Using Machine Learning to Investigate Customer Segmentation – Recency Frequency Monetary model, Principal Component Analysis and t-distributed Stochastic Neighbour Embedding



By

Desmond Leung

Faculty of Engineering and Technology

Supervised by Dr Ivan Olier-Caparroso

April 2021

# Abstract

Within the marketing industry, it is essential that a company can recognise future potential customers and identify current most valuable customers. This can be done using clustering techniques in which customers of similar shopping behaviour are categorised into groups. Effective use of these techniques can help recognise customers’ needs and therefore makes it easier to tailor to them, increasing the likelihood of retaining high-value customers. Additionally, companies can recognise future potential customers, for example if they belong in a valuable cluster. Overall, this can boost the performance of a business and therefore increase revenue of the company.

This report uses an online UK-based retail dataset. The data contains all transactions between 01/12/2009 and 09/12/2011, in which the company mainly sells unique giftware. This project first uses the recency, frequency, monetary (RFM) model to identify customers’ RFM score. This was based on how recent a customer’s last purchase was, how often they buy and the total price of the items purchased. The RFM scores range from 111 which is considered most valuable to a business to 444 which is considered least valuable. The features recency, frequency and monetary were defined into certain scores based on their quartiles.

Then, principal component analysis (PCA) is carried out for dimensionality reduction of the RFM features and clustering of the RFM scores. The feature of lowest variance is excluded which allows a two dimensional map to be produced, where variance of the features are determined by the eigenvalues of the covariance matrix of the standardised data.

Finally, t-distributed stochastic neighbour embedding (t-SNE) will be used to see if there is an improvement in the two dimensional embeddings compared to PCA, which unlike t-SNE, is a linear dimensionality reduction algorithm. In contrast to PCA, t-SNE preserves local structure as best it can instead of larger pairwise distances that PCA tries to preserve.

The results of the embeddings for PCA and t-SNE show that t-SNE only gives a slight improvement to the map compared to PCA. This may indicate that there is no real structure present in the original data to begin with or that a different way of distinguishing the RFM scores is more suitable.

Contents

[Abstract 2](#_Toc69486503)

[Introduction 4](#_Toc69486504)

[Literature Review 4](#_Toc69486505)

[Methods 6](#_Toc69486506)

[Data Source and Description 6](#_Toc69486507)

[Software Information 7](#_Toc69486508)

[Data Pre-processing 7](#_Toc69486509)

[Recency, Frequency, Monetary 9](#_Toc69486510)

[Principal Component Analysis 9](#_Toc69486511)

[t-SNE 10](#_Toc69486512)

[Results and Discussion 11](#_Toc69486513)

[RFM Analysis 11](#_Toc69486514)

[13](#_Toc69486515)

[PCA 14](#_Toc69486516)

[17](#_Toc69486517)

[t-SNE 18](#_Toc69486518)

[Conclusion 19](#_Toc69486519)

[References 20](#_Toc69486520)

[Appendix 23](#_Toc69486521)

# Introduction

This project looks at how Python can be used to carry out RFM Analysis, PCA and t-SNE to investigate customer segmentation. The dataset used is an online retail dataset for a UK-based company that mainly sells giftware. Firstly, RFM assigns a recency, frequency and monetary value for each customer which allows an RFM score to be assigned to each customer to reflect how valuable they are to the business. Then PCA is used to identify the two most features with most variation which will be used in the cluster analysis with t-SNE. Both embeddings of PCA and t-SNE will be compared.

## Literature Review

Because customer satisfaction and retention in the marketing industry are becoming increasingly important each year, businesses need to be accurately identify customers’ needs and behavioural attributes to increase the revenue, performance and sustainability of the company in such a competitive sector (Rust and Zahorik, 1993). One way of doing this is by creating a working customer relationship management (CRM) process (Ngai, Xiu and Chau, 2009).

Efficient use of CRM means being able to maintain long-term and valuable relationships with the most loyal customers which will ultimately grow the business (Ngai, Xiu and Chau, 2009). This is done by focusing on customer retention and relationship development using people, processes and information technology (Chen and Popovich, 2003). CRM can give varying knowledge of customers on the socialisation, externalisation, combination, and internalisation (SECI) model (Srisamran and Ractham, 2014). In addition to knowledge about customers, there are two other types of knowledge in CRM processes: knowledge for customers and knowledge from customers (Gebert *et al.*, 2003). It is important for companies to manage these exchanges of knowledge as they will ultimately aid in how improvements to existing products in stock can be made due to a better understanding in customers’ needs and view of the company. Appropriate advertising of items which directly meets a customer’s needs will solidify the customer-company relationship (García-Murillo and Annabi, 2002). According to (Khodakarami and Chan, 2014), CRM can be split into three main categories: analytical, collaborative and operational.

Analytical CRM uses data mining techniques on customer data such as purchasing history to give meaningful insights and trends into customer information and behaviour. It can also help businesses assess how well certain products are selling in comparison (Payne and Frow, 2005).

Some data mining techniques have already been used as a form of analytical CRM. For example, (Maryani and Riana, 2017) used RFM which groups customers into clusters based on the time of latest visit, frequency of visits and income from specific customers in motorcycle and car exhaust industry companies. RFM has been used in a wide range of areas for example hotels in the hospitality industry (Dursun and Caber, 2016) and retail banking (Khajvand and Tarokh, 2011). All of which indicates the importance of segmentation techniques to better understand customers for customer retention. It is also believed that retaining existing valuable customers is more important than finding new ones (Christy *et al.*, 2018).

Another commonly used clustering method for customer segmentation is k-means clustering (Kansal *et al.*, 2018). This is a type of unsupervised learning where you tell the algorithm how many clusters you want which is the ‘k’ in ‘k-means’. Firstly, centroids are placed in random locations in a dimensional space. The algorithm then finds the nearest centroid for every data point and assigns this point to the respective cluster. The position of each centroid is recomputed using the average of the data points assigned to that cluster. The process is iterated until no data points change their previously allocated cluster i.e. the algorithm has converged (Kansal *et al.*, 2018), (Luo *et al.*, 2012). This method of clustering is found to be simple to use and more effective on large datasets. However, there are drawbacks such as high unpredictability of ‘k’ and inability to process noise in data (Luo *et al.*, 2012; Kansal *et al.*, 2018), (Wu *et al.*, 2008).

This project will be looking at the methods principal component analysis (PCA) and t-distributed stochastic neighbour embedding (t-SNE) which has been widely used for clustering analysis (Granato *et al.*, 2018; Liu *et al.*, 2021). However, these methods are rarely used in the context of customer segmentation which is what this project will be exploring.

PCA is an unsupervised machine learning algorithm most commonly used for dimensionality reduction, as seen in (Alkhayrat, Aljnidi and Aljoumaa, 2020; Bandyopadhyay, Thakur and Mandal, 2020; Margaritis *et al.*, 2020). These studies find that PCA takes advantage of the original dataset having highly correlated variables which causes noise and redundant information. PCA is then applied such that as much of variation in the original data is preserved while reducing how many columns/features are present in the data (Galarnyk, 2017). Firstly, (Alkhayrat, Aljnidi and Aljoumaa, 2020) shows that the original dataset matrix is standardized by use of the Standard Scaler:

This is done to avoid bias towards the features originally possessing a lot of the explained variation in the features.

Next, the covariance matrix is calculated by multiplying the standardised matrix by its transpose. The result is a square, symmetric matrix that tells us the variance of each feature on its diagonal and the covariance between every possible pair of variables everywhere else (Alkhayrat, Aljnidi and Aljoumaa, 2020). Any positive covariances means that the two features are correlated and any negative covariances means that the two features are inversely correlated. For example, if there are 100 features in the standardised dataset, the covariance matrix is structured as follows:

The pairs of eigenvalues and eigenvectors were then found from the covariance matrix which allowed them to determine the principal components (PCs) (Lateef, 2020). These were calculated as the new uncorrelated variables that are linear combinations of the original variables. As much information as possible is retained into the first PCs. After computing this, a scree bar chart was used to show the proportion of explained variance each PC has (Maklin, 2019). PC1 had the most percentage of explained variance, PC2 had the second most, etc. Eigenvectors of the covariance matrix are the directions of the lines or axes such that most variance in the data can be captured. These are known as the principal components. By ordering the eigenvalues from highest to lowest, which their associated eigenvector, they found the PCs in order of their variance. The percentage of variance associated with each component is calculated by dividing the eigenvalue by the sum of all eigenvalues.

After computing all the components in order of their variance, the decision of how many variables to keep can be made. It is crucial to have less features than the original data while retaining a lot of the information. Studies into PCA have labelled the acceptable total variation in the new data differently. For example, (Alkhayrat, Aljnidi and Aljoumaa, 2020) stated that “72% of the total variation is an acceptably large percentage” while (Margaritis *et al.*, 2020) used the threshold of 80%.

Conversely, t-SNE is a non-linear dimensionality reduction technique, originally developed by (García-Alonso, Pérez-Naranjo and Fernández-Caballero, 2014). This technique allows data to be separated into clusters that could not be separated clearly with linear methods such as PCA and multidimensional scaling while maintaining the structure of the high dimensional data. Similarly to PCA, t-SNE “visualizes high-dimensional data by giving each datapoint a location in a two or three-dimensional map”. It is also said that this algorithm is a “variation of Stochastic Neighbor Embedding that is much easier to optimize, and produces significantly better visualizations by reducing the tendency to crowd points together in the center of the map”.

t-SNE works by calculating the similarity between each pair of data points which are converted into joint probability distribution *P* that the points belong in the same cluster. Likewise a joint-probability distribution *Q* is computed, that describes the similarity in the low-dimensional space. “The goal is to achieve a representation, referred to as embedding, in the low dimensional space where *Q* faithfully represents *P*” (Pezzotti *et al.*, 2017).

# Methods

## Data Source and Description

The data, found at (Dua, Dheeru; Graff, 2019) was taken from the UCI Machine Learning Repository which is a widely used collection of datasets in which machine learning algorithms can be applied.

The data contains all transactions of purchase from 01/12/2009 to 09/12/2011.

The columns of the data are given as “invoice number, stock code, description, quantity, invoice date, price, customer ID, country”.

## Software Information

Python versions used: 3.7, 3.9.

Packages used for RFM: DateTime version 4.3, matplotlib version 3.3.3, pandas version 1.1.4.

Packages used for PCA: pandas version 1.1.4, numpy version 1.19.4, sklearn version 0.0, matplotlib version 3.3.3, plotly version 4.14.3.

Packages used for t-SNE: sklearn version 0.0, pandas version 1.1.4, matplotlib version 3.3.3, plotly version 4.14.3, seaborn version 0.11.0.

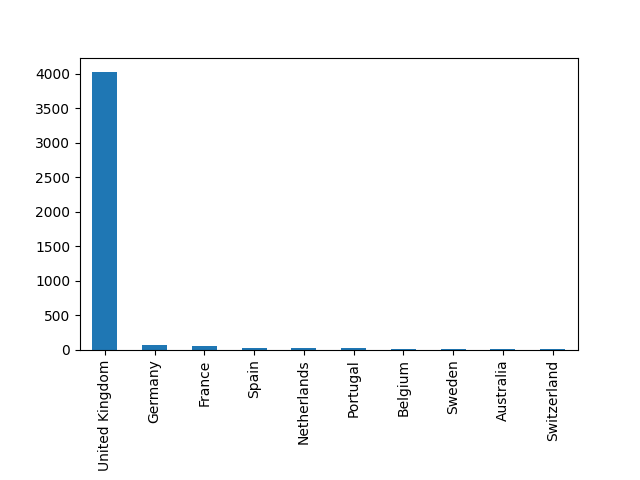
The PyCharm IDE was used throughout to input and run the Python code.

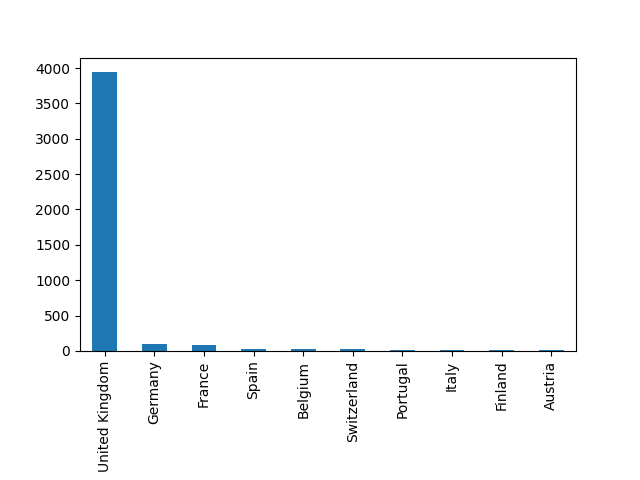
## Data Pre-processing

Before analysis, data pre-processing is crucial to clean the dataset. This is because real data collection can contain a lot of unwanted rows of data arising from errors such as missing values, duplicates of rows etc. Performing data pre-processing will ensure that the performance of the analysis is not negatively affected and that in some cases, the code can be ran in the first place.

Firstly, rows of empty customer ID are omitted by keeping the rows that have an entry of customer ID. This is done by using the pd.notnull() command in the pandas package.

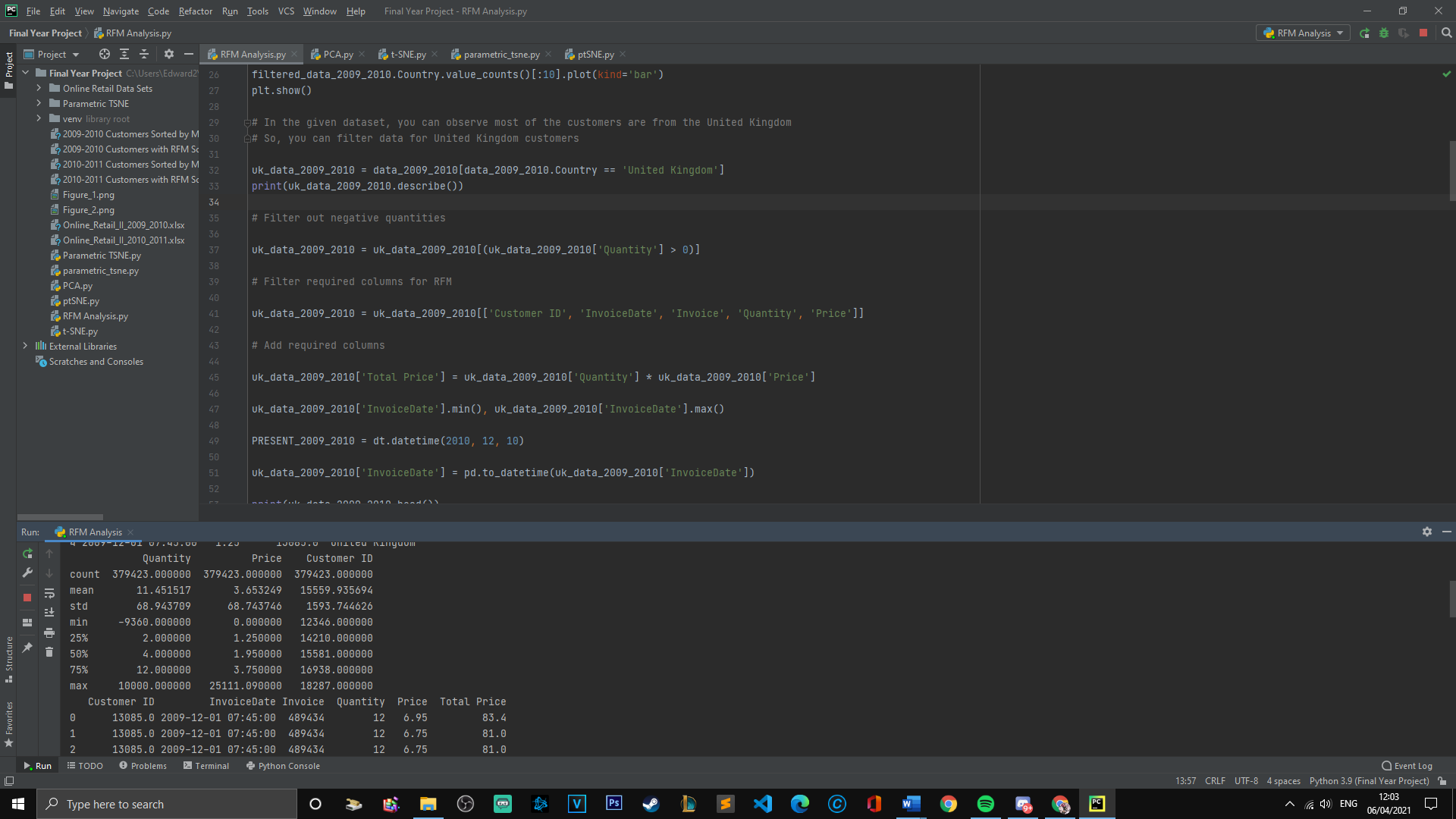
Then, eliminate rows that have the same country of residence and customer ID, the drop\_duplicates() command in python can be used. This will also tell us how many number of unique customers there are.

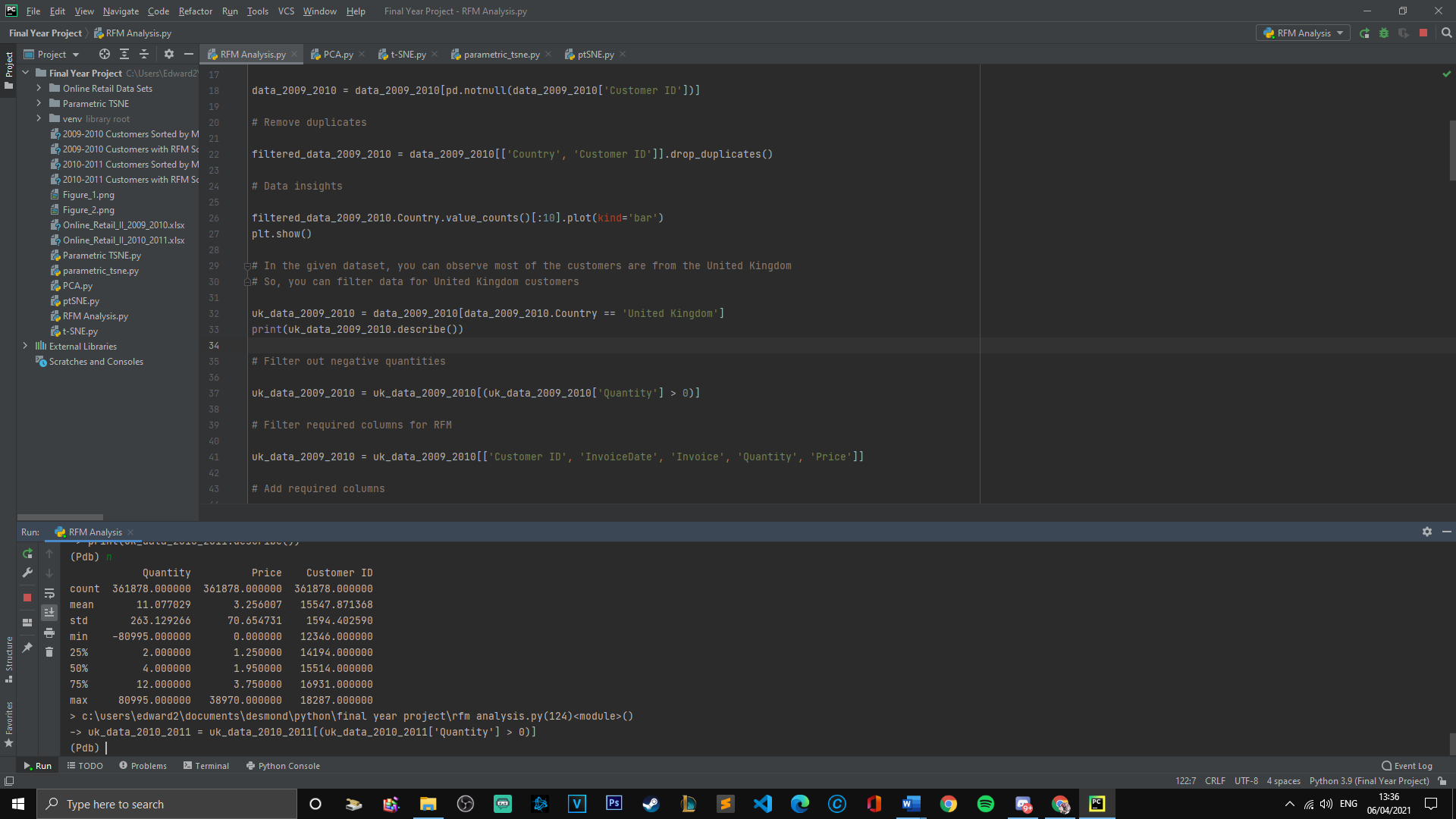




From the bar charts showing the number of unique customers from each country for both years, the majority are from the UK. This means we can filter out any customers who do not reside in the United Kingdom (Navlani, 2018).

Now we can obtain summary statistics of the UK based data by using the describe() function in Python.





From both years, there are negative quantities of items purchased which is impossible. So we can filter rows out where the quantity is negative.

## Recency, Frequency, Monetary

RFM has been a commonly used marketing analysis tool for customer segmentation based on customer behaviour as seen in the following studies (Navlani, 2018; Sparks, 2019). For RFM, we need the recency of the latest purchase of every customer, how many times each customer has ordered and the sum of the prices of the items bought for each customer. For this, we will need only certain columns. These are customer ID, InvoiceDate, Invoice, Quantity and Price. For recency, calculate the number of days between the present day and the latest day of purchase for each customer. For frequency, sum up the number of invoices for each customer. Finally for monetary, calculate the sum of quantity multiplied by price.

Next, separate the features recency, frequency and monetary into quartiles. Customers that have had the most recent invoices and made the most transactions most frequently are thought to be the most valuable to a business . An RFM score can then be assigned to each customer, a score of 111 indicates a low recency value, a high frequency value and a high monetary value which is most valuable whereas a score of 444 indicates a high recency value, a low frequency value and a low monetary value which is least valuable.

## Principal Component Analysis

In this project, PCA was used for dimensionality reduction and clustering for the remaining features in the reduced data set. The three features obtained from RFM will be used in PCA (recency, frequency and monetary) for each customer. These features are then standardised onto unit scale to avoid any bias towards variables with high variance.

Using the standardised data, the covariance matrix can be computed along with its eigenvector and eigenvalue pairs by eigendecomposition. Ordering the eigenvalues in descending order, this will tell us where the most variance in the data set lies. Alternatively, a scree plot can be produced in python which will also tell us the percentage variation for each variable. The PC scores or linear combinations of the original features can then be plotted for the two first two PCs with the highest total variance in the data. Finally, the loadings plot can be computed which shows which features are more prominent for which observation, in this case RFM score.

## t-SNE

In this project, t-SNE was implemented Python using the sklearn.manifold module which allows data embedding techniques to be carried out.[[1]](#footnote-1) The motive of the development of t-SNE (Pezzotti *et al.*, 2017) was to make low dimensional embeddings which are more reliable than that of PCA. This is because PCA is mainly concerned with preserving large pairwise distances, making dissimilar points far apart on the low dimensional map. t-SNE was developed such that small Euclidian distances are preserved instead. This ensures that the local structure of data points and their nearest neighbours can be seen after reducing the dimension of many types of manifolds, making t-SNE more reliable.

t-SNE starts by measuring the local similarity between points in the high dimensional space such that a probability, *pij*, of picking two pairs of points is proportional to their similarity in the high dimensional space. This gives a set of probabilities *pij* that measures the similarity between pairs of points *ij*, given from the following (Pezzotti *et al.*, 2017):

Dissimilar pairs points on the high dimensional space are given a *pij* value that is extremely small and similar pairs of points on the high dimensional space are given a large *pij* value.

Similarly, each high dimensional object is then represented on a 2D or 3D low dimensional map such that similarities between pairs of points *ij* is measured which gives a set of probabilities *qij* (Pezzotti *et al.*, 2017) using a Student’s t-distribution:

A heavy-tailed Student’s t-distribution is used so that dissimilar points in the high dimensional data are modelled to be very far apart on the embedding (García-Alonso, Pérez-Naranjo and Fernández-Caballero, 2014). The aim is to have the probabilities *pij* reflect the *qij* probabilities as this means that the structure of the high dimensional data is preserved into the low dimensional map.

This is achieved by minimising the following cost function known as the Kullback-Leibler divergence (Pezzotti *et al.*, 2017):

The Kullback-Leibler divergence tries to model large *pij* values with large *qij* values. This is because large *pij* values modelled with small *qij* values will result in a large result for the Kullback-Leibler divergence which is unwanted. This ensures that the local similarity of the high dimensional data is preserved.

The TSNE() command in Python has many parameters that can be altered to give different qualities of embedding, namely: perplexity, early exaggeration factor, learning rate, maximum number of iterations and angle.[[2]](#footnote-2)

For the application of t-SNE in this project, the learning rate and perplexity of the model are adjusted to see if there will be improvements in the embedding of the data points. The default values of perplexity and learning rate of 30 and 200 respectively will be used for one mapping and another mapping will be produced with a perplexity of 100 and learning rate 500.

Perplexity describes how many nearest neighbours will be considered for each data point. Hence a larger and denser data set will likely need a higher perplexity. Learning rate is also a crucial parameter because it affects the gradient descent of the Kullback-Leibler divergence. If learning rate is too high, the cost function to be minimised will increase during optimization[[3]](#footnote-3).

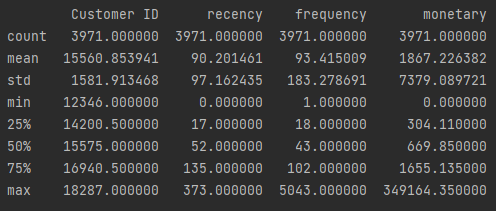
# Results and Discussion

## RFM Analysis

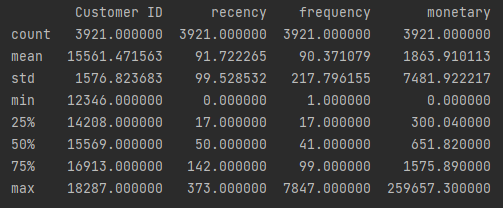
After finding the RFM score for each customer, 378/3971 and 409/3921 customers were found to have an RFM score of 111 for years 2009-2010 and 2010-2011 respectively. These customers are considered to be most valuable.

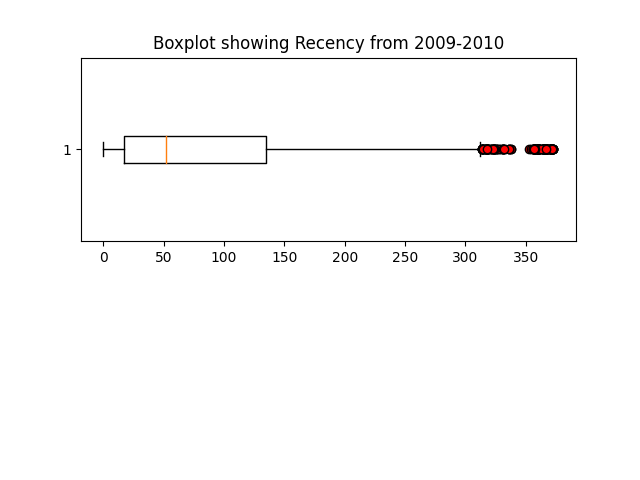
It is important to note what the numerical cut-offs at the quartiles were found to be. These ultimately determine the differences between RFM scores. For example in 2009-2010, the upper bound for a frequency score of 4 is 18 whereas in 2010-2011, it is 17.

2009-2010

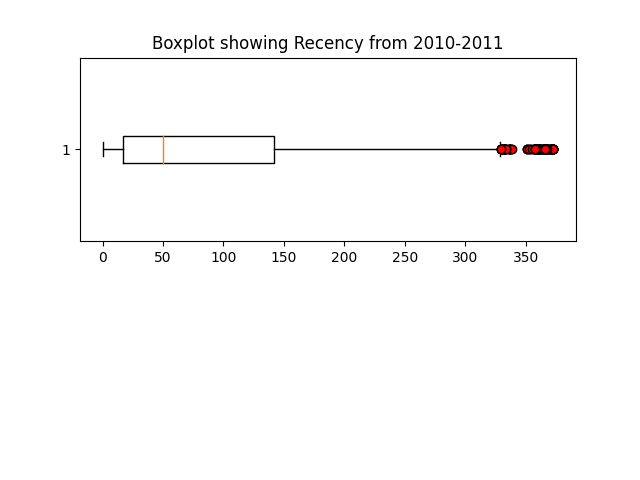


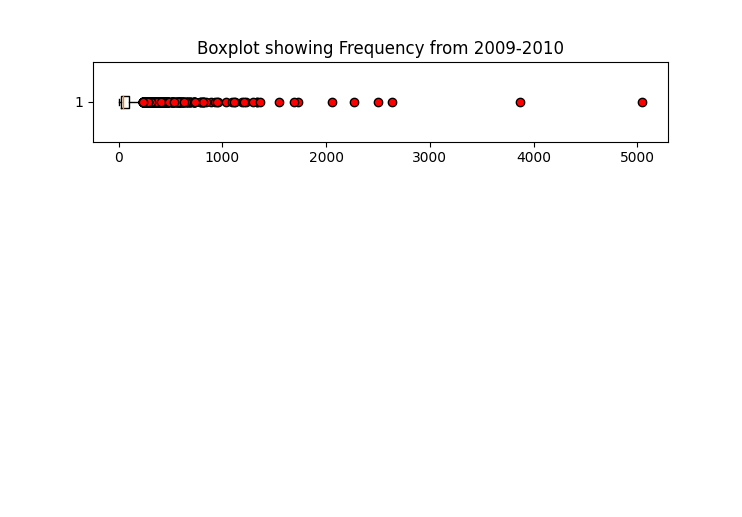
2010-2011

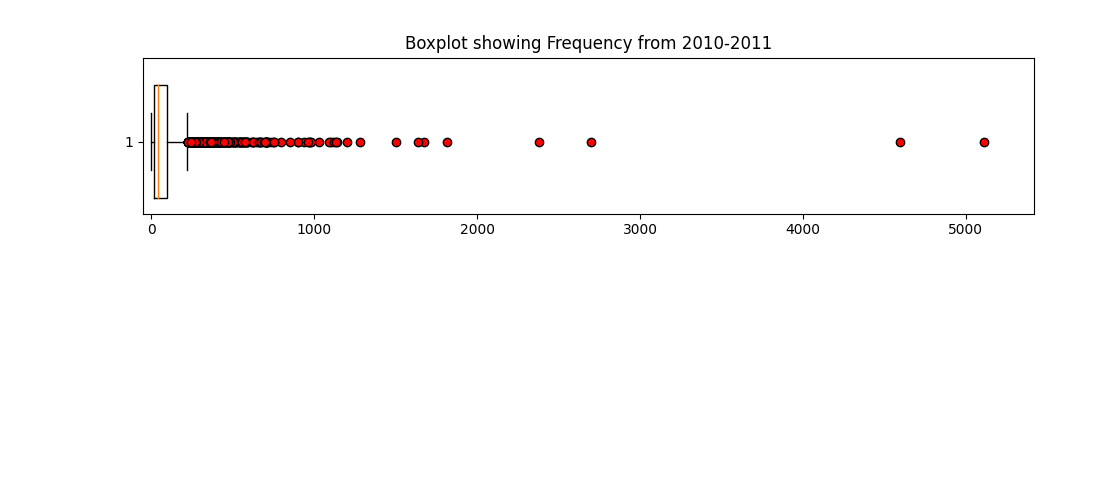


From the quartiles, boxplots were plotted for each feature from both years:

The recency boxplots show that the feature data is extremely positively skewed since 50% of the data lies at or below the value of 50 and 52 for 2009-2010 and 2010-2011 respectively while the range of the data goes from 0-373. Additionally, the mean values of recency are around 40 higher than the medians for both years which also indicates positive skew.







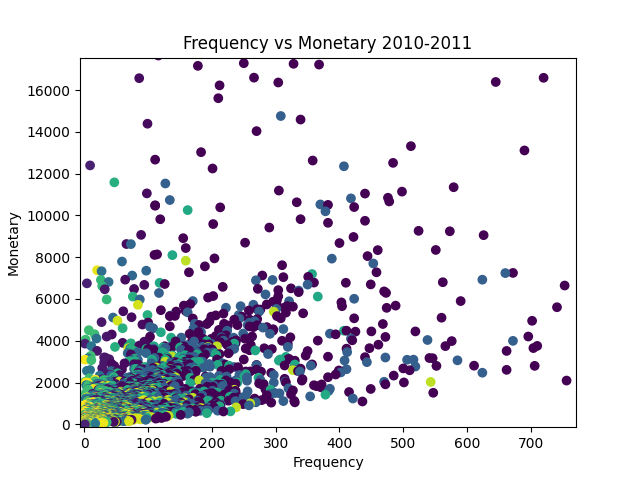
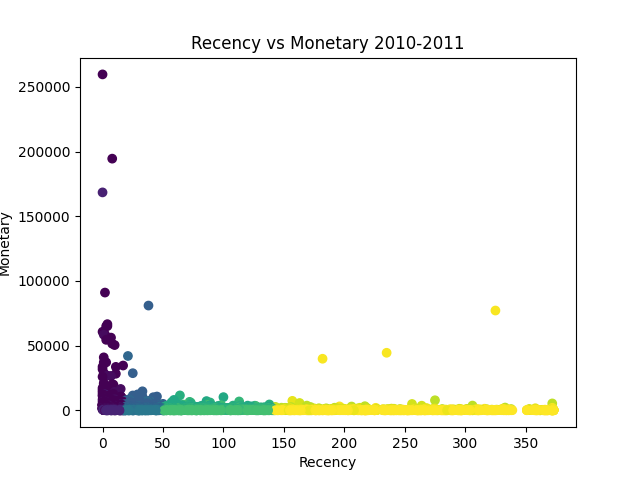
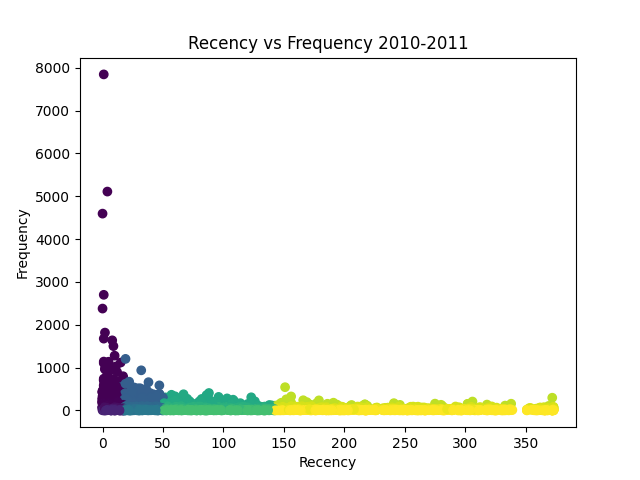
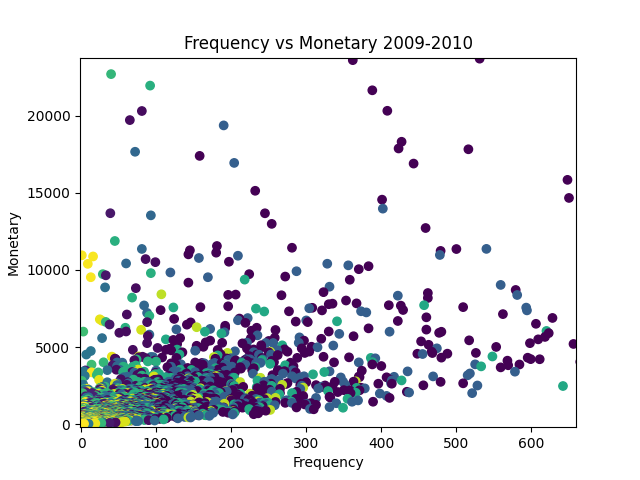
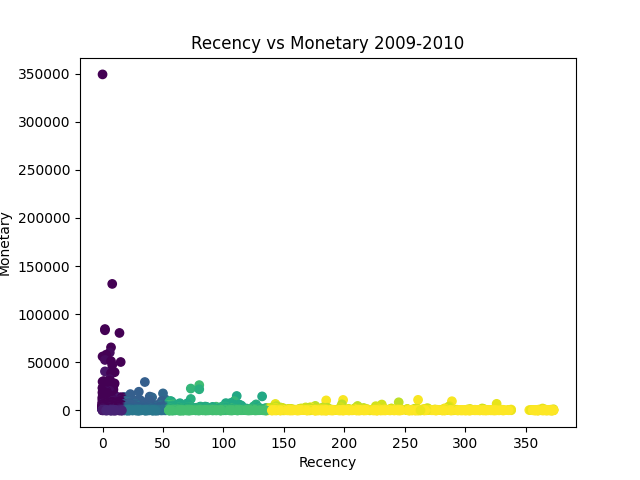
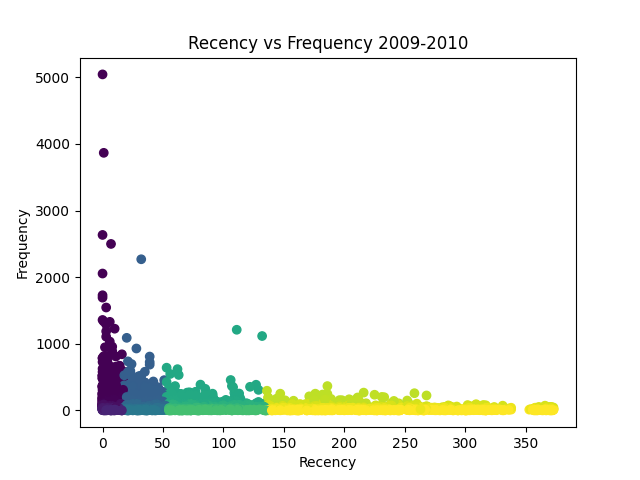
## 

Overall, the boxplots of the 3 features in the data set all show positive skew, meaning the data is most dense near the value of 0 and becomes less dense as the points move away from 0.

The outliers shown in the boxplots as red circles were not removed in the methods because the boxplots show that there are a reasonably large amount of outliers in the data, especially for the monetary and frequency features.

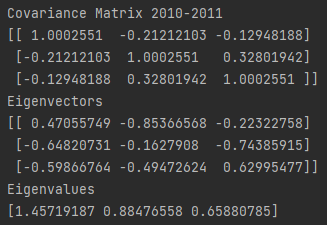
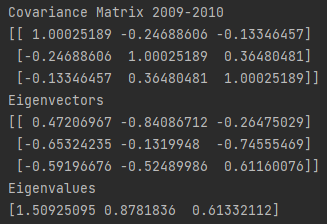
## PCA

Before PCA was used, the 2 dimensional plots of the RFM features were plotted first. (Some of these plots have been zoomed in to make them easier to see. Full plots are included in the appendix.)



The plots from both years are very similar. The plots with recency on the x axis show that the cut offs at the quartiles are comparable between the two years. The shades dark purple points indicate a recency score of 1, blue indicates a score of 2, green 3 and yellow 4. The uppermost left points which are separated from the rest of the points on these plots are the most valuable customers as they have an extremely high frequency value and a low recency value. These customers are very likely to have an RFM score of 111. There are 3 yellow anomalies on the graph of recency vs monetary 2010-2011. These points have a high recency value, high monetary value but a very low frequency value which is unexpected.

For the plots of frequency against monetary, the RFM scores appear to be scattered for the most part. However, there are more yellow points the closer the distance to the origin. This is not a surprise since a low frequency and monetary value will often indicate that a customer does not purchase often. On the other hand, due to the vast amount of overlap in the plot, it is also fair to say that recency does not depend on monetary or frequency values. This overlap is likely caused because the features were separated into scores at their quartiles and that the monetary value dominates due to their values being much larger than recency and frequency. A large portion is caused by every customer having different shopping habits and wealthiness which results in discrepancy in the data.



The covariance matrices show that recency is inversely proportional to frequency and monetary value of the customers because the covariance between them have negative values of -0.24688606 and -0.21212103. This is to be expected because a lower value of recency is more valuable whereas a higher value of frequency and monetary is more valuable. Additionally, frequency and monetary are proportional to each other in both data sets as indicated by positive values of covariance 0.36480481 and 0.32801942.

Eigenvectors tell us the direction of maximum variance and the corresponding eigenvalues are their scalar representations.

From the eigenvalues for the 2009-2010 dataset, it was found that:

The first PC explains of the total variance.

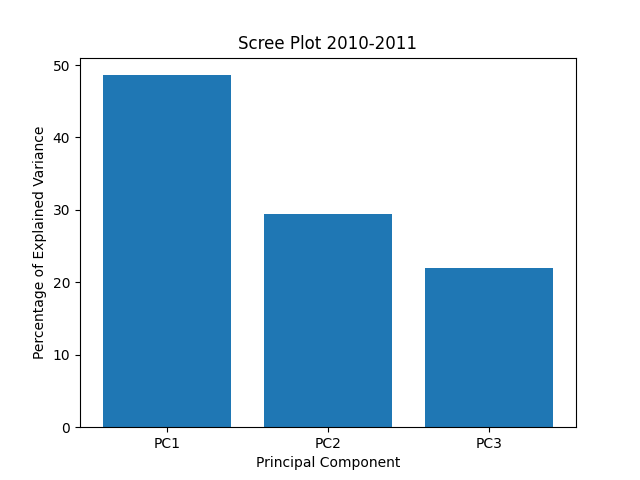
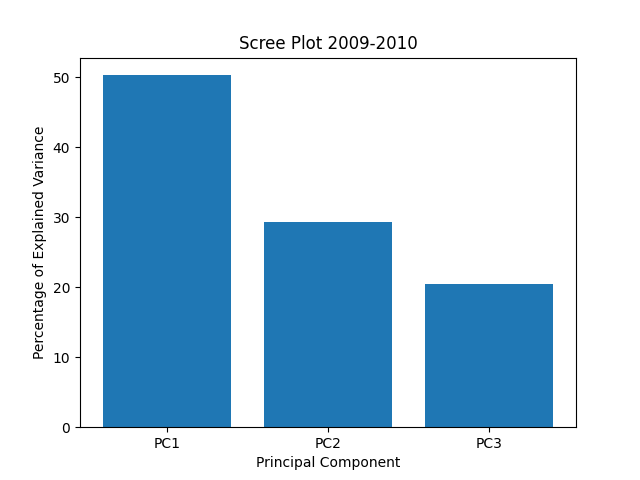
The first two PCs explains of the total variance.

From the eigenvalues for the 2010-2011 dataset, it was found that:

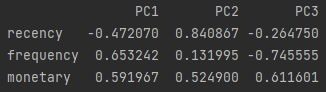
The first PC explains of the total variance.

The first two PCs explains of the total variance.

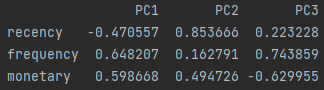
In both cases, the first two PCs were used because the first PC alone did not capture enough total information in the data.

Alternatively, scree plots can show the same results:

The loadings, that are used to calculate the PC scores, can be computed to be as follows:

2009-2010:

2010-2011:



The loading scores associated with each variable are very similar across both years. These values allow us to calculate the PC scores using the scaled data. For example, the highest monetary customer in 2009-2010 after scaling has a recency -0.92847426, frequency 2.91169795 and monetary 47.07096325.

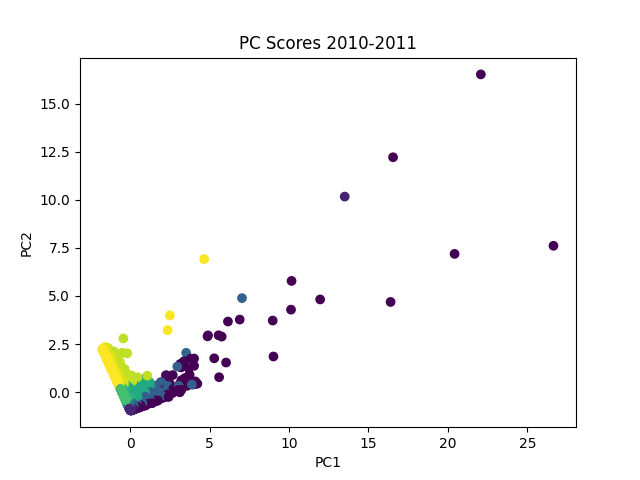
For this customer:

This gives this customer the coordinate (30.20, 24.31) on the PC scores plot of PC1 against PC2.

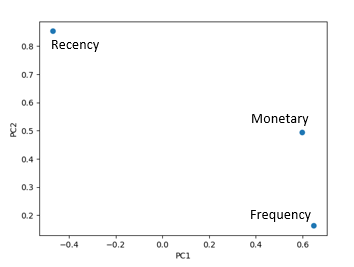
The first PC highlights the contrast between the recency variable with the frequency and monetary variables due to the loading scores having opposite signs and similar absolute values. Whereas PC2 is dominated by the loading scores for recency and monetary and frequency having very little impact on the direction of the PC.

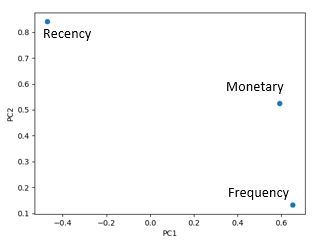
The loading scores can also be plotted with the corresponding PC scores plot (more detailed plots in appendix):

# 



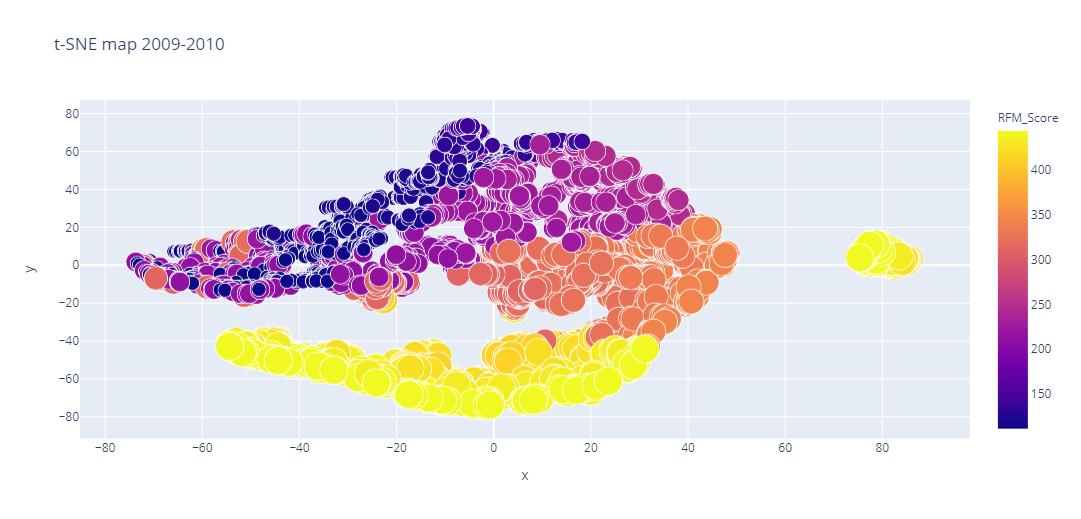
# 

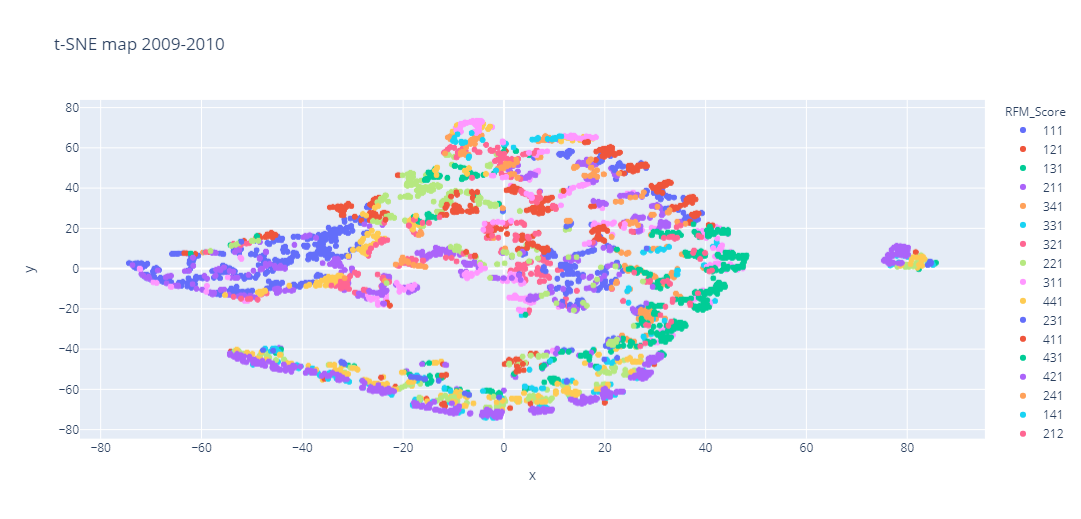




The loading scores plot confirms the similarities between the two years. The recency feature is most important for the yellow points on each PC Score plot whereas the dark purple points that separate from the cluster of the points at the origin are predominantly explained the monetary feature and a little by frequency. Generally, the further the points are to the right of the PC scores plots, the more valuable their RFM score is.

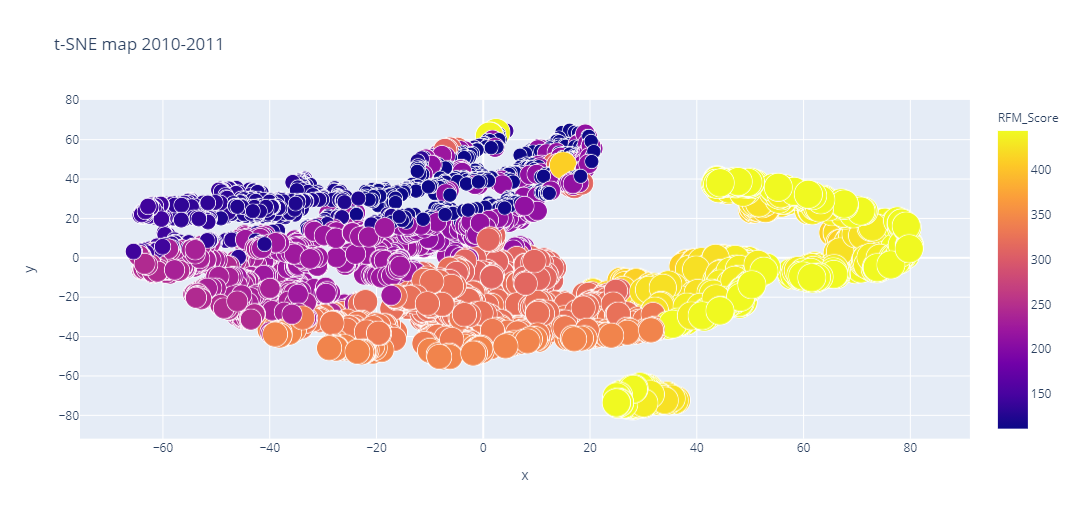
## t-SNE

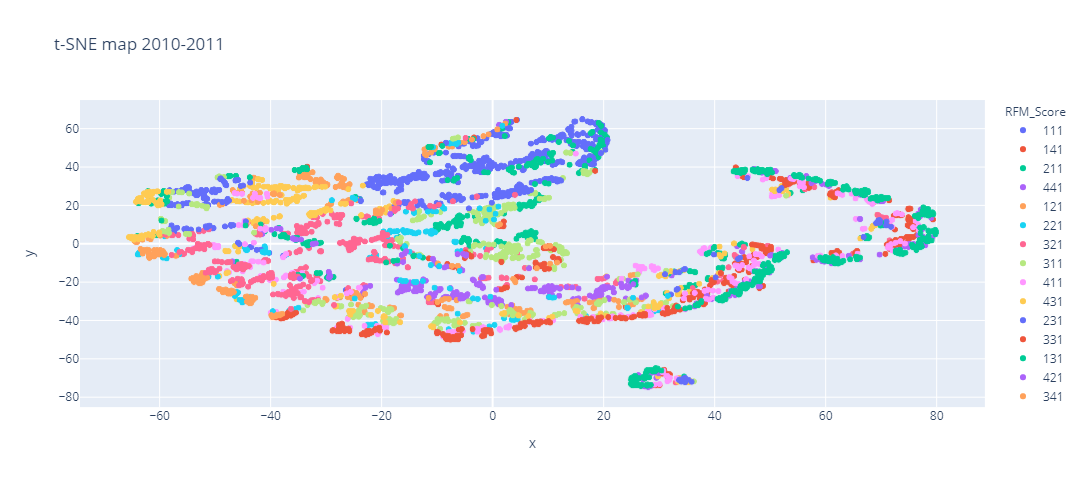




Least Valuable

Most Valuable





Least Valuable

Most Valuable

The visual performance of the 2D embeddings are very different depending on whether a discrete or continuous colouring system was used. In the 2009-2010 map, there are 57 unique colours of points, each representing a different RFM score and in the 2010-2011 map, there are 58 colours. This means that the first year is missing 7 possible RFM scores and the second year is missing 6, since there are 64 possible RFM scores in total. Because the discrete colour embedding uses very distinct colours for similar RFM scores, the embedding does not look very good. This was likely caused because the features were separated into scores at their quartiles, namely that while t-SNE interprets a pair of 3 dimensional data points as similar, a different RFM score is given to them. In this case, local structure is arguably preserved but the embedding does not support that. For example in the discrete 2009-2010 map, the cluster of RFM scores 111 (blue) and 211 (purple) circled in black indicate similar customers which t-SNE is picking up on, but should belong in different clusters because they have different RFM score. Additionally, the cluster of red points indicated with a blue circle all have an RFM score of 222. However, they appear to be somewhat separated from each other which could mean that t-SNE may interpret these points as not very similar while they have identical RFM scores. The small gaps of separations of points scattered throughout the discrete maps shows that t-SNE is trying to preserve local structure as much as possible.

# Conclusion

This report has shown how specific real-world retail data columns can be related to the RFM model, which allowed an RFM score to be assigned to each customer. PCA was then used to reduce the three features of RFM to two, using ideas of variance of features, the covariance matrix and its eigenvectors and corresponding eigenvalues. This gave a two dimensional visualisation of the data, which was compared to that of t-SNE. Both methods visually appear to perform similarly in that continuous colour plots are better than discrete colour plots as they show a more gradual colour change between similar RFM scores. This can possibly mean that too many clusters (different RFM scores) was used in the model so the clusters could not be separate too well. This also shows that when there is no structure in your original data to begin with, the two dimensional embeddings may not show clearly separated clusters.

# References

Alkhayrat, M., Aljnidi, M. and Aljoumaa, K. (2020) ‘A comparative dimensionality reduction study in telecom customer segmentation using deep learning and PCA’, *Journal of Big Data*, 7(1). doi: 10.1186/s40537-020-0286-0.

Bandyopadhyay, S., Thakur, S. S. and Mandal, J. K. (2020) ‘Product recommendation for e-commerce business by applying principal component analysis (PCA) and K-means clustering: benefit for the society’, *Innovations in Systems and Software Engineering*, 17(1), pp. 45–52. doi: 10.1007/s11334-020-00372-5.

Chen, I. J. and Popovich, K. (2003) ‘Understanding customer relationship management (CRM): People, process and technology’, *Business Process Management Journal*. doi: 10.1108/14637150310496758.

Christy, A. J. *et al.* (2018) ‘RFM ranking – An effective approach to customer segmentation’, *Journal of King Saud University - Computer and Information Sciences*. doi: 10.1016/j.jksuci.2018.09.004.

Dua, Dheeru; Graff, C. (2019) *UCI Machine Learning Repository*, *Center for Machine Learning and Intelligent Systems*. Available at: https://archive.ics.uci.edu/ml/datasets/Online+Retail+II#.

Dursun, A. and Caber, M. (2016) ‘Using data mining techniques for profiling profitable hotel customers: An application of RFM analysis’, *Tourism Management Perspectives*. doi: 10.1016/j.tmp.2016.03.001.

Galarnyk, M. (2017) ‘PCA using Python (scikit-learn)’, *PCA using Python (scikit-learn)*. Available at: https://towardsdatascience.com/pca-using-python-scikit-learn-e653f8989e60.

García-Alonso, C. R., Pérez-Naranjo, L. M. and Fernández-Caballero, J. C. (2014) ‘Multiobjective evolutionary algorithms to identify highly autocorrelated areas: The case of spatial distribution in financially compromised farms’, *Annals of Operations Research*, 219(1), pp. 187–202. doi: 10.1007/s10479-011-0841-3.

García-Murillo, M. and Annabi, H. (2002) ‘Customer knowledge management’, *Journal of the Operational Research Society*. doi: 10.1057/palgrave.jors.2601365.

Gebert, H. *et al.* (2003) ‘Knowledge-enabled customer relationship management: Integrating customer relationship management and knowledge management concepts[1]’, *Journal of Knowledge Management*. doi: 10.1108/13673270310505421.

Granato, D. *et al.* (2018) ‘Use of principal component analysis (PCA) and hierarchical cluster analysis (HCA) for multivariate association between bioactive compounds and functional properties in foods: A critical perspective’, *Trends in Food Science and Technology*, 72(December 2017), pp. 83–90. doi: 10.1016/j.tifs.2017.12.006.

Kansal, T. *et al.* (2018) ‘Customer Segmentation using K-means Clustering’, in *Proceedings of the International Conference on Computational Techniques, Electronics and Mechanical Systems, CTEMS 2018*. doi: 10.1109/CTEMS.2018.8769171.

Khajvand, M. and Tarokh, M. J. (2011) ‘Estimating customer future value of different customer segments based on adapted RFM model in retail banking context’, in *Procedia Computer Science*. doi: 10.1016/j.procs.2011.01.011.

Khodakarami, F. and Chan, Y. E. (2014) ‘Exploring the role of customer relationship management (CRM) systems in customer knowledge creation’, *Information and Management*. doi: 10.1016/j.im.2013.09.001.

Lateef, Z. (2020) *All You Need To Know About Principal Component Analysis (PCA)*, *All You Need To Know About Principal Component Analysis (PCA)*. Available at: https://www.edureka.co/blog/principal-component-analysis/.

Liu, H. *et al.* (2021) ‘Using t-distributed Stochastic Neighbor Embedding (t-SNE) for cluster analysis and spatial zone delineation of groundwater geochemistry data’, *Journal of Hydrology*, 597(December 2020), p. 126146. doi: 10.1016/j.jhydrol.2021.126146.

Luo, Y. *et al.* (2012) ‘Telecom customer segmentation with K-means clustering’, in *ICCSE 2012 - Proceedings of 2012 7th International Conference on Computer Science and Education*. doi: 10.1109/ICCSE.2012.6295158.

Maklin, C. (2019) ‘Principal Component Analysis Example In Python’, *Principal Component Analysis Example In Python*. Available at: https://towardsdatascience.com/machine-learning-part-15-dimensionality-reduction-with-principal-component-analysis-a5b3bb7353bc.

Margaritis, A. *et al.* (2020) ‘Identification of ageing state clusters of reclaimed asphalt binders using principal component analysis (PCA) and hierarchical cluster analysis (HCA) based on chemo-rheological parameters’, *Construction and Building Materials*, 244, p. 118276. doi: 10.1016/j.conbuildmat.2020.118276.

Maryani, I. and Riana, D. (2017) ‘Clustering and profiling of customers using RFM for customer relationship management recommendations’, in *2017 5th International Conference on Cyber and IT Service Management, CITSM 2017*. doi: 10.1109/CITSM.2017.8089258.

Navlani, A. (2018) *Introduction to Customer Segmentation in Python*, *Introduction to Customer Segmentation in Python*. Available at: https://www.datacamp.com/community/tutorials/introduction-customer-segmentation-python.

Ngai, E. W. T., Xiu, L. and Chau, D. C. K. (2009) ‘Application of data mining techniques in customer relationship management: A literature review and classification’, *Expert Systems with Applications*. doi: 10.1016/j.eswa.2008.02.021.

Payne, A. and Frow, P. (2005) ‘A strategic framework for customer relationship management’, *Journal of Marketing*. doi: 10.1509/jmkg.2005.69.4.167.

Pezzotti, N. *et al.* (2017) ‘Approximated and user steerable tSNE for progressive visual analytics’, *IEEE Transactions on Visualization and Computer Graphics*, 23(7), pp. 1739–1752. doi: 10.1109/TVCG.2016.2570755.

Rust, R. T. and Zahorik, A. J. (1993) ‘Customer satisfaction, customer retention, and market share’, *Journal of Retailing*. doi: 10.1016/0022-4359(93)90003-2.

Sparks, U. (2019) *A Python documentation website*, *A Python documentation website*. Available at: https://ugoproto.github.io/ugo\_py\_doc/.

Srisamran, P. and Ractham, V. V. (2014) ‘Customer-centric knowledge creation for customer relationship management’, *Journal of Applied Business Research*. doi: 10.19030/jabr.v30i2.8410.

Wu, X. *et al.* (2008) ‘Top 10 algorithms in data mining’, *Knowledge and Information Systems*. doi: 10.1007/s10115-007-0114-2.

# Appendix

RFM:

# Import modules  
  
import pandas as pd # for dataframes  
import matplotlib.pyplot as plt # for plotting graphs  
import datetime as dt  
  
pd.set\_option('display.max\_columns', None)  
pd.set\_option('display.max\_rows', None)  
  
# 2009-2010 #########################################################################################  
  
data\_2009\_2010 = pd.read\_excel("Online\_Retail\_II\_2009\_2010.xlsx")  
  
print(data\_2009\_2010.head())  
  
# Omit rows where there is no Customer ID  
  
data\_2009\_2010 = data\_2009\_2010[pd.notnull(data\_2009\_2010['Customer ID'])]  
  
# Remove duplicates  
  
filtered\_data\_2009\_2010 = data\_2009\_2010[['Country', 'Customer ID']].drop\_duplicates()  
  
# Show customers based on their country of residence  
  
filtered\_data\_2009\_2010.Country.value\_counts()[:10].plot(kind='bar')  
plt.show()  
  
# In the given dataset, it can be observed that most countries are UK  
# So we can filter data for UK customers  
  
uk\_data\_2009\_2010 = data\_2009\_2010[data\_2009\_2010.Country == 'United Kingdom']  
print(uk\_data\_2009\_2010.describe())  
  
# Omit rows where the quantity of items bought is negative since this is impossible  
  
uk\_data\_2009\_2010 = uk\_data\_2009\_2010[(uk\_data\_2009\_2010['Quantity'] > 0)]  
  
# Filter necessary columns for RFM  
  
uk\_data\_2009\_2010 = uk\_data\_2009\_2010[['Customer ID', 'InvoiceDate', 'Invoice', 'Quantity', 'Price']]  
  
# Add required columns  
  
uk\_data\_2009\_2010['Total Price'] = uk\_data\_2009\_2010['Quantity'] \* uk\_data\_2009\_2010['Price']  
  
uk\_data\_2009\_2010['InvoiceDate'].min(), uk\_data\_2009\_2010['InvoiceDate'].max()  
  
PRESENT\_2009\_2010 = dt.datetime(2010, 12, 10)  
  
uk\_data\_2009\_2010['InvoiceDate'] = pd.to\_datetime(uk\_data\_2009\_2010['InvoiceDate'])  
  
print(uk\_data\_2009\_2010.head())  
  
# Calculate the RFM features  
  
rfm\_2009\_2010 = uk\_data\_2009\_2010.groupby('Customer ID').agg({'InvoiceDate': lambda date: (PRESENT\_2009\_2010 -  
 date.max()).days,  
 'Invoice': lambda num: len(num),  
 'Total Price': lambda price: price.sum()})  
  
# Change the name of columns  
  
rfm\_2009\_2010.columns = ['recency', 'frequency', 'monetary']  
  
rfm\_2009\_2010['recency'] = rfm\_2009\_2010['recency'].astype(int)  
  
print(rfm\_2009\_2010.head())  
  
# Customers with the lowest recency, highest frequency and highest monetary amounts considered most valuable  
  
rfm\_2009\_2010['r\_quartile'] = pd.qcut(rfm\_2009\_2010['recency'], 4, ['1', '2', '3', '4'])  
rfm\_2009\_2010['f\_quartile'] = pd.qcut(rfm\_2009\_2010['frequency'], 4, ['4', '3', '2', '1'])  
rfm\_2009\_2010['m\_quartile'] = pd.qcut(rfm\_2009\_2010['monetary'], 4, ['4', '3', '2', '1'])  
  
print(rfm\_2009\_2010.head())  
  
rfm\_2009\_2010['RFM\_Score'] = rfm\_2009\_2010.r\_quartile.astype(str) + rfm\_2009\_2010.f\_quartile.astype(str) \  
 + rfm\_2009\_2010.m\_quartile.astype(str)  
print(rfm\_2009\_2010.head())  
  
# Filter out best customers  
  
print(rfm\_2009\_2010[rfm\_2009\_2010['RFM\_Score'] == '111'].sort\_values('monetary', ascending=False).head())  
  
# Make CSV files of customers with score 111 and all customers in descending monetary order  
  
# rfm\_111\_2009\_2010 = rfm\_2009\_2010[rfm\_2009\_2010['RFM\_Score'] == '111']  
# rfm\_111\_2009\_2010.to\_csv("2009-2010 Customers with RFM Score 111")  
#  
# sorted\_by\_monetary = rfm\_2009\_2010.sort\_values('monetary', ascending=False)  
# sorted\_by\_monetary.to\_csv('2009-2010 Customers Sorted by Monetary Value')  
  
# 2010-2011 #########################################################################################  
  
data\_2010\_2011 = pd.read\_excel("Online\_Retail\_II\_2010\_2011.xlsx")  
  
print(data\_2010\_2011.head())  
  
# Omit rows where Customer ID is empty  
  
data\_2010\_2011 = data\_2010\_2011[pd.notnull(data\_2010\_2011['Customer ID'])]  
  
# Remove duplicates  
  
filtered\_data\_2010\_2011 = data\_2010\_2011[['Country', 'Customer ID']].drop\_duplicates()  
  
# Bar chart showing customers based on country  
  
filtered\_data\_2010\_2011.Country.value\_counts()[:10].plot(kind='bar')  
plt.show()  
  
# In the given dataset, most customers are from the UK   
# So filter data for UK customers  
  
uk\_data\_2010\_2011 = data\_2010\_2011[data\_2010\_2011.Country == 'United Kingdom']  
print(uk\_data\_2010\_2011.describe())  
  
# Filter impossible negative quantities  
  
uk\_data\_2010\_2011 = uk\_data\_2010\_2011[(uk\_data\_2010\_2011['Quantity'] > 0)]  
  
# Filter required columns for RFM  
  
uk\_data\_2010\_2011 = uk\_data\_2010\_2011[['Customer ID', 'InvoiceDate', 'Invoice', 'Quantity', 'Price']]  
  
# Add required columns  
  
uk\_data\_2010\_2011['Total Price'] = uk\_data\_2010\_2011['Quantity'] \* uk\_data\_2010\_2011['Price']  
  
uk\_data\_2010\_2011['InvoiceDate'].min(), uk\_data\_2010\_2011['InvoiceDate'].max()  
  
PRESENT\_2010\_2011 = dt.datetime(2011, 12, 10)  
  
uk\_data\_2010\_2011['InvoiceDate'] = pd.to\_datetime(uk\_data\_2010\_2011['InvoiceDate'])  
  
print(uk\_data\_2010\_2011.head())

# Calculate the RFM features  
  
rfm\_2010\_2011 = uk\_data\_2010\_2011.groupby('Customer ID').agg({'InvoiceDate': lambda date: (PRESENT\_2010\_2011 -  
 date.max()).days,  
 'Invoice': lambda num: len(num),  
 'Total Price': lambda price: price.sum()})  
  
# Change the name of columns  
  
rfm\_2010\_2011.columns = ['recency', 'frequency', 'monetary']  
  
rfm\_2010\_2011['recency'] = rfm\_2010\_2011['recency'].astype(int)  
  
print(rfm\_2010\_2011.head())  
  
# Customers with the lowest recency, highest frequency and highest monetary amounts considered as most valuable  
  
rfm\_2010\_2011['r\_quartile'] = pd.qcut(rfm\_2010\_2011['recency'], 4, ['1', '2', '3', '4'])  
rfm\_2010\_2011['f\_quartile'] = pd.qcut(rfm\_2010\_2011['frequency'], 4, ['4', '3', '2', '1'])  
rfm\_2010\_2011['m\_quartile'] = pd.qcut(rfm\_2010\_2011['monetary'], 4, ['4', '3', '2', '1'])  
  
print(rfm\_2010\_2011.head())  
  
rfm\_2010\_2011['RFM\_Score'] = rfm\_2010\_2011.r\_quartile.astype(str) + rfm\_2010\_2011.f\_quartile.astype(str) \  
 + rfm\_2010\_2011.m\_quartile.astype(str)  
print(rfm\_2010\_2011.head())  
  
# Filter out best customers  
  
print(rfm\_2010\_2011[rfm\_2010\_2011['RFM\_Score'] == '111'].sort\_values('monetary', ascending=False).head())  
  
# Make CSV files of customers with score 111 and all customers in descending monetary order  
  
# rfm\_111\_2010\_2011 = rfm\_2010\_2011[rfm\_2010\_2011['RFM\_Score'] == '111']  
# rfm\_111\_2010\_2011.to\_csv("2010-2011 Customers with RFM Score 111")  
#  
# sorted\_by\_monetary\_2010\_2011 = rfm\_2010\_2011.sort\_values('monetary', ascending=False)  
# sorted\_by\_monetary\_2010\_2011.to\_csv('2010-2011 Customers Sorted by Monetary Value')  
  
# Box plots  
  
red\_circle = dict(markerfacecolor='r')  
plt.boxplot(rfm\_2009\_2010["recency"], vert=False, flierprops=red\_circle)  
plt.title("Boxplot showing Recency from 2009-2010")  
plt.show()  
  
plt.boxplot(rfm\_2010\_2011["recency"], vert=False, flierprops=red\_circle)  
plt.title("Boxplot showing Recency from 2010-2011")  
plt.show()  
  
plt.boxplot(rfm\_2009\_2010["frequency"], vert=False, flierprops=red\_circle)  
plt.title("Boxplot showing Frequency from 2009-2010")  
plt.show()  
  
plt.boxplot(rfm\_2010\_2011["frequency"], vert=False, flierprops=red\_circle)  
plt.title("Boxplot showing Frequency from 2010-2011")  
plt.show()  
  
plt.boxplot(rfm\_2009\_2010["monetary"], vert=False, flierprops=red\_circle)  
plt.title("Boxplot showing Monetary from 2009-2010")  
plt.show()  
  
plt.boxplot(rfm\_2010\_2011["monetary"], vert=False, flierprops=red\_circle)  
plt.title("Boxplot showing Monetary from 2010-2011")  
plt.show()

PCA:

import pandas as pd  
import numpy as np  
from sklearn.preprocessing import StandardScaler  
from sklearn.decomposition import PCA  
import matplotlib.pyplot as plt  
import plotly.express as px  
  
pd.set\_option('display.max\_columns', None)  
pd.set\_option('display.max\_rows', None)  
  
# 2009-2010 #####################################################################################  
  
df\_2009\_2010 = pd.read\_excel("2009-2010 Customers Sorted by Monetary Value.xlsx",  
 names=["Customer ID", "recency", "frequency", "monetary",  
 "r\_quartile", "f\_quartile", "m\_quartile", "RFM\_Score"])  
  
features\_2009\_2010 = ["recency", "frequency", "monetary"]  
  
print(df\_2009\_2010.describe())  
  
# The following scatter plots produced are included in the appendix  
fig1 = px.scatter(df\_2009\_2010,  
 x="recency", y="frequency",  
 color="RFM\_Score", size="RFM\_Score", opacity=1, title="Recency vs Frequency 2009-2010")  
fig1.show()  
  
fig2 = px.scatter(df\_2009\_2010,  
 x="recency", y="monetary",  
 color="RFM\_Score", size="RFM\_Score", opacity=1, title="Recency vs Monetary 2009-2010")  
fig2.show()  
  
fig3 = px.scatter(df\_2009\_2010,  
 x="frequency", y="monetary",  
 color="RFM\_Score", size="RFM\_Score", opacity=1, title="Frequency vs Monetary 2009-2010")  
fig3.show()  
  
# Separating features  
x\_2009\_2010 = df\_2009\_2010.loc[:, features\_2009\_2010].values  
# Separating target  
y\_2009\_2010 = df\_2009\_2010.loc[:, ["RFM\_Score"]].values  
print(y\_2009\_2010)  
# Standardizing features  
x\_2009\_2010 = StandardScaler().fit\_transform(x\_2009\_2010)  
print(x\_2009\_2010)  
  
# Covariance Matrix  
mean\_vec\_2009\_2010 = np.mean(x\_2009\_2010, axis=0)  
cov\_mat\_2009\_2010 = (x\_2009\_2010 - mean\_vec\_2009\_2010).T.dot((x\_2009\_2010 - mean\_vec\_2009\_2010)) \  
 / (x\_2009\_2010.shape[0]-1)  
print("Covariance Matrix 2009-2010 \n%s" % cov\_mat\_2009\_2010)  
  
# Eigendecomposition  
cov\_mat\_2009\_2010 = np.cov(x\_2009\_2010.T)  
eig\_vals\_2009\_2010, eig\_vecs\_2009\_2010 = np.linalg.eig(cov\_mat\_2009\_2010)  
print("Eigenvectors \n%s" % eig\_vecs\_2009\_2010)  
print("Eigenvalues \n%s" % eig\_vals\_2009\_2010)  
  
eig\_pairs\_2009\_2010 = [(np.abs(eig\_vals\_2009\_2010[i]), eig\_vecs\_2009\_2010[:, i])  
 for i in range(len(eig\_vals\_2009\_2010))]  
print("Eigenvalues in descending order:")  
for i in eig\_pairs\_2009\_2010:  
 print(i[0])  
  
pca\_2009\_2010 = PCA(n\_components=3)  
  
principalComponents\_2009\_2010 = pca\_2009\_2010.fit\_transform(x\_2009\_2010)  
  
print(pca\_2009\_2010.explained\_variance\_ratio\_)  
  
principalDf\_2009\_2010 = pd.DataFrame(data=principalComponents\_2009\_2010  
 , columns=['PC1', 'PC2', 'PC3'])  
  
finalDf\_2009\_2010 = pd.concat([principalDf\_2009\_2010, df\_2009\_2010[['RFM\_Score']]], axis=1)  
print(finalDf\_2009\_2010)  
  
per\_variable\_2009\_2010 = np.round(pca\_2009\_2010.explained\_variance\_ratio\_ \* 100, decimals=1)  
labels = ["PC" + str(x) for x in range(1, len(per\_variable\_2009\_2010) + 1)]  
  
loadings\_2009\_2010 = pca\_2009\_2010.components\_.T  
df\_loadings\_2009\_2010 = pd.DataFrame(loadings\_2009\_2010, columns=["PC1", "PC2", "PC3"],  
 index=["recency", "frequency", "monetary"])  
print(df\_loadings\_2009\_2010)  
  
loadings\_label = df\_loadings\_2009\_2010.index  
  
fig4 = px.scatter(df\_loadings\_2009\_2010, x="PC1", y="PC2", text=loadings\_label)  
  
fig4.show()  
  
plt.scatter(df\_loadings\_2009\_2010.PC1, y=df\_loadings\_2009\_2010.PC2)  
plt.xlabel("PC1")  
plt.ylabel("PC2")  
plt.show()  
  
# Scree plot  
  
plt.bar(x=range(1, len(per\_variable\_2009\_2010) + 1), height=per\_variable\_2009\_2010, tick\_label=labels)  
plt.ylabel("Percentage of Explained Variance")  
plt.xlabel("Principal Component")  
plt.title("Scree Plot 2009-2010")  
plt.show()  
  
# PC plot of first 2 PCs with most variation  
  
plt.scatter(finalDf\_2009\_2010.PC1, finalDf\_2009\_2010.PC2, c=finalDf\_2009\_2010.RFM\_Score)  
plt.xlabel("PC1")  
plt.ylabel("PC2")  
plt.title("PC Scores 2009-2010")  
plt.show()  
  
fig5 = px.scatter(finalDf\_2009\_2010,  
 x="PC1", y="PC2",  
 color="RFM\_Score", size="RFM\_Score", opacity=1, title="PCA Plot 2009-2010")  
fig5.show()  
  
finalDf\_2009\_2010["RFM\_Score"] = finalDf\_2009\_2010["RFM\_Score"].astype(str)  
fig6 = px.scatter(finalDf\_2009\_2010,  
 x="PC1", y="PC2",  
 color="RFM\_Score", opacity=1, title="PCA Plot 2009-2010")  
fig6.show()  
  
# 2010-2011 #####################################################################################  
  
df\_2010\_2011 = pd.read\_excel("2010-2011 Customers Sorted by Monetary Value.xlsx",  
 names=["Customer ID", "recency", "frequency", "monetary",  
 "r\_quartile", "f\_quartile", "m\_quartile", "RFM\_Score"])  
  
features\_2010\_2011 = ["recency", "frequency", "monetary"]  
print(df\_2010\_2011.describe())  
  
# The following scatter plots produced are included in the appendix  
fig7 = px.scatter(df\_2010\_2011,  
 x="recency", y="frequency",  
 color="RFM\_Score", size="RFM\_Score", opacity=1, title="Recency vs Frequency 2010-2011")  
fig7.show()  
  
fig8 = px.scatter(df\_2010\_2011,  
 x="recency", y="monetary",  
 color="RFM\_Score", size="RFM\_Score", opacity=1, title="Recency vs Monetary 2010-2011")  
fig8.show()  
  
fig9 = px.scatter(df\_2010\_2011,  
 x="frequency", y="monetary",  
 color="RFM\_Score", size="RFM\_Score", opacity=1, title="Frequency vs Monetary 2010-2011")  
fig9.show()  
  
# Separating features  
x\_2010\_2011 = df\_2010\_2011.loc[:, features\_2010\_2011].values  
# Separating target  
y\_2010\_2011 = df\_2010\_2011.loc[:, ["RFM\_Score"]].values  
print(y\_2010\_2011)  
# Standardizing features  
x\_2010\_2011 = StandardScaler().fit\_transform(x\_2010\_2011)  
print(x\_2010\_2011)  
  
# Covariance Matrix  
mean\_vec\_2010\_2011 = np.mean(x\_2010\_2011, axis=0)  
cov\_mat\_2010\_2011 = (x\_2010\_2011 - mean\_vec\_2010\_2011).T.dot((x\_2010\_2011 - mean\_vec\_2010\_2011)) \  
 / (x\_2010\_2011.shape[0]-1)  
print("Covariance Matrix 2010-2011 \n%s" % cov\_mat\_2010\_2011)  
  
# Eigendecomposition  
cov\_mat\_2010\_2011 = np.cov(x\_2010\_2011.T)  
eig\_vals\_2010\_2011, eig\_vecs\_2010\_2011 = np.linalg.eig(cov\_mat\_2010\_2011)  
print("Eigenvectors \n%s" % eig\_vecs\_2010\_2011)  
print("Eigenvalues \n%s" % eig\_vals\_2010\_2011)  
  
eig\_pairs\_2010\_2011 = [(np.abs(eig\_vals\_2010\_2011[i]), eig\_vecs\_2010\_2011[:, i])  
 for i in range(len(eig\_vals\_2010\_2011))]  
print("Eigenvalues in descending order:")  
for i in eig\_pairs\_2010\_2011:  
 print(i[0])  
  
pca\_2010\_2011 = PCA(n\_components=3)  
  
principalComponents\_2010\_2011 = pca\_2010\_2011.fit\_transform(x\_2010\_2011)  
  
print(pca\_2010\_2011.explained\_variance\_ratio\_)  
  
principalDf\_2010\_2011 = pd.DataFrame(data=principalComponents\_2010\_2011  
 , columns=['PC1', 'PC2', 'PC3'])  
  
finalDf\_2010\_2011 = pd.concat([principalDf\_2010\_2011, df\_2010\_2011[['RFM\_Score']]], axis=1)  
print(finalDf\_2010\_2011)  
  
per\_variable\_2010\_2011 = np.round(pca\_2010\_2011.explained\_variance\_ratio\_ \* 100, decimals=1)  
labels = ["PC" + str(x) for x in range(1, len(per\_variable\_2010\_2011) + 1)]  
  
loadings\_2010\_2011 = pca\_2010\_2011.components\_.T  
df\_loadings\_2010\_2011 = pd.DataFrame(loadings\_2010\_2011, columns=["PC1", "PC2", "PC3"],  
 index=["recency", "frequency", "monetary"])  
print(df\_loadings\_2010\_2011)  
  
loadings\_label = df\_loadings\_2010\_2011.index  
  
fig10 = px.scatter(df\_loadings\_2010\_2011, x="PC1", y="PC2", text=loadings\_label)  
  
fig10.show()  
  
plt.scatter(df\_loadings\_2010\_2011.PC1, y=df\_loadings\_2010\_2011.PC2)  
plt.xlabel("PC1")  
plt.ylabel("PC2")  
plt.show()  
  
# Scree plot  
  
plt.bar(x=range(1, len(per\_variable\_2010\_2011) + 1), height=per\_variable\_2010\_2011, tick\_label=labels)  
plt.ylabel("Percentage of Explained Variance")  
plt.xlabel("Principal Component")  
plt.title("Scree Plot 2010-2011")  
plt.show()  
  
# PC plot of first 2 PCs with most variation  
  
plt.scatter(finalDf\_2010\_2011.PC1, finalDf\_2010\_2011.PC2, c=finalDf\_2010\_2011.RFM\_Score)  
  
plt.xlabel("PC1")  
plt.ylabel("PC2")  
plt.title("PC Scores 2010-2011")  
plt.show()  
  
fig11 = px.scatter(finalDf\_2010\_2011,  
 x="PC1", y="PC2",  
 color="RFM\_Score", size="RFM\_Score", opacity=1, title="PCA Plot 2010-2011")  
fig11.show()  
  
finalDf\_2010\_2011["RFM\_Score"] = finalDf\_2010\_2011["RFM\_Score"].astype(str)  
fig12 = px.scatter(finalDf\_2010\_2011,  
 x="PC1", y="PC2",  
 color="RFM\_Score", opacity=1, title="PCA Plot 2010-2011")  
fig12.show()

t-SNE:

# Import modules  
  
from sklearn.manifold import TSNE  
from sklearn.preprocessing import StandardScaler  
import pandas as pd  
import seaborn as sns  
import matplotlib.pyplot as plt  
import plotly.express as px  
  
pd.set\_option('display.max\_columns', None)  
pd.set\_option('display.max\_rows', None)  
  
# 2009-2010 ########################################################  
  
df\_2009\_2010 = pd.read\_excel("2009-2010 Customers Sorted by Monetary Value.xlsx",  
 names=["Customer ID", "recency", "frequency", "monetary",  
 "r\_quartile", "f\_quartile", "m\_quartile", "RFM\_Score"])  
  
df\_2009\_2010\_features = df\_2009\_2010[["recency", "frequency", "monetary"]]  
print(df\_2009\_2010\_features)  
  
df\_2009\_2010\_RFM = df\_2009\_2010["RFM\_Score"]  
  
scaled\_data\_2009\_2010 = StandardScaler().fit\_transform(df\_2009\_2010\_features)  
print(scaled\_data\_2009\_2010)  
  
model\_2009\_2010 = TSNE(n\_components=2, random\_state=0, perplexity=30, learning\_rate=200)  
  
tsne\_features\_2009\_2010 = model\_2009\_2010.fit\_transform(scaled\_data\_2009\_2010)  
  
print(tsne\_features\_2009\_2010)  
  
x\_2009\_2010 = tsne\_features\_2009\_2010[:, 0]  
y\_2009\_2010 = tsne\_features\_2009\_2010[:, 1]  
  
sns.scatterplot(x=x\_2009\_2010, y=y\_2009\_2010, c=df\_2009\_2010.RFM\_Score)  
plt.show()  
  
df1 = pd.DataFrame(tsne\_features\_2009\_2010)  
  
finalDf\_2009\_2010 = pd.concat([df1, df\_2009\_2010[['RFM\_Score']]], axis=1)  
  
finalDf\_2009\_2010["RFM\_Score"] = finalDf\_2009\_2010["RFM\_Score"].astype(str)  
fig1 = px.scatter(finalDf\_2009\_2010,  
 x=x\_2009\_2010, y=y\_2009\_2010,  
 color="RFM\_Score", opacity=1, title="t-SNE map 2009-2010")  
fig1.show()  
  
fig2 = px.scatter(df\_2009\_2010,  
 x=x\_2009\_2010, y=y\_2009\_2010,  
 color="RFM\_Score", size="RFM\_Score", opacity=1, title="t-SNE map 2009-2010")  
fig2.show()  
  
# 2010-2011 ########################################################  
  
df\_2010\_2011 = pd.read\_excel("2010-2011 Customers Sorted by Monetary Value.xlsx",  
 names=["Customer ID", "recency", "frequency", "monetary",  
 "r\_quartile", "f\_quartile", "m\_quartile", "RFM\_Score"])  
  
df\_2010\_2011\_features = df\_2010\_2011[["recency", "frequency", "monetary"]]  
print(df\_2010\_2011\_features)  
  
df\_2010\_2011\_RFM = df\_2010\_2011["RFM\_Score"]  
  
scaled\_data\_2010\_2011 = StandardScaler().fit\_transform(df\_2010\_2011\_features)  
print(scaled\_data\_2010\_2011)  
  
model\_2010\_2011 = TSNE(n\_components=2, random\_state=0, perplexity=30, learning\_rate=200)  
# Consider learning rate  
tsne\_features\_2010\_2011 = model\_2010\_2011.fit\_transform(scaled\_data\_2010\_2011)  
  
print(tsne\_features\_2010\_2011)  
  
x\_2010\_2011 = tsne\_features\_2010\_2011[:, 0]  
y\_2010\_2011 = tsne\_features\_2010\_2011[:, 1]  
  
sns.scatterplot(x=x\_2010\_2011, y=y\_2010\_2011, c=df\_2010\_2011.RFM\_Score)  
plt.show()  
  
df2 = pd.DataFrame(tsne\_features\_2010\_2011)  
  
finalDf\_2010\_2011 = pd.concat([df2, df\_2010\_2011[['RFM\_Score']]], axis=1)  
  
finalDf\_2010\_2011["RFM\_Score"] = finalDf\_2010\_2011["RFM\_Score"].astype(str)  
fig3 = px.scatter(finalDf\_2010\_2011,  
 x=x\_2010\_2011, y=y\_2010\_2011,  
 color="RFM\_Score", opacity=1, title="t-SNE map 2010-2011")  
fig3.show()  
  
fig4 = px.scatter(df\_2010\_2011,  
 x=x\_2010\_2011, y=y\_2010\_2011,  
 color="RFM\_Score", size="RFM\_Score", opacity=1, title="t-SNE map 2010-2011")  
fig4.show()

1. <https://scikit-learn.org/stable/modules/generated/sklearn.manifold.TSNE.html> [↑](#footnote-ref-1)
2. [https://scikit-learn.org/stable/modules/manifold.html#optimizing-t-sne](https://scikit-learn.org/stable/modules/manifold.html%23optimizing-t-sne) [↑](#footnote-ref-2)
3. <https://lvdmaaten.github.io/tsne/> [↑](#footnote-ref-3)