9133.4679
4	2022-01-01 16:00:00	491990.9490	439542.1253	236670.9607	236747.6983	7790.9360

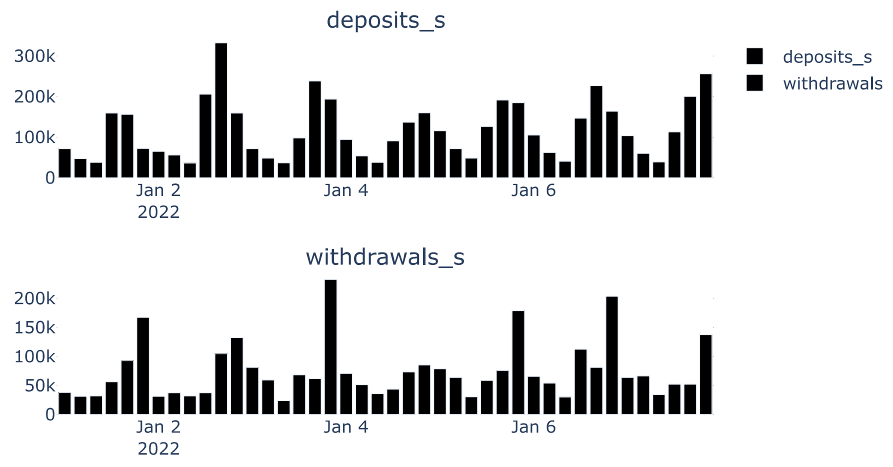
5 rows × 29 columns

31
```
print('GOC')
goc_pmt_grouped['deposits_s'] = goc_pmt_grouped['sum_Deposit_CreditCard_s']+goc_pmt_grouped['sum_Depo
goc_pmt_grouped['withdrawals_s'] =goc_pmt_grouped['sum_Withdrawal_BankPayout_s']+ goc_pmt_grouped['sur
subplots = get_subplots(goc_pmt_grouped,2,1,'created_timesegments', ['deposits_s', 'withdrawals_s']) #
subplots.update_layout({
'plot_bgcolor': 'rgba(0, 0, 0, 0)',
'paper_bgcolor': 'rgba(0, 0, 0, 0)',
})
subplots.show()

print('SB')
sb_pmt_grouped['deposits_s'] = sb_pmt_grouped['sum_Deposit_CreditCard_s']+sb_pmt_grouped['sum_Deposit_
sb_pmt_grouped['withdrawals_s'] =sb_pmt_grouped['sum_Withdrawal_BankPayout_s']+ sb_pmt_grouped['sum_W
subplots = get_subplots(sb_pmt_grouped,2,1,'created_timesegments', ['deposits_s', 'withdrawals_s']) #
subplots.update_layout({
'plot_bgcolor': 'rgba(0, 0, 0, 0)',
'paper_bgcolor': 'rgba(0, 0, 0, 0)',
})
subplots.show()
```

```
GOC
```



SB



```
32  print('Total amount of deposits from customers who bet on GOC :','{:,}'.format(goc_pmt_grouped['depos
    print('Total amount of withdrawals from customers who bet on GOC :', '{:,}'.format(goc_pmt_grouped['w
    print('Total amount of deposits from customers who bet on SB :', '{:,}'.format(sb_pmt_grouped['deposi
    print('Total amount of withdrawals from customers who bet on SB :', '{:,}'.format(sb_pmt_grouped['wit
Total amount of deposits from customers who bet on GOC : 6,609,574.797799999
Total amount of withdrawals from customers who bet on GOC : 4,280,581.945800001
Total amount of deposits from customers who bet on SB : 4,921,328.4236
Total amount of withdrawals from customers who bet on SB : 3,118,503.8054
```

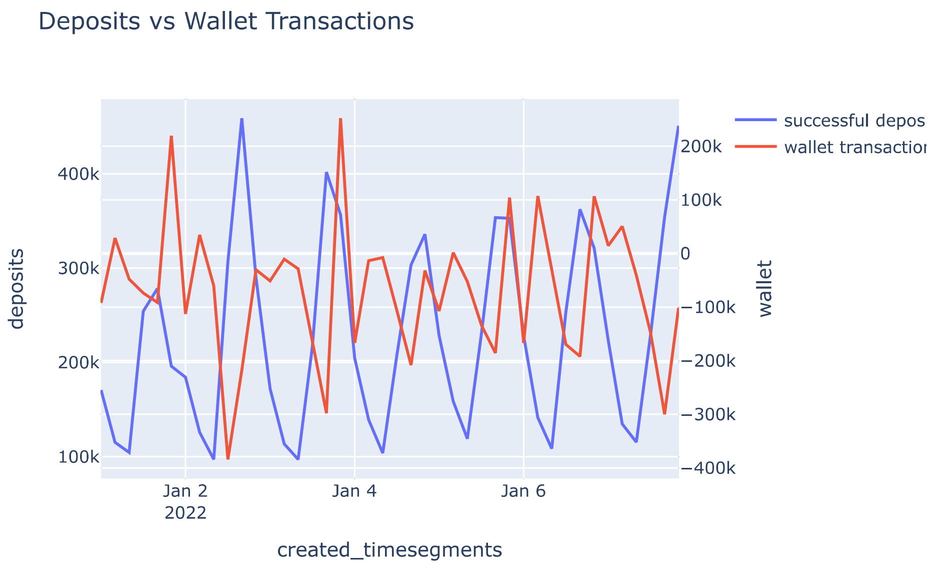
Remarks\ From the above visualizations, we can see that for customers playing GOC, there seems to be a steadier flow of payments. This is due to the fact that most SB customers are more likely to deposit/withdraw when events are taking place.

Investigating the relationship between deposits, bets & wins\ This is a function which returns a graph with 2 axes - allowing for comparison of values on different scales.

33
```
def create_2axes(x, y1, y2, y1_line, y2_line, y1_axis, y2_axis, x_axis, title):
    # Create figure with secondary y-axis
    fig = make_subplots(specs=[[{"secondary_y": True}]])

    # Add traces
    fig.add_trace(
        go.Scatter(x=x, y=y1, name=y1_line),
        secondary_y=False,
    )

    fig.add_trace(
        go.Scatter(x=x, y=y2, name=y2_line),
        secondary_y=True,
    )

    # Add figure title
    fig.update_layout(
        title_text=title
    )

    # Set x-axis title
    fig.update_xaxes(title_text=x_axis)

    # Set y-axes titles
    fig.update_yaxes(title_text=y1_axis, secondary_y=False)
    fig.update_yaxes(title_text=y2_axis, secondary_y=True)

    return fig
```

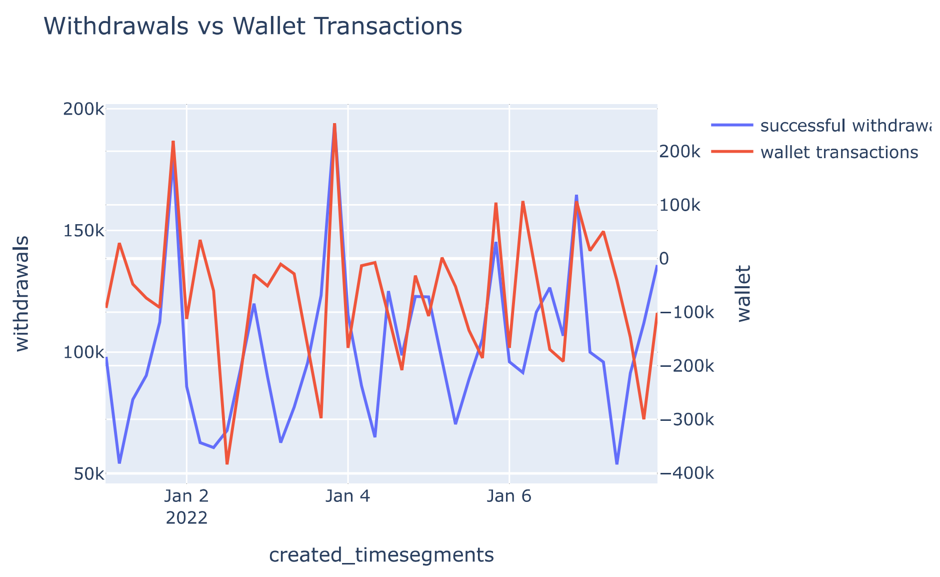
1. General Relationship between wallet transactions and deposits

34
```
fig = create_2axes(grouped_data['created_timesegments'], grouped_data['deposits_s'], grouped_data['ag
                  'successful deposits', 'wallet transactions', 'deposits', 'wallet', 'created_timese
                   'Deposits vs Wallet Transactions')
fig.show()
```



2. General Relationship between wallet transactions and withdrawals

```
35  fig = create_2axes(grouped_data['created_timesegments'], goc_pmt_grouped['withdrawals_s'], grouped_da
                      'successful withdrawals', 'wallet transactions', 'withdrawals', 'wallet', 'created_
                       'Withdrawals vs Wallet Transactions')
    fig.show()
```



Remarks:\ One can note a negative relationship between the wallet transactions and successful deposits. This ultimately shows that when a lot of negative transactions are affecting customers' wallets, thus lowering their balances, customers tend to deposit more.

Such a relationship cannot be seen between the wallet transactions and withdrawals. This visulaization shows that sometimes these values are directly proportional, whilst at other times they are indirectely proportional.

Scaling Values

Values scaled using StandardScaler from sklearn - which uses Z-score normalizaion - *Z-score normalization refers to the process of normalizing every value in a dataset such that the mean of all of the values is 0 and the standard deviation is 1.*

36
```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

scaled_features = scaler.fit_transform(data.iloc[:,3:])

scaled_data = pd.DataFrame(scaled_features)
scaled_data.insert(0, 'customerGuid', data['customerGuid'])
scaled_data.insert(1, 'countryCode', data['countryCode'])
scaled_data.insert(2, 'created_timesegments', data['created_timesegments'])

scaled_data.columns = data.columns
scaled_data.head()
```

```
#print(scaled_data)
```

36

	customerGuid	countryCode	created_timesegments	sum_n_SB	sum_p_SB	sum_n_GOC
0	00002912-####-####-####-B070	PE	2022-01-02 16:00:00	-0.088454	-0.082947	-0.090681
1	00002912-####-####-####-B070	PE	2022-01-02 20:00:00	-0.071328	-0.082947	-0.090681
2	00002912-####-####-####-B070	PE	2022-01-05 20:00:00	-0.096914	-0.082947	-0.090681
3	00002912-####-####-####-B070	PE	2022-01-06 16:00:00	-0.062213	-0.082947	-0.090681
4	000069A8-####-####-####-48FB	PE	2022-01-02 20:00:00	-0.100447	-0.082947	-0.090681

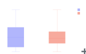
5 rows × 31 columns

37
```
import plotly.graph_objs as go
from plotly.offline import plot

def get_plot(df, columns, date, title):
    fig = go.Figure()
    scatter = go.Scatter(x=df[date],
                         mode='lines', name=title,
                         opacity=0.9)
    #fig.add_trace(scatter)
    for c in columns :

      fig.add_scatter(x=df[date], y=df[c], mode='lines', name=c)
    #fig.add_scatter(x=df['date'], y=df['Temperature'], mode='lines')

    fig = fig.update_xaxes(rangeslider_visible=True)
    fig.update_layout(title_text=title, title_font_family='Verdana')
    #fig.show()
    return fig
```

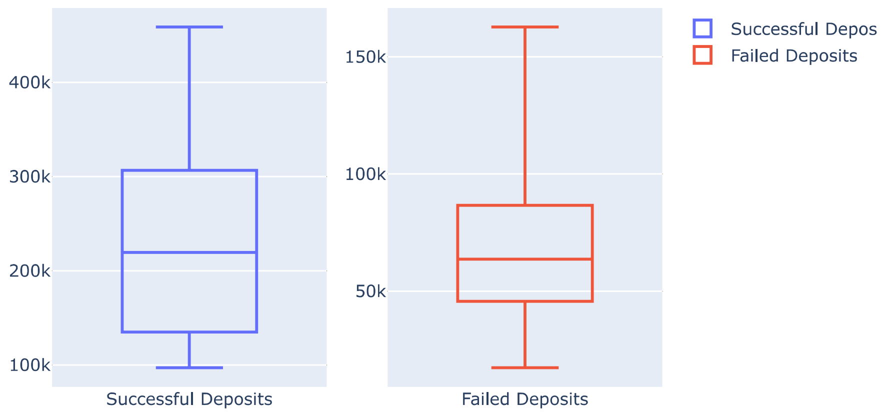
Box Plots

Box Plots for totals per day

38
```
fig = make_subplots(rows=1, cols=2)

trace2 = go.Box(y=total_deposits_s_sum, name='Successful Deposits', boxpoints=False)
fig.append_trace(trace2, row=1, col=1)
```



```
trace3 = go.Box(y=total_deposits_f_sum, name='Failed Deposits',boxpoints=False)
fig.append_trace(trace3, row=1, col=2)

fig.show()
```

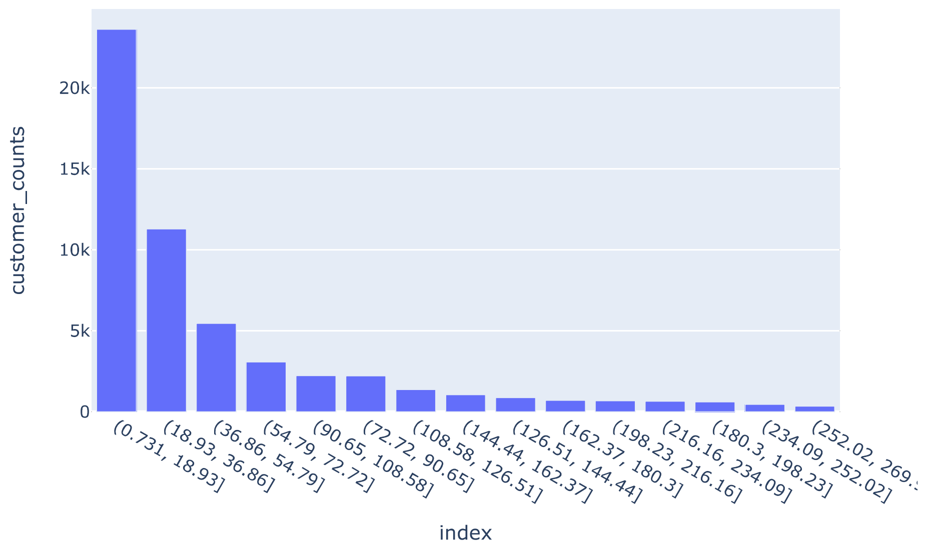


Box Plots for total payments per customer

```
39  customer_data = data.drop(['created_timesegments', 'countryCode'], axis=1)
    customer_data = customer_data.groupby(["customerGuid"]).agg('sum')
    customer_data = customer_data.reset_index()
    customer_data['deposits_s'] = customer_data['sum_Deposit_CreditCard_s']+customer_data['sum_Deposit_Wa
    customer_data['withdrawals_s'] =customer_data['sum_Withdrawal_BankPayout_s']+ customer_data['sum_With
```

We will remove outliers to get a better picture.

```
40  pd.options.mode.chained_assignment = None  # default='warn'

    q_low = customer_data["deposits_s"].quantile(0.01)
    q_hi  = customer_data["deposits_s"].quantile(0.9)

    bucket_data = customer_data[(customer_data["deposits_s"] < q_hi) & (customer_data["deposits_s"] > q_l

    bucket_data['customer_counts'] = pd.cut(bucket_data['deposits_s'], bins=15)
    quantile_counts = bucket_data['customer_counts'].value_counts().to_frame().reset_index()
    quantile_counts['index'] = quantile_counts['index'].astype('str')
    px.bar(quantile_counts, x='index', y='customer_counts')
```

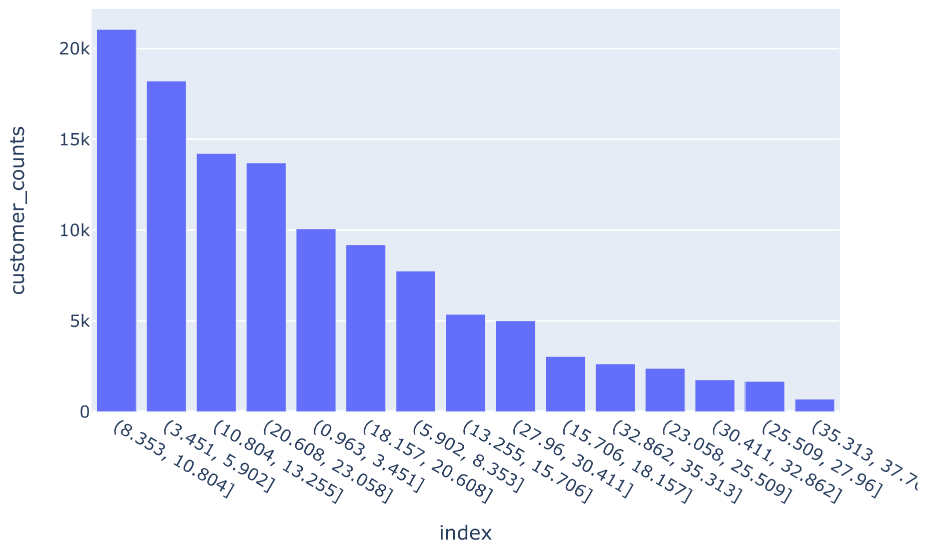


Remarks:\ From the above plot, we can see that the higher the total number of deposits, the lower the customer count. This means that most customers do not deposit high amounts.

Box Plots for total payments per time-segment per customer

41

```
data['deposits_s'] = data['sum_Deposit_CreditCard_s']+data['sum_Deposit_Wallet_s']
data['withdrawals_s'] =data['sum_Withdrawal_BankPayout_s']+ data['sum_Withdrawal_Wallet_s']
q_low = data["deposits_s"].quantile(0.01)
q_hi  = data["deposits_s"].quantile(0.9)

bucket_data = data[(data["deposits_s"] < q_hi) & (data["deposits_s"] > q_low)]

bucket_data['customer_counts'] = pd.cut(bucket_data['deposits_s'], bins=15)
quantile_counts = bucket_data['customer_counts'].value_counts().to_frame().reset_index()
quantile_counts['index'] = quantile_counts['index'].astype('str')
fig = px.bar(quantile_counts, x='index', y='customer_counts')
fig.show()
```



Remarks:\ From the above plot, we can see that the most common deposit amount per segment is 8.35 - 10.80. Most total successful deposits per time-segment per customer range between 1 and 38 EUR.

Correlation Analysis

What values should we check for correlation?

Here we can see that there is a strong negative correltion between positive wallet transactions and deposit transactions. This makes sense as when people are winning, they usually do not need to deposit.

One can also see a correlation between negative wallet transactions and successful deposit attmepts.

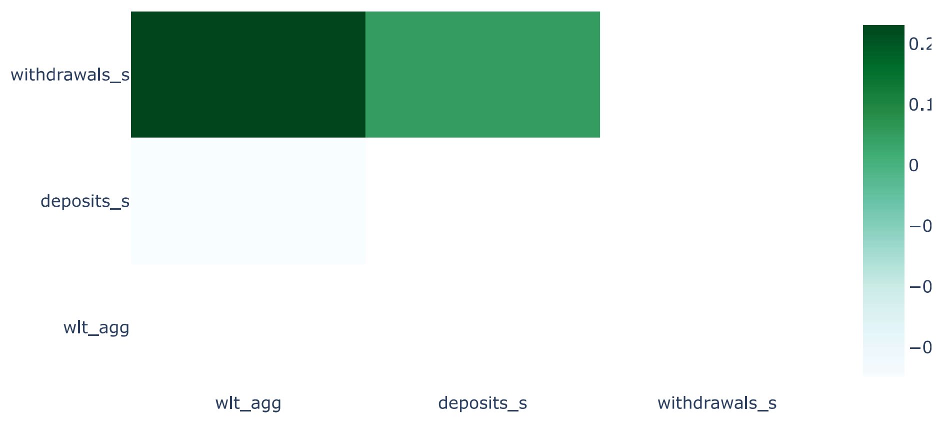
42
```
data.head()
data['wlt_agg'] =  data['sum_p_GOC'] + data['sum_p_O'] + data['sum_p_SB'] - data['sum_n_GOC'] - data[
totals = [data['wlt_agg'], data['deposits_s'], data['withdrawals_s']]
totals_df = pd.concat(totals, axis=1)
totals_df.head()
```

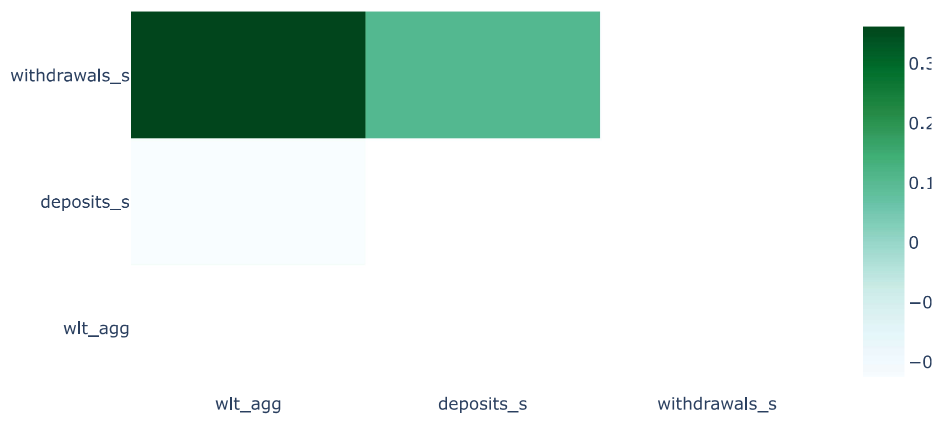
42

	wlt_agg	deposits_s	withdrawals_s
0	-4.4103	0.0000	0.0
1	-8.8188	0.0000	0.0

	wlt_agg	deposits_s	withdrawals_s
2	-2.2324	0.0000	0.0
3	-11.1653	0.0000	0.0
4	-1.3228	4.4094	0.0

43
```
import plotly.io as pio
import plotly.express as px
import plotly.graph_objects as go

def get_corr_heatmap(df, corr_type):

  corr = df.corr(method=corr_type)
  pio.templates.default = "plotly_white"

  mask = np.triu(np.ones_like(corr, dtype=bool))

  fig = go.Figure(go.Heatmap(
      z=corr.mask(mask),
      x=corr.columns,
      y=corr.columns,
      colorscale=px.colors.sequential.BuGn,

  ))

  fig.update_layout(xaxis_showgrid=False, yaxis_showgrid=False)

  return fig

spearman = get_corr_heatmap(totals_df, 'spearman')
spearman.show()

pearson = get_corr_heatmap(totals_df, 'pearson')
pearson.show()

# see how to get better correlation analysis with aggregated data
```

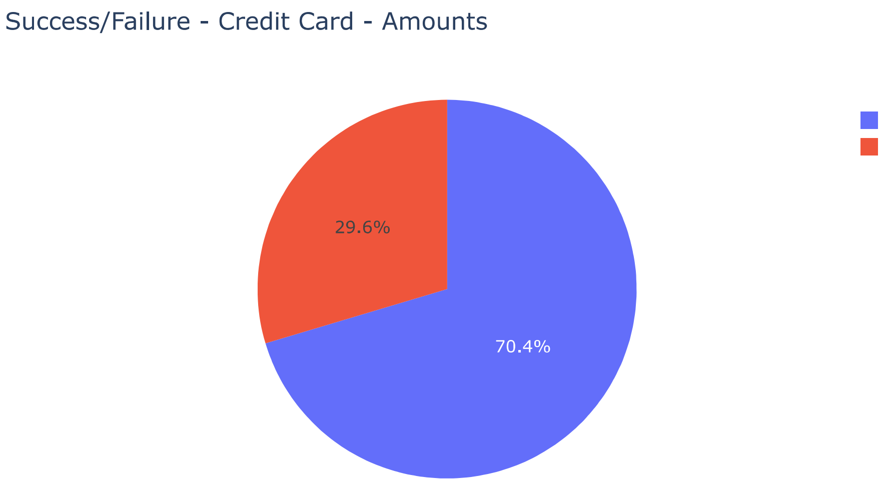


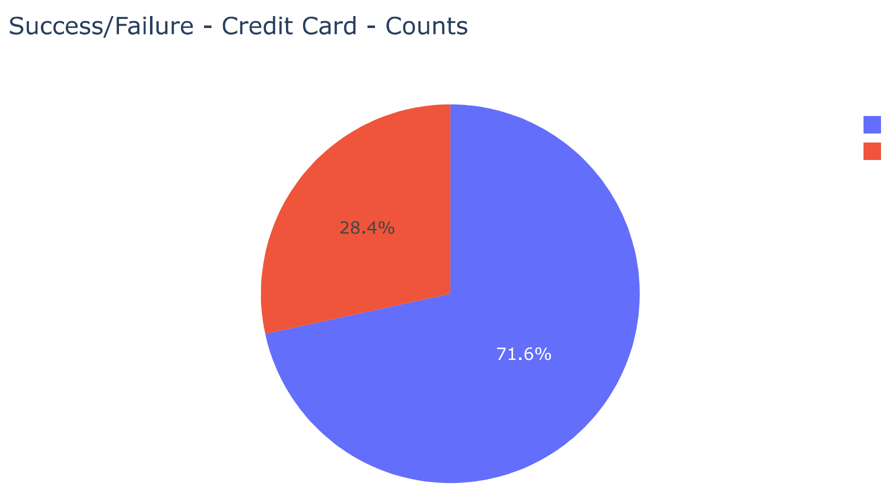


Remarks:\ Using both correlation tests, we can see a positive correlation between the successful withdrawal amounts and the total wallet transaction amounts and a negative one between the successful deposit amoutns and the total wallet transaction amounts. This makes sense since customers will typically withdraw after positive transactions to their wallet and deposit after negative ones.

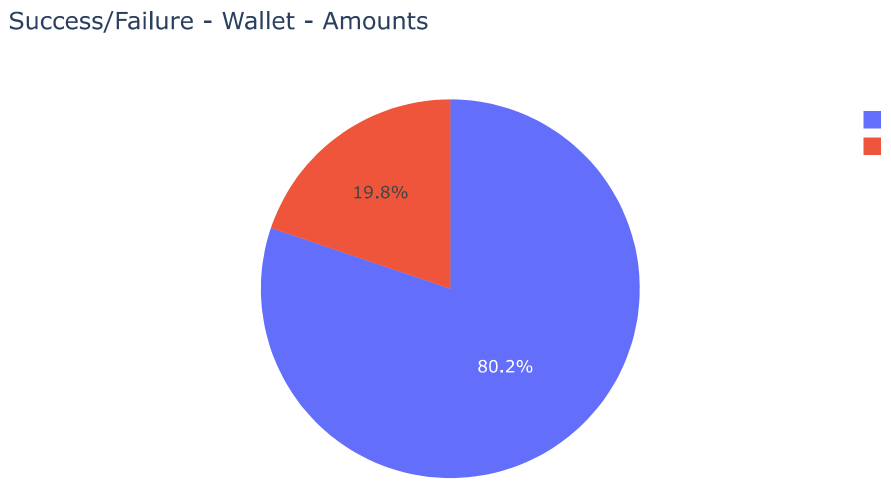
Inspect correlation between method type and success/failure

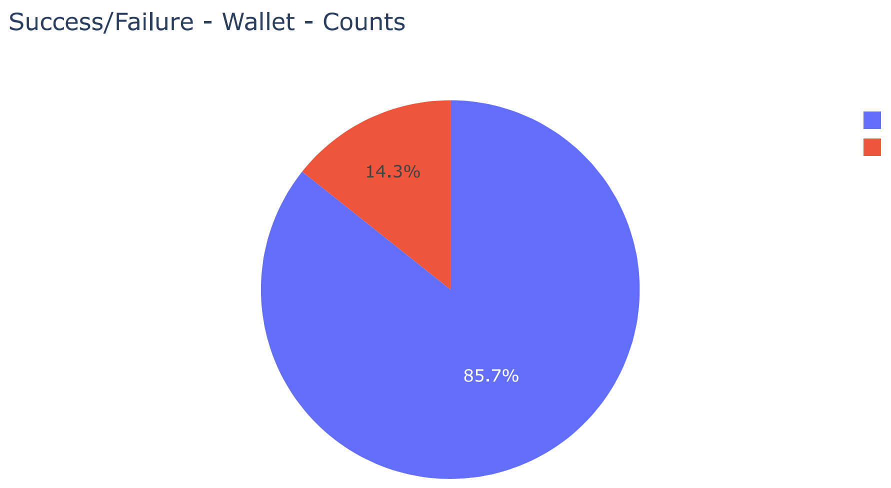
44
```
deposits_grouped.head()

fig = px.pie(deposits_grouped[deposits_grouped.pmtMethodTypeName == 'CreditCard'], values='sum', name
fig.show()

fig2 = px.pie(deposits_grouped[deposits_grouped.pmtMethodTypeName == 'CreditCard'], values='count', n
fig2.show()
```





45
```
fig = px.pie(deposits_grouped[deposits_grouped.pmtMethodTypeName == 'Wallet'], values='sum', names='s
fig.show()

fig2 = px.pie(deposits_grouped[deposits_grouped.pmtMethodTypeName == 'Wallet'], values='count', names
fig2.show()
```





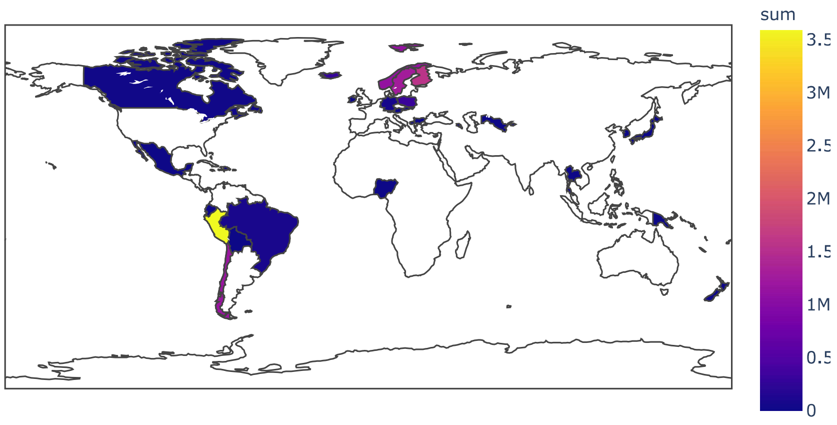
Data Distribution measures

46
```
stats = data.describe()
# transpose rows and columns
stats = stats.T
stats['kurtosis'] = data.kurtosis().tolist()
stats['var'] = data.var().tolist()
stats['skew'] = data.skew().tolist()

print(stats)
```

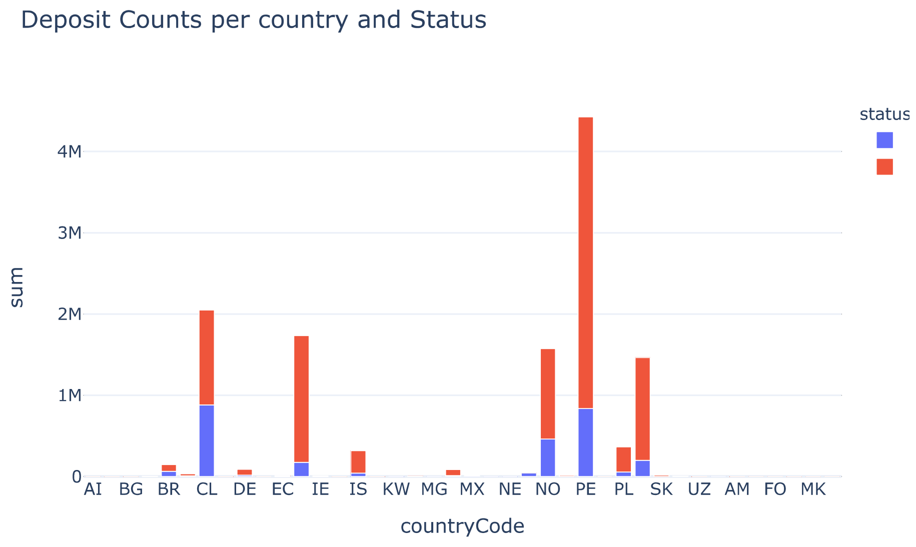
```
C:\Users\frans\.conda\envs\python_3_7\lib\site-packages\ipykernel_launcher.py:4: FutureWarning:

Dropping of nuisance columns in DataFrame reductions (with 'numeric_only=None') is deprecated; in a fu

C:\Users\frans\.conda\envs\python_3_7\lib\site-packages\ipykernel_launcher.py:5: FutureWarning:

Dropping of nuisance columns in DataFrame reductions (with 'numeric_only=None') is deprecated; in a fu


                                count        mean          std       min  \
sum_n_SB                     473498.0   27.180454   257.425359      0.00
sum_p_SB                     473498.0   24.485883   295.199335      0.00
sum_n_GOC                    473498.0  101.390219  1118.093587      0.00
sum_p_GOC                    473498.0   97.062349  1124.743616      0.00
sum_p_O                      473498.0    1.099816    33.130206      0.00
sum_n_O                      473498.0    0.220694     5.149499      0.00
sum_Deposit_Wallet_s         473498.0   14.723019   106.971477      0.00
sum_Deposit_CreditCard_s     473498.0    5.555347    73.074772      0.00
sum_Deposit_CreditCard_f     473498.0    2.338671    58.060720      0.00
sum_Deposit_Wallet_f         473498.0    3.635707   162.385188      0.00
sum_Withdrawal_BankPayout_f  473498.0    1.753922    86.674128      0.00
sum_Withdrawal_BankPayout_s  473498.0    7.776564   131.731629      0.00
sum_Withdrawal_Wallet_s      473498.0    5.206727   161.593586      0.00
sum_Withdrawal_Wallet_f      473498.0    0.708528    59.116203      0.00
sum_Withdrawal_CreditCard_s  473498.0    1.010988    46.773819      0.00
sum_Withdrawal_CreditCard_f  473498.0    0.171913    19.054868      0.00
sum_Deposit_Bank_f           473498.0    0.000422     0.223253      0.00
count_Deposit_Wallet_s       473498.0    0.368631     0.800848      0.00
count_Deposit_CreditCard_s   473498.0    0.113606     0.549789      0.00
count_Deposit_CreditCard_f   473498.0    0.045088     0.605573      0.00
count_Deposit_Wallet_f       473498.0    0.061479     0.400757      0.00
count_Withdrawal_BankPayout_f  473498.0    0.007635     0.144928      0.00
count_Withdrawal_BankPayout_s  473498.0    0.046659     0.246780      0.00
count_Withdrawal_Wallet_s    473498.0    0.023487     0.194431      0.00
count_Withdrawal_Wallet_f    473498.0    0.001981     0.064896      0.00
count_Withdrawal_CreditCard_s  473498.0    0.004313     0.078033      0.00
count_Withdrawal_CreditCard_f  473498.0    0.000612     0.032490      0.00
count_Deposit_Bank_f         473498.0    0.000008     0.002906      0.00
deposits_s                   473498.0   20.278365   130.163633      0.00
withdrawals_s                473498.0   12.983291   211.116549      0.00
wlt_agg                      473498.0   -6.143318   328.365557 -24193.53

                               25%     50%      75%          max  \
sum_n_SB                     0.000  2.2324  11.1622   96201.3439
sum_p_SB                     0.000  0.0000   3.8845   76725.5956
sum_n_GOC                    0.000  0.0000   0.0000  413729.0000
sum_p_GOC                    0.000  0.0000   0.0000  403599.0000
sum_p_O                      0.000  0.0000   0.0000    7963.1386
sum_n_O                      0.000  0.0000   0.0000     600.0000
sum_Deposit_Wallet_s         0.000  0.0000   4.1921   22410.8214
sum_Deposit_CreditCard_s     0.000  0.0000   0.0000   16102.5986
sum_Deposit_CreditCard_f     0.000  0.0000   0.0000   10665.3632
sum_Deposit_Wallet_f         0.000  0.0000   0.0000   83234.4004
sum_Withdrawal_BankPayout_f  0.000  0.0000   0.0000   33483.9366
sum_Withdrawal_BankPayout_s  0.000  0.0000   0.0000   43010.6587
sum_Withdrawal_Wallet_s      0.000  0.0000   0.0000   59558.6600
sum_Withdrawal_Wallet_f      0.000  0.0000   0.0000   21248.0579
sum_Withdrawal_CreditCard_s  0.000  0.0000   0.0000   14182.9707
sum_Withdrawal_CreditCard_f  0.000  0.0000   0.0000    6911.9524
sum_Deposit_Bank_f           0.000  0.0000   0.0000     150.0000
```

```
count_Deposit_Wallet_s          0.000  0.0000   1.0000      44.0000
count_Deposit_CreditCard_s      0.000  0.0000   0.0000      32.0000
count_Deposit_CreditCard_f      0.000  0.0000   0.0000     128.0000
count_Deposit_Wallet_f          0.000  0.0000   0.0000      55.0000
count_Withdrawal_BankPayout_f   0.000  0.0000   0.0000      19.0000
count_Withdrawal_BankPayout_s   0.000  0.0000   0.0000      42.0000
count_Withdrawal_Wallet_s       0.000  0.0000   0.0000      22.0000
count_Withdrawal_Wallet_f       0.000  0.0000   0.0000       8.0000
count_Withdrawal_CreditCard_s   0.000  0.0000   0.0000      12.0000
count_Withdrawal_CreditCard_f   0.000  0.0000   0.0000       5.0000
count_Deposit_Bank_f            0.000  0.0000   0.0000       1.0000
deposits_s                      0.000  0.0000   9.7150   22410.8214
withdrawals_s                   0.000  0.0000   0.0000   59558.6600
wlt_agg                       -12.135 -2.2324   0.3113  106609.3690

                                   kurtosis           var        skew
sum_n_SB                       45413.435016  6.626782e+04  151.259590
sum_p_SB                       27180.812652  8.714265e+04  130.582073
sum_n_GOC                      41901.373433  1.250133e+06  137.133188
sum_p_GOC                      37736.764597  1.265048e+06  130.567348
sum_p_O                        19282.955684  1.097611e+03  112.783953
sum_n_O                         2733.844496  2.651734e+01   42.486386
sum_Deposit_Wallet_s            8238.725545  1.144290e+04   60.032050
sum_Deposit_CreditCard_s       12497.314670  5.339922e+03   81.400544
sum_Deposit_CreditCard_f        9680.133610  3.371047e+03   82.731882
sum_Deposit_Wallet_f          159714.563411  2.636895e+04  346.382616
sum_Withdrawal_BankPayout_f    60087.668763  7.512405e+03  198.924299
sum_Withdrawal_BankPayout_s    28619.509828  1.735322e+04  117.467990
sum_Withdrawal_Wallet_s        74656.215264  2.611249e+04  232.873921
sum_Withdrawal_Wallet_f        51770.129195  3.494725e+03  197.010303
sum_Withdrawal_CreditCard_s    27913.310364  2.187790e+03  135.630607
sum_Withdrawal_CreditCard_f    54750.372741  3.630880e+02  206.737622
sum_Deposit_Bank_f            431092.405334  4.984174e-02  646.071396
count_Deposit_Wallet_s           116.721795  6.413577e-01    6.298530
count_Deposit_CreditCard_s       217.588560  3.022679e-01   10.879844
count_Deposit_CreditCard_f     12066.797870  3.667192e-01   78.398904
count_Deposit_Wallet_f          1203.721620  1.606058e-01   20.197120
count_Withdrawal_BankPayout_f   3572.733530  2.100415e-02   46.512267
count_Withdrawal_BankPayout_s   2075.584780  6.090041e-02   19.215770
count_Withdrawal_Wallet_s        607.898137  3.780341e-02   15.513096
count_Withdrawal_Wallet_f       4363.065798  4.211519e-03   55.189322
count_Withdrawal_CreditCard_s   1943.229408  6.089149e-03   29.818480
count_Withdrawal_CreditCard_f   8057.815129  1.055598e-03   77.287424
count_Deposit_Bank_f          118370.749978  8.447712e-06  344.052685
deposits_s                      5147.110150  1.694257e+04   49.223388
withdrawals_s                  32213.876137  4.457020e+04  140.942144
wlt_agg                        31828.794967  1.078239e+05  118.524142
C:\Users\frans\.conda\envs\python_3_7\lib\site-packages\ipykernel_launcher.py:6: FutureWarning:

Dropping of nuisance columns in DataFrame reductions (with 'numeric_only=None') is deprecated; in a fu
```

```
47  from pycountry_convert import country_alpha2_to_country_name, country_name_to_country_alpha3
    deposit_data = pmts_data.loc[pmts_data['pmtTypeName']=='Deposit']

    country_transactions_status = deposit_data.groupby(["countryCode","status"]).pmtAmount_EUR.agg(['sum'
    country_transactions_status = country_transactions_status.reset_index()
    country_transactions = country_transactions_status.loc[country_transactions_status['status']=='s']
    country_transactions['country_alpha_3'] = country_transactions.countryCode.apply(lambda x: country_nar
    fig = px.choropleth(country_transactions, locations="country_alpha_3",
                        color="sum",
                        hover_name="country_alpha_3", # column to add to hover information
                        color_continuous_scale=px.colors.sequential.Plasma)
    fig.show()
```

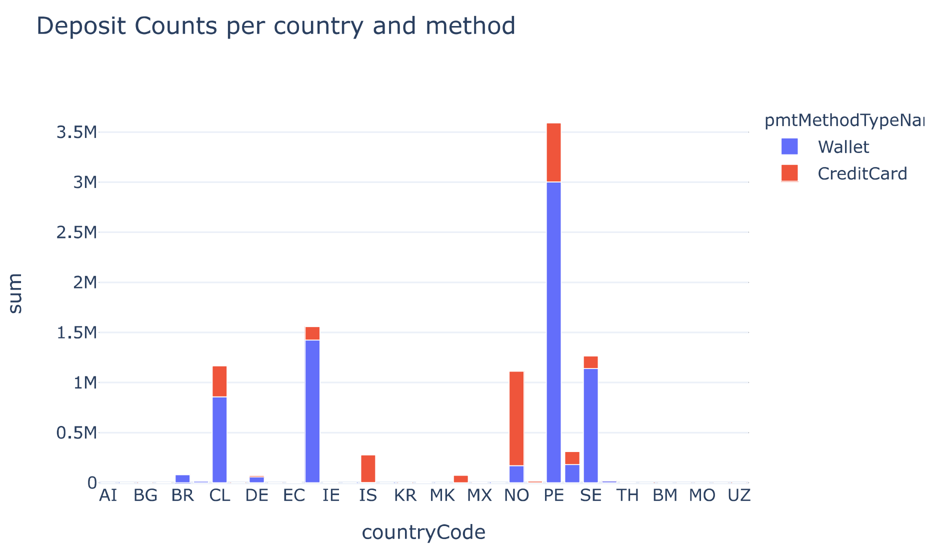


```
48  fig = px.bar(country_transactions_status,x="countryCode", y="sum", color='status', barmode = 'stack',
    fig.show()
```



```
49  country_transactions_method = deposit_data.groupby(["countryCode","status","pmtMethodTypeName"]).pmtAı
```

```
country_transactions_method = country_transactions_method.reset_index()
country_transactions_method = country_transactions_method.loc[country_transactions_method['status']==
fig = px.bar(country_transactions_method,x="countryCode", y="sum", color='pmtMethodTypeName', barmode
fig.show()
```



7.4 Data Segmentation Analysis

Overview

In this notebook, we are performing an analysis on how the data may be segmented for the training of the model.

2

```
import datetime
import math
import glob
import pandas as pd
import numpy as np
from sklearn import preprocessing
import keras
from keras.models import Sequential
from keras.layers import Dense, Flatten, Dropout
from keras.layers import LSTM
from keras import layers
# import lime
# from lime import lime_tabular

np.random.seed(1337)
```

```
Using TensorFlow backend.
```

Prepare the data.

3

```
def get_data(customerGuid, pmt_data, wlt_data):
    # pmt
    if customerGuid != '':
        pmt_data =  pmt_data.loc[pmt_data['customerGuid']==customerGuid]
    pmt_data['created_timesegments'] = pd.to_datetime(pmt_data['pmtCreatedDate']).dt.date
    pmt_data['status'] = np.where(np.logical_or(pmt_data['sk_pmtStatus'] == 3, pmt_data['sk_pmtStatus
    count_transactions = pmt_data.groupby(["customerGuid", "countryCode","pmtTypeName", "created_time
    sum_transactions = pmt_data.groupby(["customerGuid", "countryCode","pmtTypeName", "created_timese
    pmt_aggr = pmt_data.groupby(["customerGuid", "countryCode","pmtTypeName", "created_timesegments",
    pmt_aggr = pmt_aggr.reset_index()
    pmt_pivot = pmt_aggr.pivot(index=["customerGuid", "countryCode", "created_timesegments"], columns
    pmt_pivot = pmt_pivot.fillna(0)
    pmt_pivot.columns = ['_'.join(str(s).strip().strip() for s in col if s) for col in pmt_pivot.colu
    final_pmts =  pmt_pivot.reset_index()
    # final_pmts = final_pmts.drop(['count_Withdrawal_f','count_Withdrawal_s', 'sum_Withdrawal_f'], a
    # wlt
    wlt_data = wlt_data[(wlt_data[['amount_EUR']] != 0).all(axis=1)]
    wlt_data['created_timesegments'] = pd.to_datetime(wlt_data['transactionTimeGMT']).dt.date
    wlt_data.loc[(wlt_data['sk_provider'] == 8, 'product')] = 'SB'
    wlt_data.loc[(wlt_data['sk_provider'] == -1, 'product')] = 'O'
    wlt_data.loc[(np.logical_and(wlt_data['sk_provider'] != 8, wlt_data['sk_provider'] != -1), 'produ

    wlt_aggr = wlt_data.groupby(["customerGuid", "countryCode", "created_timesegments","product"])['a
    wlt_aggr = wlt_aggr.reset_index()
    wlt_pivot = wlt_aggr.pivot(index=["customerGuid", "countryCode", "created_timesegments"], columns
    wlt_pivot = wlt_pivot.fillna(0)
    wlt_pivot.columns = ['_'.join(str(s).strip().strip().strip() for s in col if s) for col in wlt_pi
    final_wlt =  wlt_pivot.reset_index()
    final_wlt.head()
    joined_data = pd.merge(final_wlt,final_pmts,on=['customerGuid','countryCode','created_timesegment
    joined_data.fillna(0, inplace=True)
    return(joined_data)
```

8

```
import glob
```

```
pmt_data1 = pd.read_csv('../Data/pmt_fulldata_masked.csv')

pmt_data1.head()
```

8

	customerGuid	countryCode	pmtCreatedDate	pmtStatusChangeDate	pmtAmount_EUR
0	F859D53E-####-####-####-2C546B6B866B	PE	2022-02-01 23:59:58.643	2022-02-02 00:03:23.110	4.5808
1	1CDA0F4A-####-####-####-7D678FF0B41B	PE	2022-02-01 23:59:57.910	2022-02-02 00:03:12.877	50.3893
2	7088505A-####-####-####-A942DE107D00	PE	2022-02-01 23:59:57.800	2022-02-03 12:00:13.153	3.4356
3	7ACBCBF0-####-####-####-718E2D63387F	PE	2022-02-01 23:59:57.613	2022-02-02 00:02:14.830	45.8084
4	B55763AB-####-####-####-134189DADF88	PE	2022-02-01 23:59:57.067	2022-02-02 00:01:17.033	20.6138

9
```
def create_timeseries(processed_data):
    dates = pd.date_range(start=min(processed_data['created_timesegments']), end=max(processed_data['
    dates = pd.DataFrame(dates)
    dates.columns=['created_timesegments']
    customerGuids = pd.DataFrame(processed_data['customerGuid'].unique())
    customerGuids.columns=['customerGuid']
    dates['key'] = 1
    customerGuids['key'] = 1
    date_guids = pd.merge(customerGuids, dates, on='key').drop('key', axis=1)
    processed_data['created_timesegments'] = processed_data['created_timesegments'].astype('datetime6
    date_guids['created_timesegments'] = date_guids['created_timesegments'].astype('datetime64')

    time_series = pd.merge(date_guids, processed_data, on=['created_timesegments','customerGuid'], how
    time_series = time_series.fillna(0)

    time_series2 = time_series.set_index(['created_timesegments'])
    rolling_window = 1
    shift_1 = time_series2.shift(1*rolling_window, freq='D')
    shift_1 = shift_1.reset_index()
    shift_1 = shift_1.add_suffix('-'+str(rolling_window))
    time_series2 = time_series2.reset_index()
    time_series2.set_index(['customerGuid', 'created_timesegments'], inplace=True)
    shift_1.set_index(['customerGuid-1', 'created_timesegments-1'], inplace=True)
    joined = pd.concat([time_series2, shift_1],  axis=1)
    joined = joined.iloc[rolling_window:]
    joined = joined.reset_index()
    joined = joined.rename(columns={"level_0": "customerGuid", "level_1": "created_timesegments"})
    joined = joined.drop(['countryCode-1'], axis=1)
    joined['created_timesegments'] = joined['created_timesegments'].dt.strftime("%Y%m%d").astype(int)
    joined['customerGuid_int'] = joined.customerGuid.astype('category').cat.codes
    joined['countryCode_int'] = joined.countryCode.astype('category').cat.codes
    customerGuids = joined[['customerGuid','customerGuid_int']].groupby(['customerGuid','customerGuid_
    customerGuids.to_csv('customerGuid_keys.csv')
    countryCodes = joined[['countryCode','countryCode_int']].groupby(['countryCode','countryCode_int'
    countryCodes.to_csv('country_keys.csv')
    joined = joined.drop(['customerGuid', 'countryCode'], axis=1)
    joined = joined.reset_index(drop=True)
    joined_final = joined.drop(['sum_GOC', 'sum_O', 'sum_SB', 'sum_Deposit_f', 'sum_Withdrawal_s', 'c
    return(joined_final)
```

10
```
wlt_data1 = pd.read_csv('../Data/wlt_fulldata_masked.csv')

wlt_data1.head()
```

10

	customerGuid	transactionTimeGMT	countryCode	amount_EUR	sk_provider	wltTra
0	B7849AC7-####-####-####-4B2111CBEFE4	2021-03-11	PE	37.0185	8	Win
1	B7849AC7-####-####-####-4B2111CBEFE4	2021-03-10	PE	1.2633	9	Win
2	B7849AC7-####-####-####-4B2111CBEFE4	2021-03-07	PE	-5.8026	8	Bet
3	B7849AC7-####-####-####-4B2111CBEFE4	2021-03-06	PE	-2.3313	9	Bet
4	B7849AC7-####-####-####-4B2111CBEFE4	2021-03-05	PE	-2.3022	8	Bet

11
```
processed_data = get_data('', pmt_data1, wlt_data1)
processed_data.head()
```

```
C:\Users\frans\.conda\envs\python_3_7\lib\site-packages\ipykernel_launcher.py:18: SettingWithCopyWarn
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexin
C:\Users\frans\.conda\envs\python_3_7\lib\site-packages\pandas\core\indexing.py:1684: SettingWithCopy
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexin
  self.obj[key] = infer_fill_value(value)
C:\Users\frans\.conda\envs\python_3_7\lib\site-packages\pandas\core\indexing.py:1817: SettingWithCopy
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexin
  self._setitem_single_column(loc, value, pi)
```

11

	customerGuid	countryCode	created_timesegments	sum_GOC	sum_O	sum_SB
0	00002912-####-####-####-0B024F0A561D	PE	2021-01-01	0.0	0.0	-113.3500
1	00002912-####-####-####-0B024F0A561D	PE	2021-01-02	0.0	0.0	341.8270

	customerGuid	countryCode	created_timesegments	sum_GOC	sum_O	sum_SB
2	00002912-####-####-####-0B024F0A561D	PE	2021-01-03	0.0	0.0	-208.4588
3	00002912-####-####-####-0B024F0A561D	PE	2021-01-04	0.0	0.0	-608.4368
4	00002912-####-####-####-0B024F0A561D	PE	2021-01-05	0.0	0.0	-334.8962

12
```
processed_data.tail()
```

12

	customerGuid	countryCode	created_timesegments	sum_GOC	sum_O	sum_S
7643398	FFF94C21-####-####-####-005056B229B5	PE	2021-05-05	0.0	0.0	0.0
7643399	FFF94C21-####-####-####-005056B229B5	PE	2021-05-10	0.0	0.0	0.0
7643400	FFF94C21-####-####-####-005056B229B5	PE	2021-06-01	0.0	0.0	0.0
7643401	FFF94C21-####-####-####-005056B229B5	PE	2021-08-11	0.0	0.0	0.0
7643402	FFFB3030-####-####-####-D41C0BA9FE19	PE	2021-09-07	0.0	0.0	0.0

Plot of how much dates there is any data for each customer

13
```
dates_cnt = processed_data.groupby(['customerGuid'])['created_timesegments'].count()
dates_cnt = pd.DataFrame(dates_cnt)
dates_cnt = dates_cnt.reset_index()

dates_cnt.columns = ['customerGuid', 'count_dates']

dates_cnt.head()
```

13

	customerGuid	count_dates
0	00002912-####-####-####-0B024F0A561D	152
1	0000483B-####-####-####-005056B229B5	64
2	00013F92-####-####-####-005056B20345	25

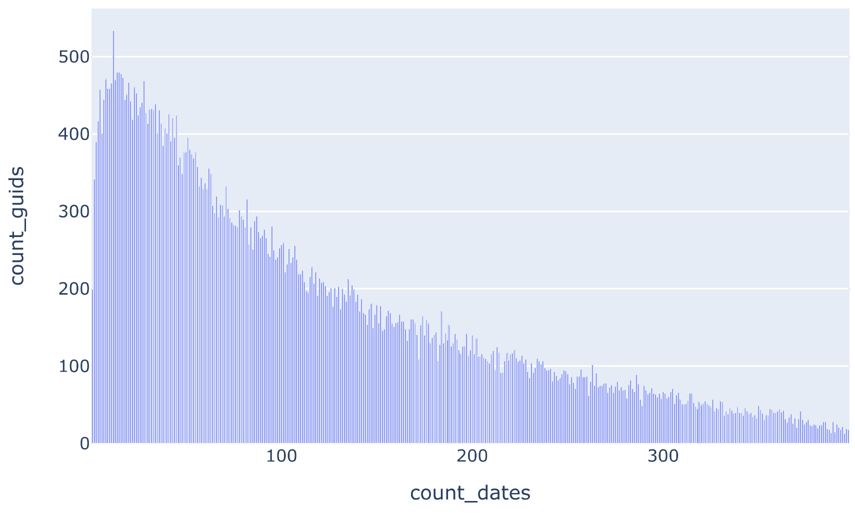
	customerGuid	count_dates
3	0006A3AB-####-####-####-7C3194B26106	340
4	00080818-####-####-####-9887BC250F64	33

14
```
dates_cnt_aggr = dates_cnt.groupby(['count_dates'])['customerGuid'].count()
dates_cnt_aggr = pd.DataFrame(dates_cnt_aggr)
dates_cnt_aggr = dates_cnt_aggr.reset_index()

dates_cnt_aggr.columns = ['count_dates', 'count_guids']

dates_cnt_aggr.tail()
```

14

	count_dates	count_guids
392	393	18
393	394	22
394	395	13
395	396	19
396	397	18

15
```
import plotly.express as px
fig = px.bar(dates_cnt_aggr, x='count_dates', y='count_guids')
fig.show()
```



Plot of how much dates there is deposit data for each customer

17
```
deposit_data = processed_data[processed_data['sum_Deposit_s']!=0]

dep_dates_cnt = deposit_data.groupby(['customerGuid'])['created_timesegments'].count()
```

18
```
dep_dates_cnt = pd.DataFrame(dep_dates_cnt)
dep_dates_cnt = dep_dates_cnt.reset_index()

dep_dates_cnt.columns = ['customerGuid', 'count_dates']
dep_dates_cnt.head()
```

18

	customerGuid	count_dates
0	00002912-####-####-####-0B024F0A561D	53
1	0000483B-####-####-####-005056B229B5	38
2	00013F92-####-####-####-005056B20345	7
3	0006A3AB-####-####-####-7C3194B26106	163
4	00080818-####-####-####-9887BC250F64	18

19
```
dep_dates_cnt[dep_dates_cnt['count_dates']==391]
```

19

	customerGuid	count_dates
10852	238C9656-####-####-####-E6940BEF58B3	391
24309	58B22DE3-####-####-####-951A4F994834	391

First column represents the number of dates whilst the second represents the number of customers who made successful deposits on that many successful dates.

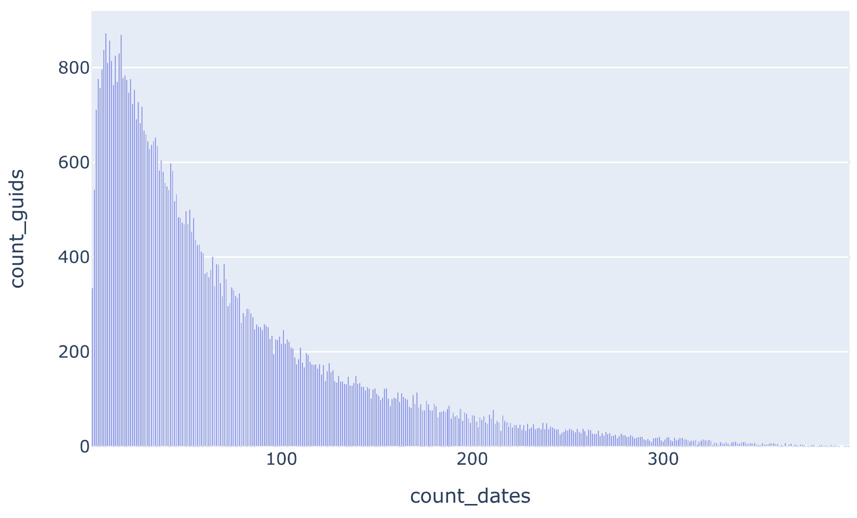
20
```
dep_dates_cnt_aggr = dep_dates_cnt.groupby(['count_dates'])['customerGuid'].count()
dep_dates_cnt_aggr = pd.DataFrame(dep_dates_cnt_aggr)
dep_dates_cnt_aggr = dep_dates_cnt_aggr.reset_index()

dep_dates_cnt_aggr.columns = ['count_dates', 'count_guids']

dep_dates_cnt_aggr.tail()
```

20

	count_dates	count_guids
388	391	2
389	392	1
390	395	1
391	396	2
392	397	1

23
```
fig = px.bar(dep_dates_cnt_aggr, x='count_dates', y='count_guids')
```

```
fig.show()
```



Distribution of any data across months.

28
```
processed_data['created_timesegments'] = processed_data['created_timesegments'].astype('datetime64[ns
processed_data['month-year'] = processed_data['created_timesegments'].dt.month.astype(str)+'-'+proces

mths_cnt = processed_data.groupby(['customerGuid'])[['month-year']].nunique()
mths_cnt = mths_cnt.reset_index()
mths_cnt.head()
```

28

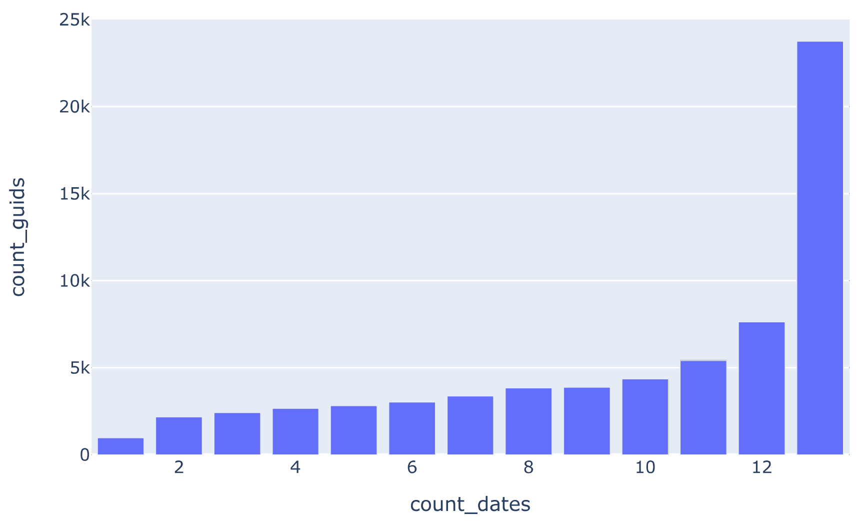
	customerGuid	month-year
0	00002912-####-####-####-0B024F0A561D	9
1	0000483B-####-####-####-005056B229B5	14
2	00013F92-####-####-####-005056B20345	4
3	0006A3AB-####-####-####-7C3194B26106	14
4	00080818-####-####-####-9887BC250F64	7

98
```
mths_cnt_aggr = mths_cnt.groupby(['month-year'])['customerGuid'].count()
mths_cnt_aggr = pd.DataFrame(mths_cnt_aggr)
mths_cnt_aggr = mths_cnt_aggr.reset_index()

mths_cnt_aggr.columns = ['count_dates', 'count_guids']

mths_cnt_aggr.tail()
```

98

	count_dates	count_guids
8	9	3877
9	10	4361
10	11	5418
11	12	7635
12	13	23749

```
99  fig = px.bar(mths_cnt_aggr, x='count_dates', y='count_guids')
    fig.show()
```



```
30  deposit_data['month-year'] = processed_data['created_timesegments'].dt.month.astype(str)+'-'+processe

    dep_mths_cnt = deposit_data.groupby(['customerGuid'])[['month-year']].nunique()
    dep_mths_cnt = mths_cnt.reset_index()
    dep_mths_cnt.head()

    C:\Users\frans\.conda\envs\python_3_7\lib\site-packages\ipykernel_launcher.py:1: SettingWithCopyWarnir


    A value is trying to be set on a copy of a slice from a DataFrame.
    Try using .loc[row_indexer,col_indexer] = value instead

    See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing
```

30

	index	customerGuid	month-year

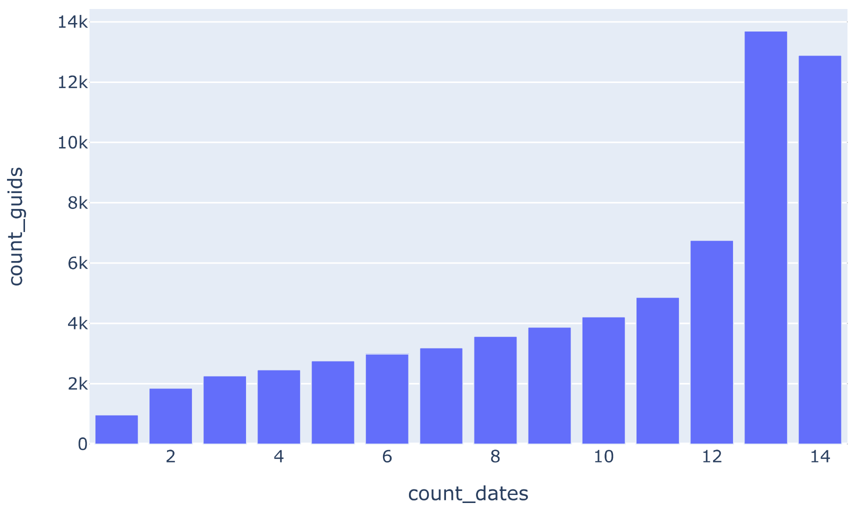
	index	customerGuid	month-year
0	0	00002912-####-####-####-0B024F0A561D	9
1	1	0000483B-####-####-####-005056B229B5	14
2	2	00013F92-####-####-####-005056B20345	4
3	3	0006A3AB-####-####-####-7C3194B26106	14
4	4	00080818-####-####-####-9887BC250F64	7

31
```
dep_mths_cnt_aggr = dep_mths_cnt.groupby(['month-year'])['customerGuid'].count()
dep_mths_cnt_aggr = pd.DataFrame(dep_mths_cnt_aggr)
dep_mths_cnt_aggr = dep_mths_cnt_aggr.reset_index()

dep_mths_cnt_aggr.columns = ['count_dates', 'count_guids']

dep_mths_cnt_aggr.tail()
```

31

	count_dates	count_guids
9	10	4220
10	11	4865
11	12	6750
12	13	13695
13	14	12891

33
```
fig = px.bar(dep_mths_cnt_aggr, x='count_dates', y='count_guids')
fig.show()
```



Conclusion

We will split the data in the following way :

1. 1-100 dates, 101-200 dates, 200-300 dates, 300 + dates #