CCT College Dublin

Assessment Cover Page

Module Title:	Data Visualization Techniques, Machine Learning for Business
Assessment Title:	CA2
Lecturer Name:	Dr. Muhammad Iqbal, James Garza
Student Full Name:	Fabiane dos Santos Teixeira (2021225) Nuno Alfredo Ribeiro Teixeira de Almeida (2021310) Valesca Soledad Bravo Bravo (2021235) Thiago Medeiros de Souza (2019410)
Student Number:	2021225 2021310 2021235 2019410
Assessment Due Date:	29th May 2022
Date of Submission:	29th May 2022

Declaration

By submitting this assessment, I confirm that I have read the CCT policy on Academic Misconduct and understand the implications of submitting work that is not my own or does not appropriately reference material taken from a third party or other source. I declare it to be my own work and that all material from third parties has been appropriately referenced. I further confirm that this work has not previously been submitted for assessment by myself or someone else in CCT College Dublin or any other higher education institution.

Table of Contents

1. Introduction

The “machine learning prediction world” is a fantastic adventure for the data analyst. After undergoing experimentations in different fields, we can see that, by following the “model agenda”, we can compile precious pieces of information for all types of audiences.

As important as performing the machine learning wisely is, communicating its results is also crucial. In doing so, of course long reports with a step-by-step guide of your analysis is a tempting thing to be done, as we all want to show how arduous the process is. However, we must agree that it’s not the most friendly way to communicate the main insights, considering that it also takes a long time to be read, understood and assimilated. Instead, “an image talks more the thousand words”, as they say. And plots, graphs, visual results are definitely the key of a data analysis communication.

In this research, we will experiment few machine learning models and play with different visualizations, in order to analyse specific behaviours: Time-Series, Content and Collaborative filtering and Market Basket.

When we talk about time-series, we automatically think about a set of random variables ordered according to the variable time. We usually analyse a time-series dataset in order to identify specifics behaviour in that data, its trend components, seasonality and to predict its future values.

When experimenting recommendation systems for online retail business in machine learning, we have it as an important ally to help with large amounts of information in the digital market. That maxima: “Everything is data” includes our profile as consumers, so that strategies to lead us to a tailored consuming must be analytically developed, and then attract us to a personalized consuming world. Such strategies are developed by filtering all this data.

Last but not least, market basket is also another ally for the online market. Using the right algorithms, throughout this technique it is possible to understand the purchase behaviour. “If this, then that” is a very clever strategy not only for retailers but for fast paced routine that we are all involved in.

When developing all these analyses, we will then answer analytical questions that will be accompanied by its visualization as well as an understanding of important statistical aspects behind it. Each topic of this project will be used to lead to the juice of the answers.

2. Time Series

A time series analysis explores a time series data by applying statistical methods. In doing that, we are able to extract meaningful statistics, patterns and other characteristics of the data. And then, in order to visualize its first – and main – insights, line charts are used to plot the data behaviour throughout the time.

Let's see how it happens when performed in a real-world data set, which brings forward information about the eurozone. Before we jump to the main findings of our analysis and, consequently, answer our questions, let's summarize an explanation for our data.

The eurozone is a monetary union of 19 member states of the European Union that have adopted the euro as their primary currency. Because of its importance, it is pretty common to measure its correspondence in many other countries around the world. Our dataset shows exactly this: information about the current exchange rate EURO (EUR) to 38 other currencies from different countries through the years, with object types.

Some preparation has been done such as setting the column Date as index, converting columns' information to numeric type, cleaning the characters in the name of each currency and deleting the total of 62 NaN values, as they represented only 1% of the total data.

Once all this prep was done, we then decided to work with the exchange rate EURO (EUR) to US DOLLAR (USD) to build up our analysis and then we can plot the very first visualization in a time-series analysis: The time series decomposition which is a representation of patterns automatically decomposed:

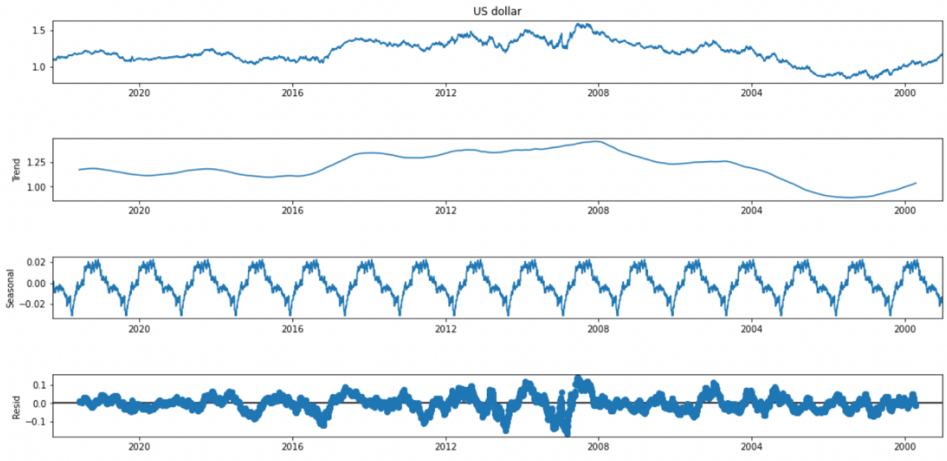

Figure 1 Data Decomposition. Source: Jupyter Notebook by the authors.

According to Brownlee, J. (2017), these 4 components plotted above are defined as follows:

Level: The average value in the series.

Trend: The increasing or decreasing value in the series.
Seasonality: The repeating short-term cycle in the series.
Noise: The random variation in the series.

The def function is created based on a statistical function called seasonal_decompose in the statsmodels library, which conveniently implements whether an additive or multiplicative combination of the base level, trend, seasonal index and the residual term for the series. That being said, the multiplicative decomposition plotted above shows us some pattern and that's now what we have to dig more into.

When analysing time series behaviour, if the data shows particular behaviour over time, it is expected that it will follow a similar behaviour in the future. This particularity can be related to its Stationarity, which is when its statistical properties such as mean, variance remain constant over time (Mesidor, F. 2019). Before any modelling, it is necessary to check if our data is stationary. As most time series models work on the assumption that the time series are stationary, it is important to validate that hypothesis. That drives us to our first question:

1) What is the purpose of The Augmented Dickey-Fuller test in time series?

Simply said, the Dickey Fuller Test is a statistical test for testing stationarity. This test will give results in a hypothesis test, with null and alternative hypotheses. As a result, we will have a p-value from which we will need to make inferences about the time series, whether it is stationary or not (Verma, Y. 2021).

Because it is a statistical significance test, the result is computed as a test statistic and p-values get reported. It is from the test statistic and the p-value, that you can make an inference as to whether a given series is stationary or not. (Prabhakaran, S. 2019). In doing that, the ADF test will then analyse the presence of a unit root ($\alpha=1$) in the time series as the presence of a unit root means the time series is non-stationary.

In mathematical terms, the null hypothesis assumes the presence of unit root, that is $\alpha=1$, the p-value obtained should be less than the significance level (say 0.05) in order to reject the null hypothesis. In python, we created a def function ADFuller from the statsmodels library and set the autolag='AIC' in order to get the number of lags that yields the lowest AIC.

```
ADF Statistic: -1.827750
p-value: 0.366774
Critical Values:
        1%: -3.431450071438642 - The data is not stationary with 99% confidence
        5%: -2.8620261296599536 - The data is not stationary with 95% confidence
        10%: -2.5670287644760372 - The data is not stationary with 90% confidence
```

Figure 2 ADF result. Source: Jupyter Notebook by the authors.

As we can see above, the p-value obtained is greater than the significance level of 0.05 and the ADF statistic is higher than any of the critical values. Clearly, there is no reason to reject the null hypothesis. So, the time series is in fact non-stationary.

a) Apply an appropriate ARIMA model to the chosen data. Check for the model adequacy.

Simply said, the AutoRegressive Integrated Moving average (ARIMA) predicts future values based on past values. In ARIMA time series forecasting, the first step is to determine the number of differencing required to make the series stationary.

We have already explored the data decomposition above, performed the ADF test and turned the dataset into stationary by detrending with rolling mean of 30 days. Now, the statsmodels library provides the capability to fit an ARIMA model, and then we just have to pass in the p, d, and q parameters. In doing that, we decided to evaluate combinations of p, d and q values for an ARIMA model by calculating the lowest AIC (Akaine information criterion) and define the function for the autorima. Let's visualize this process:

```
Performing stepwise search to minimize aic
 ARIMA(0,1,0)(0,0,0)[0] intercept   : AIC=-62745.545, Time=0.93 sec
 ARIMA(1,1,0)(0,0,0)[0] intercept   : AIC=-62743.571, Time=1.01 sec
 ARIMA(0,1,1)(0,0,0)[0] intercept   : AIC=-62743.571, Time=0.81 sec
 ARIMA(0,1,0)(0,0,0)[0]             : AIC=-62747.517, Time=0.46 sec
 ARIMA(1,1,1)(0,0,0)[0] intercept   : AIC=-62741.544, Time=2.02 sec

Best model:  ARIMA(0,1,0)(0,0,0)[0]
Total fit time: 5.282 seconds
```

Figure 3 AutoARIMA. Source: Jupyter Notebook by the authors.

Based on that, we fit an ARIMA(0,1,0) model. This sets the lag value to 0 for autoregression, uses a difference order of 1 to make the time series stationary, and uses a moving average model of 0. Train and test defined (0.85) and we can now summarize the coefficient values used as well as the skill of the fit on the in-sample observations:

Dep. Variable:	y	**No. Observations:**	8495
Model:	ARIMA(1, 0, 0)	**Log Likelihood**	31373.307
Date:	Sun, 29 May 2022	**AIC**	-62740.614
Time:	11:24:04	**BIC**	-62719.473
Sample:	0	**HQIC**	-62733.399
	- 8495		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
const	1.1980	0.097	12.352	0.000	1.008	1.388
ar.L1	0.9993	0.000	2655.946	0.000	0.999	1.000
sigma2	3.625e-05	2.6e-07	139.359	0.000	3.57e-05	3.68e-05

Ljung-Box (L1) (Q):	0.02	**Jarque-Bera (JB):**	19045.45
Prob(Q):	0.90	**Prob(JB):**	0.00
Heteroskedasticity (H):	0.66	**Skew:**	-0.12
Prob(H) (two-sided):	0.00	**Kurtosis:**	10.33

Figure 4 Coefficient values. Source: Jupyter Notebook by the authors.

Before we get the prediction, let's check the accuracy metrics for our time series forecast:

Model	r2_score	mean_absolute_error	mean_squared_error	root_mean_squared_error	mean_absolute_percentage_error
ARIMA Prediction	0.990847	0.002612	0.000017	0.004104	0.227072

Figure 5 Prediction Scores. Source: Jupyter Notebook by the authors.

Simple yet relevant analysis that can be validated via R2 score. In a nutshell, the R2 score varies between 0 and 100. If it is 100%, the two variables are perfectly correlated, with no variance at all, whereas a low value would show a low level of correlation (Rowe, W. 2018). So we can see that our R2 score is 0.99.

b) Make one-step-ahead forecasts of the last 10 observations. Determine the forecast errors.

As a result of doing this, we created a range of data considering the last 10 observations, we added residues and fitted values and can see the result below:

Figure 6 Arima Forecast. Source: Jupyter Notebook by the authors.

The function Arima() will fit a regression model with ARIMA errors. The order argument specifies the order of the ARIMA error model. If differencing is specified, then the differencing is applied to all variables in the regression model before the model is estimated:

Figure 7 ARIMA diffval. Source: Jupyter Notebook by the authors.

c) Time series plot / ACF/ PACF: Is there significant autocorrelation in the time series?

To get to that answer and as mentioned in the step above, we have already stationarized the data. Let's dig into the correlation step of our analysis.

Correlation in a time series context is calculated based on previous time steps called lags. Because the correlation of the time series observations is calculated with values of the same series at previous times, this is called a serial correlation, or an autocorrelation. (Brownlee, J. 2017). Whereas partial autocorrelation is the amount of correlation between a variable and a lag of itself that is not explained by correlations at all lower-order-lags.

When then visualize it by using the plot_acf() function from the statsmodels library and limit the number of lags on the x-axis to 60. The ACF and PACF plots should be considered together to define the process:

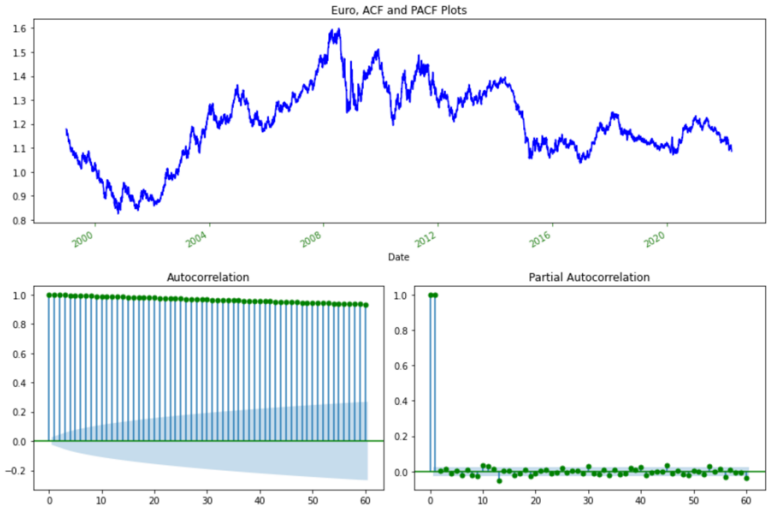

Figure 8 ACF/ PACF. Source: Jupyter Notebook by the authors.

The autocorrelations are significant for a large number of lags, although we should also consider that the autocorrelations at lags 2 and above are merely due to the propagation of the autocorrelation at lag 1. This is confirmed by the PACF plot as the PACF plot has a significant spike only at lag 1, meaning that all the higher-order autocorrelations are effectively explained by the lag-1 autocorrelation.

d) Visualizing the density of the data: what is it worth to be highlighted?

To answer this question, we will use a special library called Plotly that not only helps us visualize the data but also creates highly interactive and visually appealing plots. First one is a 2D density plot or 2D histogram, an extension of the well-known histogram. It shows the distribution of values in a data set across the range of two quantitative variables (Sharma, H. 2020):

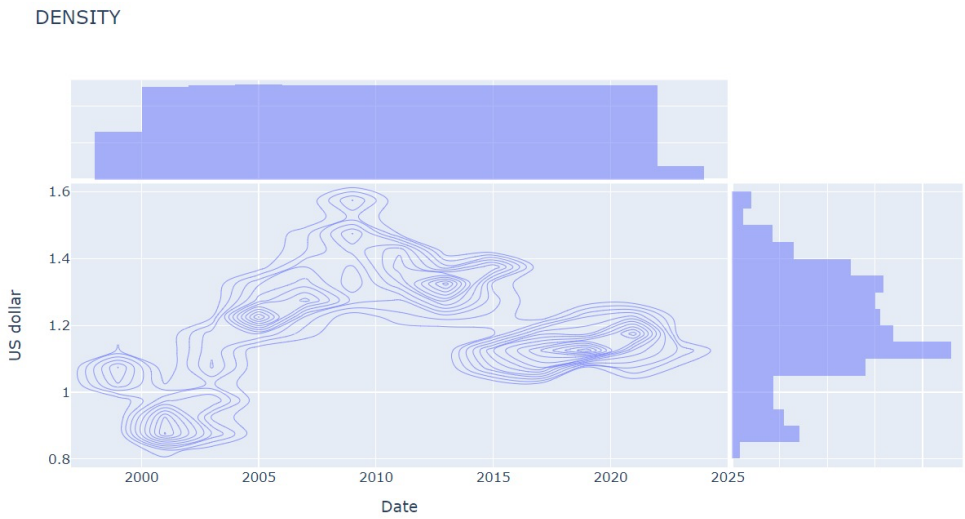

Figure 9 Density - Plotly. Source: Jupyter Notebook by the authors.

In the above plot, the Plotly Express function density_contour() was used to produce density contours of the variable US Dollar through the years, that means the distribution of this variable. Along with that, marginal plots were added to visualize the 1-dimensional distributions of the variables Euro/ US Dollar and Date (Year). Here we use a marginal histogram.

Based on that, we can see high quotations before the year 2010, a more dense behaviour before 2020 and a volatility after 2020, which is the period equivalent to the start of the COVID-19 pandemic. Speaking about that, let's have a closer look at the year 2020:

Figure 10 2020 view - Plotly. Source: Jupyter Notebook by the authors.

That can be easily summarized by the huge impact of the covid-19 pandemic around the world and, as well explained by Ghosh, P. (2021), the weakness of the dollar began after the pandemic risk was perceived to be reduced by the scale of the U.S. response and anticipation of vaccines of greater efficacy to counter the spread of Covid-19.

e) Showing different times of the series:

When exploring different windows of our data, we decided to highlight 9 different milestones in the world that had an effect on the Dollar/Euro behaviour by defining the specific window for each period.

Using the Matplotlib library, we set the figure by specifying the number of plots and size of each one, and selecting different colours for ticks and title. After that, we plot the graph for each period, calling the function created beforehand in order to set the visual parameters. Let's have a look at the results:

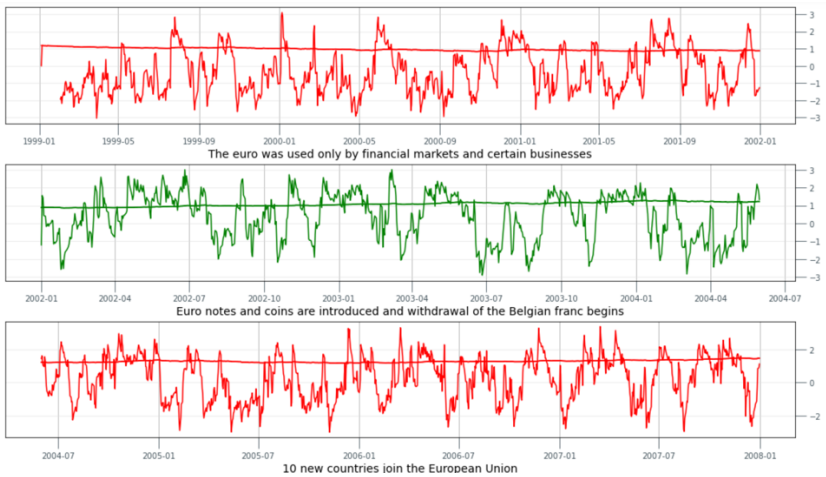

Figure 11 Set of windows 1. Source: Jupyter Notebook by the authors.

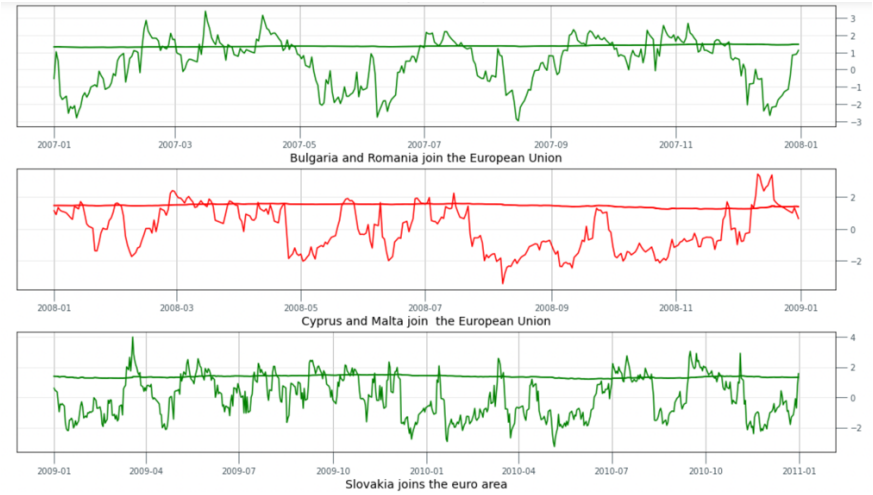

Figure 12 Set of windows 2. Source: Jupyter Notebook by the authors

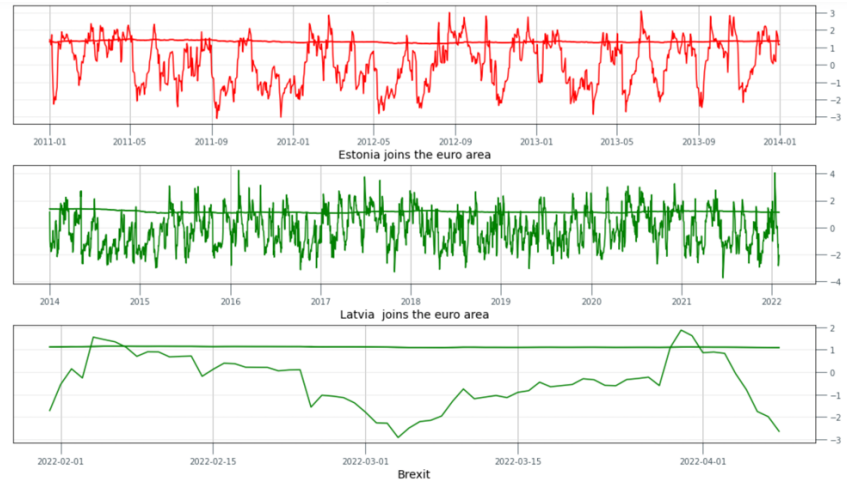

Figure 13 Set of windows 3. Source: Jupyter Notebook by the authors

3. Recommendation System

2) What is the purpose of a recommendation system for online retail business in machine learning?

In a reality where we consume everything online – from utensils to entertainment – we can ask 3 questions: how would it be if we have everything displayed at once? Answer: it is impossible. We have thousands and thousands of consumers and, consequently items to be consumed. 2) How can all of this be wisely directed then? Answer: Consumer tailoring. Same logic when you apply for a job and you have to tailor your cv to match with the requirements and potentially get an interview. 3) Who does it then? Answer: the machine, once you teach it.

That's when Recommendation System comes in handy. Recommendation system is basically a tool that filters information and then recommends the relevant products or services to the users (Muvi, M. 2021). A recommendation engine can be of three types: collaborative

recommendation engine, content based recommendation engine, and hybrid recommendation engine.

Considering that, we use the concept of nodes and edged. Each node is a user and an edge between two users represents that those two users have once read the same book. By using a similar set of data, we will compare collaborative recommendation and content based recommendation engine. We can previously differentiate the emphasis: collaborative filtering emphasizes the users preference and their likes and dislikes similarities, whereas content filtering emphasizes the content features and similarities between features that a user has already seen or bought.

We picked a set of data with information on books and its users, and we explored how these users and books are related to each other based on their ratings. Then we can model this as a network of users and use the recommendation engine to filter the data using different algorithm and recommend the relevant items to the users.

a) Train and test machine learning models for the user-user collaborative filtering:

This algorithm first finds the similarity score between users. Based on this similarity score, it picks out the most similar users and recommends products which these similar users have liked or bought (Ajitsaria, A. 2022)

In our analysis, this algorithm found the similarity between each user based on the ratings they have previously given to different books, having a total of 95513 users who rated the books. The prediction of an item for a user (u) is calculated by computing the weighted sum of the user ratings given by other users to an item (i).

In order to handle the complexity of the algorithm and to avoid picking users that have only registered to the website and have only read a few books, we selected users that have given more than 200 ratings, and then we extracted the books that have received more than 50 ratings from users. Now, to prepare the dataset for the modelling, as there were a huge amount of zeros, and on clustering this would increase the computer power in order to calculate the distance of zero values, we then sparse the matrix and fed it to the model.

We performed the nearest neighbors model to train the dataset, and then we used the algorithm ‘brute’ that will find the distance of every point to every other point. Using as an example the index 17 of the data, we made a prediction and got the suggested books printed based on the rating from users neighbours:

```
for i in range(len(suggestions)):
  print(book_pivot_df.index[suggestions[i]])
```

```
Index(['Degree of Guilt', 'The Reef', 'Red Storm Rising', 'A Patchwork Planet',
       'A Fine Balance'],
      dtype='object', name='title')
```

Figure 14 User-User Recommendation. Source: Jupyter Notebook by the authors

This algorithm is useful when the number of users is less. It is not effective when there are a large number of users as it will take a lot of time to compute the similarity between all user pairs. If we want to dig further into this part, then we explore the item-item collaborative filtering, which is effective when the number of users is more than the items being recommended.

b) Train and test machine learning models for the item-item collaborative filtering

In this algorithm, we find the similarity between each book pair and based on that, we recommend similar books which are liked by the user in the past. Different from the user-user explained above, instead of taking the weighted sum of ratings of "user-neighbours", we take the weighted sum of ratings of "item-neighbours".

In our analysis, we randomly picked two book titles and its descriptions for checking, then we calculated the word count for book description by using the lambda function and had it plotted in a histogram. We then used the text blob function to get and plot the distribution of top part-of-speech tags in the book descriptions.

We then converted the text descriptions into vectors using TF-IDF using Bigram (two-word sequence of words) and found the word frequency, plotting then the top 20 words in the text description. After that, we converted the text descriptions into vectors using TF-IDF using Trigram (three-word sequence of words) and also plotted the top 20.

Now, def functions was created for removing NonAscii characters (encoding and decoding of strings into Unicode), converting into lower case, removing stop words, removing punctuation and removing the html tags, and then compiled.

Finally, a def function has been created for recommending books based on Book title. It takes book title and genre as an input and following these steps: index converted into series, book title converted into vectors, similarity measures calculated based on Cosine similarity, got the pairwise similarity scores and we then have the function ready to show the top 5 recommended book URL and print the images:

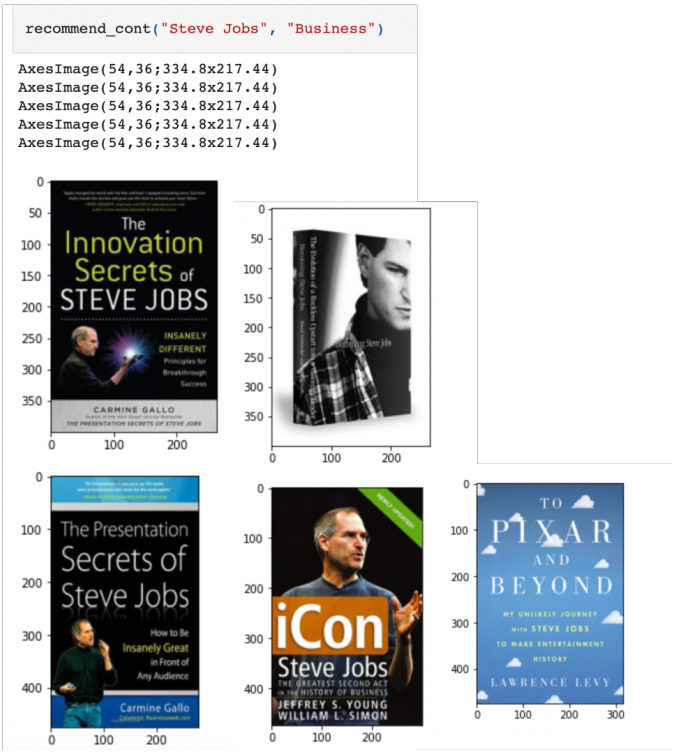

Figure 15 Content-Content Recommendation - Title. Source: Jupyter Notebook by the authors

We can also do a recommendation based on the book description and following the same steps explained above, obviously converting the book description into vectors instead:

Figure 16 Content-Content Recommendation - Description. Source: Jupyter Notebook by the authors

4. Visualization

Tableau is a very effective tool for creating interactive data visualizations very quickly. Let's then use this tool to summarize important aspects of the data.

First aspect to be shown is the average of votes per book, considering the books who got more than 80 votes in order to have a friendly plot. As we can see below, many titles got impressive votes which means that different readers around the world might have the same taste in books:

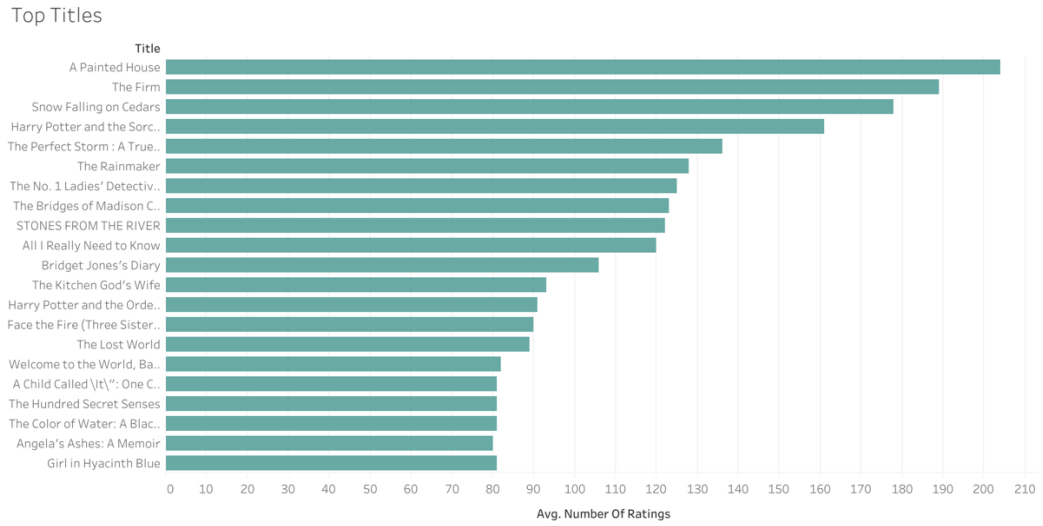

Figure 17 Top Titles. Source: Tableau plot by the authors.

Let's dig deeper into how these votes are distributed in our data and analyse per author. And again: some authors are quite popular amongst readers who also share same taste for narratives:

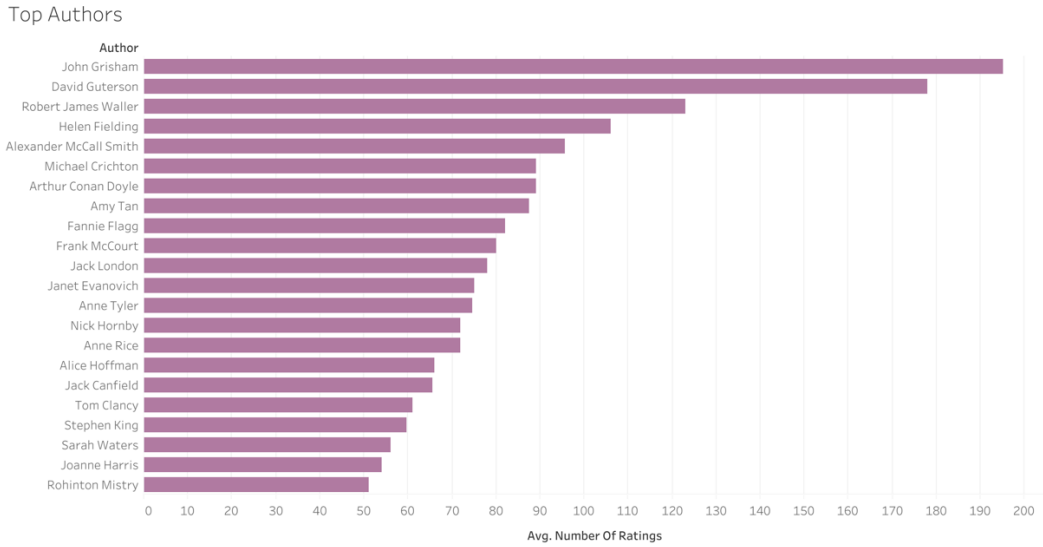

Figure 18 Top Authors. Source: Tableau plot by the authors.

Now, if we consider number of votes (ratings) according to the year when each book is published and plot on the top of that the average of this rating, we will see how heterogeneous the users tastes are. However, still having Year as a parameter, we can see that the amount of votes/rating are still impressive and show clusters of readers for the same genre/author/year/title.

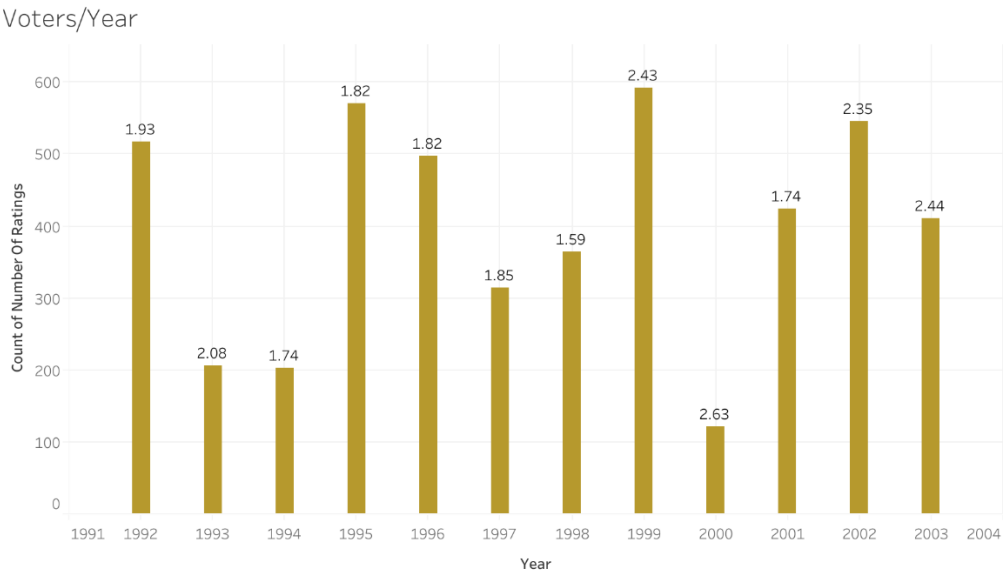

Figure 19 Votes per Year + Average Rating. Source: Tableau plot by the authors.

2.b) Why this dataset is suitable for Machine Learning models in an online retail business

The 3 visualization plotted above lead us to this answer. Therefore, we can conclude that we have enough parameters to develop an analysis in which we can build up a recommendation engine based on features or behaviours of a user given the item's feature as well as considering other users' reactions.

5. Correlation

3.a) Explain the meaning of "correlation" between the variables/ Visualizing it / Colours:

According to Brownlee, J. (2018), correlation can be useful in data analysis and modelling to better understand the relationships between variables. The statistical relationship between two variables is referred to as their correlation.

The result of this correlation could be positive, neutral or negative:

- Positive Correlation: both variables change in the same direction.
- Neutral Correlation: no relationship in the change of the variables.
- Negative Correlation: variables change in opposite directions.

To illustrate that, we carried out a correlation analysis between the variables Reading and Writing, in a dataset where we explore scores for a classroom. We visualize such correlation when we review the generated scatter plot where we can see that the highest scores of both have a bigger correlation. The plot below includes the gender differentiation, plotted in different colours to make its visualization even easier:

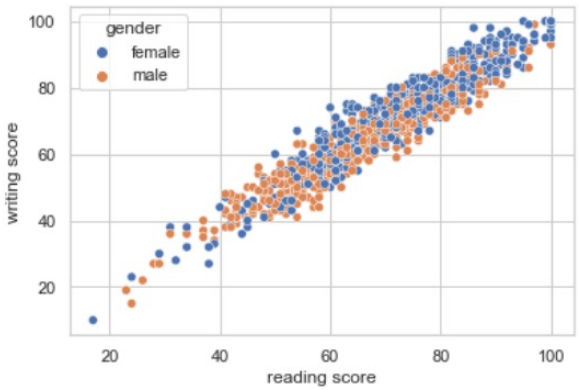

Figure 20 Heatmap. Source: Jupyter Notebook by the authors.

We can also have this visualization in a jointplot and analyse the distribution of each variable:

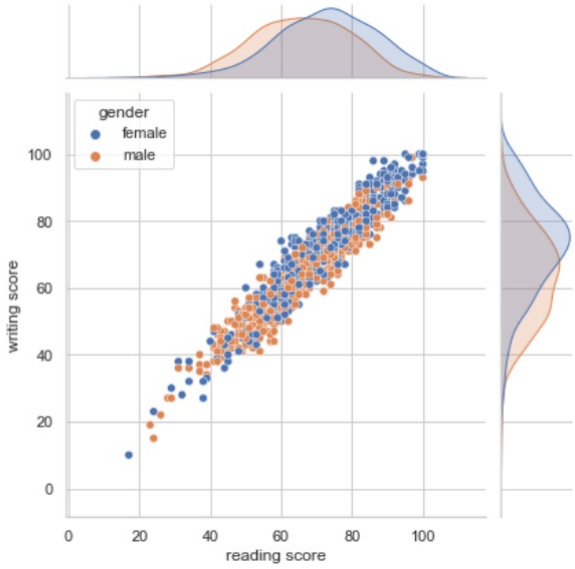

Figure 21 Jointplot. Source: Jupyter Notebook by the authors.

Before we look at calculating some correlation scores, let's have a look at our covariance, which is the linear relationship summarized by two variables: the covariance

between the two main variables is 211.78. We can see that it is positive, suggesting that the variables change in the same direction as we expected.

In order to interpretate this result, we are going to calculate the Pearson's correlation coefficient. The Pearson correlation coefficient correlation coefficient is a number that summarizes the strength of the linear relationship between two data samples (Brownlee, J., 2018) and it's the most common measure of correlation. Based on this, we can see that the two variables are positively correlated and that the correlation is 0.955. This suggests a strong level of correlation.

In relation to a nonlinear relationship, we can calculate the Spearman's correlation coefficient to find out if the relationship is stronger or weaker across the distribution of the variables. Instead of calculating the coefficient using covariance and standard deviations on the samples themselves, these statistics are calculated from the relative rank of values on each sample and it is equal to 0.949.

The library **seaborn** provides a function called **heatmap** that allows us to visualize the correlation between variables. Through this Correlation heatmap we can see a graphical representation of the **correlation matrix** representing correlation between our variables.
The coefficient returns a value between -1 and 1 that represents the limits of correlation from a full negative correlation to a full positive correlation. A value of 0 means no correlation. Where often a value below -0.5 or above 0.5 indicates a notable correlation, and values below those values suggests a less notable correlation.

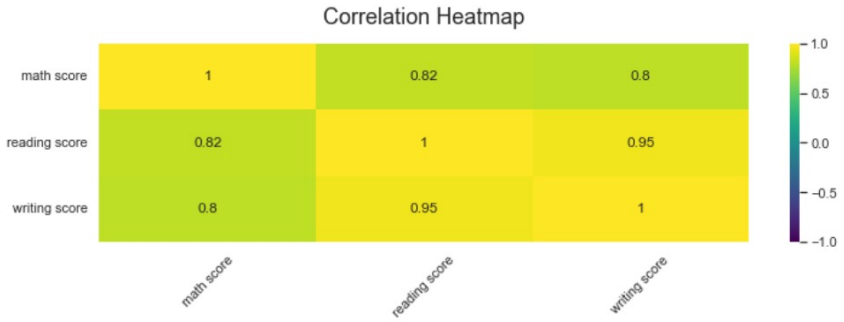

Figure 22 Heatmap. Source: Jupyter Notebook by the authors.

That being the case, let's keep in mind our strong level of correlation 0.95 between the variable Reading score and Writing score.

6. Market Basket

3.b) Perform Market Basket Analysis on the chosen dataset by using Apriori and FP growth algorithms

In performing this analysis we got a dataset that belongs to "The Bread Basket" a bakery located in Edinburgh Kuila, A. (2021). The dataset provides the transaction details of customers who ordered different items from this bakery online during the time period from 26-01-11 to 27-12-03. The dataset has 20507 entries, over 9000 transactions, and 5 columns.

Data has been explored and EDA performed, no null values found and emphasis to the statistical aspects of the data, in which we can see a total of 94 products and the top product transaction is Coffee, followed by Bread and then Tea.

In order to prepare the data to perform both algorithms Apriori and FP growth, we first created a function to put all the items in a list according to the transactions. After that, we used OneHot Enconder to relate the items transactions.

Starting with Apriori, a mining technique for creating Boolean association rules from frequent itemsets. It finds (k + 1)-itemsets from k-itemsets using an iterative level-wise search approach. An example of transactional data consisting of product items purchased at various transactions. (Agrawal R. and Srikant R. 1994). The Apriori Algorithm was used to generate the frequent item sets. When we calculated this frequency, the method Apriori was then used in all items and specified the minimum support as 0.01 out of the total transactions. This value is calculated using the following: Support(A) = Transactions(A) / Total Transactions.

When compiling these results, we can see the itemsets with a minimum support of 1%. Meaning, the column “support” may be used for various downstream activities such as developing marketing strategies or even for offers in buying product combinations.

Next step is called Association Rules, in which we filter for Lift values greater than 1%, which represents how many times the probability of getting B product increased when A product is received. When displaying the rules, we used min_threshold equals to 0.01, which works as a minimal support criterion, so that things with support equal to 0.01 are evaluated for subsequent phases of the algorithms in the very first step:

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
73	(Cake)	(Coffee, Tea)	0.103856	0.049868	0.010037	0.096643	1.937977	0.004858	1.051779
68	(Coffee, Tea)	(Cake)	0.049868	0.103856	0.010037	0.201271	1.937977	0.004858	1.121962
26	(Hot chocolate)	(Cake)	0.058320	0.103856	0.011410	0.195652	1.883874	0.005354	1.114125
27	(Cake)	(Hot chocolate)	0.103856	0.058320	0.011410	0.109868	1.883874	0.005354	1.057910
28	(Tea)	(Cake)	0.142631	0.103856	0.023772	0.166667	1.604781	0.008959	1.075372

Figure 23 Association Rules - Apriori. Source: Jupyter Notebook by the authors.

We then created another rule, associating with more than one item between products:

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
68	(Coffee, Tea)	(Cake)	0.049868	0.103856	0.010037	0.201271	1.937977	0.004858	1.121962
62	(Coffee, Bread)	(Pastry)	0.090016	0.086107	0.011199	0.124413	1.444872	0.003448	1.043749
69	(Coffee, Cake)	(Tea)	0.054728	0.142631	0.010037	0.183398	1.285822	0.002231	1.049923
58	(Coffee, Bread)	(Cake)	0.090016	0.103856	0.010037	0.111502	1.073621	0.000688	1.008606
57	(Bread, Cake)	(Coffee)	0.023349	0.478394	0.010037	0.429864	0.898557	-0.001133	0.914880

Figure 24 Top association rules. Source: Jupyter Notebook by the authors.

As we can see above in that analysis using Metric: "Lift" according to Sivek, S. (2020):

- Lift, greater than one indicates that items are positively correlated.
- Lift Less than one indicates that items are negatively correlated
- If Item A and B are independent the lift has a value of one.
- And according to our data, if someone has already added to his basket (Tea, Coffee) or (Coffee, Bread) then the item which is 93% more likely to be added is Cake.

Moving now to the FP-growth algorithm, it is defined as a technique for mining frequent itemsets without requiring a time-consuming candidate generation procedure. It uses a divide-and-conquer strategy to condense the frequent items into a Frequent Pattern Tree (FP-Tree) that maintains the frequent items' association information. The FP-Tree is further subdivided into Conditional FP-Trees for each frequent item, allowing them to be mined independently (Han et al. 2000).

In developing our analysis, we specified in FPGrowth the minimum support 0.01 out of total transactions. The association rules are generated and used Lift value as a metric > 1%. The result of that is pretty similar to the one plotted in figure 23 and 24 and this happened because we ranked our products using the Lift metric for both algorithms.

When analysing the correlation between those items, in order to have a better visualization of it, we decided to use as a parameter antecedents that have more than one item and its respective consequents. Let's have a look at this:

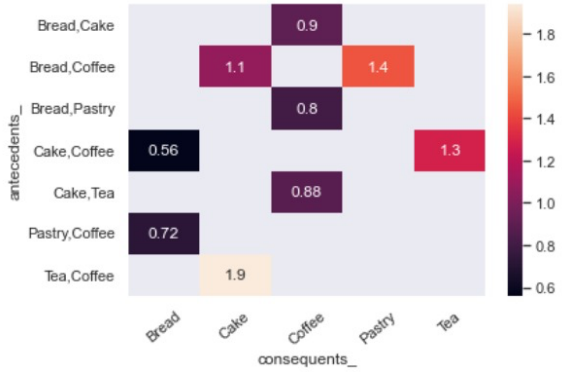

Figure 25 Association Rule Heatmap. Source: Jupyter Notebook by the authors.

The most relevant value is (Tea and Coffee) with "Cake" as a consequence choice. Then considering the correlation of 1.9 we conclude that the customer buys these items together (Tea, Coffee and Cake). On the other hand, we have less correlation between (Bread) with (Cake, Coffee) and (Pastry, Coffee).

c) Can you express major divergence between these models?

The Apriori method (retrieves [n + 1] itemsets from n items) evaluates each item as an itemset and counts the support based on their frequency in the database, then captures those with support equal to or more than the minimum support threshold. The extraction of each frequent itemset in this procedure needs the algorithm to scan the complete database until no more itemsets with more than the minimal threshold support remain (Verma, Y. 2021).

Apriori algorithm was generating the candidates for making the item sets. The data in the FP-Growth method is represented in a tree form. The FP-tree is a type of lexicographic tree construction. Which is responsible for keeping the association information between the frequently used items.

After constructing the FP-Tree, it is divided into a series of conditional FP-Trees for each frequent item. Separately, a collection of conditional FP-Trees can be mined and assessed.

d) Compare and contrast the machine learning results obtained based on both algorithms.

What distinguishes FP-growth from the Apriori frequent pattern mining method is that FP-Growth is a frequent pattern mining algorithm that does not need candidate creation.

Instead, the FP-Growth uses an FP-tree (frequent pattern tree) data structure without explicitly developing candidate sets, making it particularly useful for huge datasets.

The set of tables below shows us that, when performing both algorithms, despite the fact that we got the same results, the order of these values per item are different:

	support	itemsets
0	0.036344	(Alfajores)
1	0.016059	(Baguette)
2	0.327205	(Bread)
3	0.040042	(Brownie)
4	0.103856	(Cake)
..	...	...
56	0.023666	(Coffee, Toast)
57	0.014369	(Sandwich, Tea)
58	0.010037	(Bread, Cake, Coffee)
59	0.011199	(Bread, Pastry, Coffee)
60	0.010037	(Coffee, Cake, Tea)

Figure 26 Support - Apriori. Source: Jupyter Notebook by the authors.

	support	itemsets
0	0.327205	(Bread)
1	0.029054	(Scandinavian)
2	0.058320	(Hot chocolate)
3	0.054411	(Cookies)
4	0.015003	(Jam)
..	...	...
56	0.019651	(Brownie, Coffee)
57	0.010777	(Bread, Brownie)
58	0.023666	(Toast, Coffee)
59	0.018067	(Scone, Coffee)
60	0.010882	(Spanish Brunch, Coffee)

Figure 27 Support - FP-growth. Source: Jupyter Notebook by the authors.

Focusing on the model performance and going deeper into the metrics of Support, Confidence, and Lift, we can illustrate it in a Heatmap and compare it according to each algorithm. It is worth noting that the values used in the Apriori have a value concentration of around 1-1.19, whereas the values used in FP-Growth vary from 0.5 to 1.49:

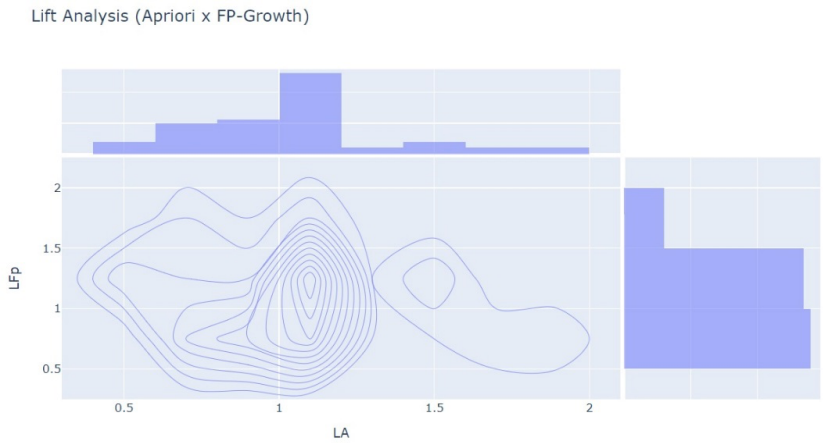

Figure 28 Algorith Comparison. Source: Jupyter Notebook by the authors.

8. Conclusion

The real-world problems are usually “compiled” in a lot of raw data that must be turned into meaningful information to solve such problems. To do that, the understanding of the data is the first step of this machine learning journey.

Going through this machine learning journey, we will face a lot of related and unrelated attributes, so that, descriptive statistics will be analysed, correlation will be checked, features will be selected according to the model performed and then we have analysis process in progress to result into a prediction outcome.

As we could see through this research and the process to get the answers of the questions highlighted in this report, theory is important, but not more important than experimenting with framed data and analysing real results by comparing performances and testing mathematical applicability.

Along with that, the data visualization plays an important role in data analysis because as soon as the human eye views charts or graphs it tries to find patterns in the graph and as a result quick understanding is generated. It is due to proper visualizations that we can understand not only the data, but also feature selections for the machine learning model.

This report contains more than 5000 words due to the complexity of the questions answered here and the fact that the group guided its respective answers in a more analytical way, mixing theory and coding practice.

9. References

Agrawal R., Srikant R. (1994). Fast algorithms for mining association rules. In: Paper presented at the proceedings of the 20th international conference on very large data bases, Santiago.

Ajitsaria, A. (2022). Build a Recommendation Engine With Collaborative Filtering. Real Python. Online]. Available at: https://realpython.com/build-recommendation-engine-collaborative-filtering/. Accessed 28th May 2022.

Brownlee, J. (2017). How to Decompose Time Series Data into Trend and Seasonality. Machine Learning Mastery. [Online]. Available at: https://machinelearningmastery.com/decompose-time-series-data-trend-seasonality/. Accessed 27th May 2022.

Brownlee, J. (2017). A Gentle Introduction to Autocorrelation and Partial Autocorrelation. Machine Learning Mastery [Online]. Available at: https://machinelearningmastery.com/gentle-introduction-autocorrelation-partial-autocorrelation/. Accessed 27th May 2022.

Brownlee, J. (2018). How to Calculate Correlation Between Variables in Python. Machine Learning Mastery [Online]. Available at: https://machinelearningmastery.com/how-to-use-correlation-to-understand-the-relationship-between-variables/. Accessed 26th May 2022.

Ghosh, P. (2021). Dollar Value Could Slide This Year As Global Economy Picks Up After Covid, Analysts Say. Forbes. [Online]. Available at: https://www.forbes.com/sites/palashghosh/2021/04/01/dollar-value-could-slide-this-year-as-global-economy-picks-up-after-covid-analysts-say/. Accessed 28th May 2022.

Han J, Pei J, Yin Y (2000) Mining frequent patterns without candidate generation. ACM SIGMOD Rec 29(2):1–12

Kuila, A. (2021). Transactions from a bakery. [Online]. Available at: https://www.kaggle.com/datasets/akashdeepkuila/bakery. Accessed 14th May 2022.

Mesidor, F. (2019). A basic guide to time series analysis. Towards Data Science. [Online]. Available at: https://towardsdatascience.com/a-basic-guide-into-time-series-analysis-2ad1979c7438. Accessed 26th May 2022.

Muvi, M. (2021). The Difference Between Collaborative and Content Based Recommendation Engines. Medium. [Online]. Available at: https://muvi-com.medium.com/the-difference-between-collaborative-and-content-based-recommendation-engines-ade24b5147f2. Accessed 26th May 2022.

Prabhakaran, S. (2019). Augmented Dickey Fuller Test (ADF Test) – Must Read Guide. Machine Learning Plus. [Online]. Available at: https://www.machinelearningplus.com/time-series/augmented-dickey-fuller-test/. Accessed 27th May 2022.

Rowe, W. (2018). Mean Square Error & R2 Score Clearly Explained. BMC. [Online]. Available at: https://www.bmc.com/blogs/mean-squared-error-r2-and-variance-in-regression-analysis/. Accessed 29th May 2022.

Sharma, H. (2020). Data Visualization Using Statistical Charts by Plotly. Towards Data Science. [Online]. Available at: https://towardsdatascience.com/data-visualization-using-statistical-charts-by-plotly-e730c27de1fd. Accessed 28th May 2022.

Sivek, S. (2020). Market Basket Analysis 101: Key Concepts. Towards Data Science. [Online]. Available at: https://towardsdatascience.com/market-basket-analysis-101-key-concepts-1ddc6876cd00. Accessed 27th May 2022.

Verma, Y. (2021). Apriori vs FP-Growth in Market Basket Analysis – A Comparative Guide. Analytics India Magazine. [Online]. Available at: https://analyticsindiamag.com/apriori-vs-fp-growth-in-market-basket-analysis-a-comparative-guide/. Accessed 29th May 2022.

Verma, Y. (2021). Complete Guide To Dickey-Fuller Test In Time-Series Analysis. [Online]. Available at: https://analyticsindiamag.com/complete-guide-to-dickey-fuller-test-in-time-series-analysis/. Accessed 18th May 2022.

Team contribution

Member	Participation
Fabiane dos Santos Teixeira	Research and storytelling of the analysis (report), focusing on the development of the time series and recommendation systems, as well as the Tableau research and plots.
Nuno Alfredo Ribeiro Teixeira de Almeida	Plotly visualizations for Time Series as well as correlation plots. Performed Market Basket analysis and research.
Valesca Soledad Bravo Bravo	Time series analysis in python from scratch, exploring each step as well as plotting different times of the time series.
Thiago Medeiros de Souza	Recommendation analysis in python from scratch, exploring each step, including performing different metrics of recommendation.