FIN42100: Machine Learning for Finance

Bank telemarketing and machine learning

Gavin Connolly (18308483), Sofia Emelianova (22205187),

Evan McGarry (22206023), Ajit Nambiyar (22200172)

Fanis-Filippos Papanikolaou (22200286)

Academic Year - 2022/23
Submitted: 23rd April, 2023

Word Count: 2996

Q1:

Perform and report exploratory data analytics in the data (e.g. visuals, descriptive statistics). State any observations which are pertinent to the purpose of this report, i.e., to inform a predictive model which can guide telephonic marketing.

To conduct exploratory analysis of the dataset, the data first needed to be cleaned up. This was achieved by first removing the 'duration' as requested in the footnotes of the outline, 'Ethnicity_African' variable due to violation of ethics and its lack of relevance to the study.
The summary statistics, mean, median, and standard deviation, for each of the numerical predictors was performed to give a better understanding of their dispersion.

	age	campaign	pdays	previous	emp.var.rate	cons.price.idx	cons.conf.idx	euribor3m	nr.employed	term_deposit
mean	40.02406	2.567593	962.475454	0.172963	0.081886	93.575664	-40.502600	3.621291	5167.035911	0.112654
median	38.00000	2.000000	999.000000	0.000000	1.100000	93.749000	-41.800000	4.857000	5191.000000	0.000000
std. dev	10.42125	2.770014	186.910907	0.494901	1.570960	0.578840	4.628198	1.734447	72.251528	0.316173

It is clear from the above that the variables are not of the same scale and may need to be standardized before certain statistical methods, such as clustering or PCA can be applied. It also gives us a better idea as to what our typical client looks like.
An examination of the categorical variables was then conducted, with the 'illiterate' education category being aggregated to the next closest category due to an insignificant number of observations. Due to the relative similarity in the distribution in the observations in the summer months (May-August), these were aggregated into a binary variable representing summer/winter. Finally, the 'education' variable was encoded as ordinal as there was a logically ordering to the categories & this would further reduce the number of required variables vs. one-hot encoding. These steps enabled a reduction in the dimension of the dataset, which was necessary given the large number of predictors present.

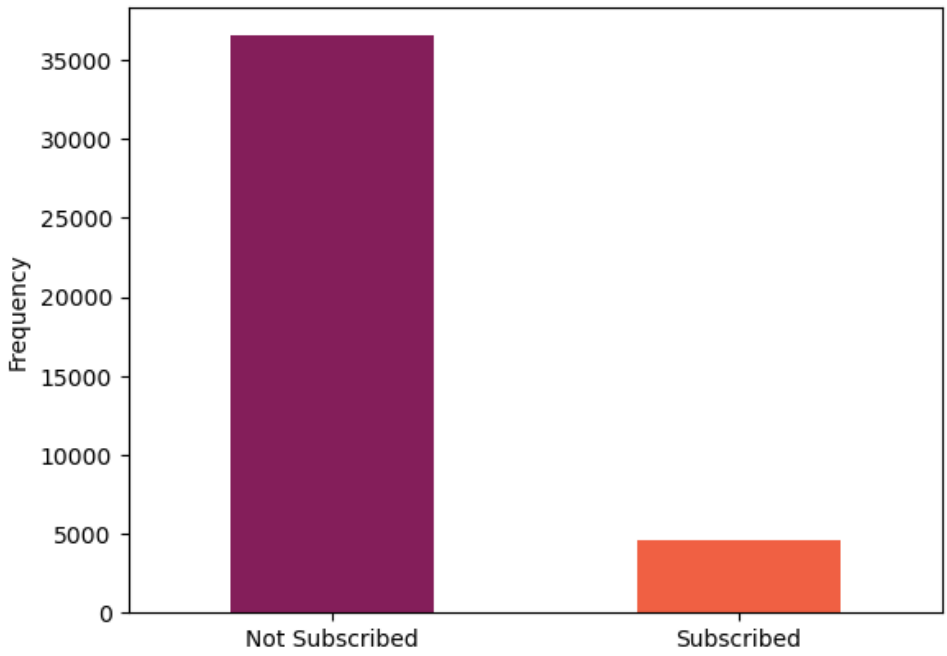


Figure 1 Class frequency for variable 'term_deposit'

From the above plot, it is evident that there exists a large class imbalance, with very few occurrences

of the event of interest. This imbalance means that metrics such as accuracy may not be optimal in evaluating the performance of our classification.

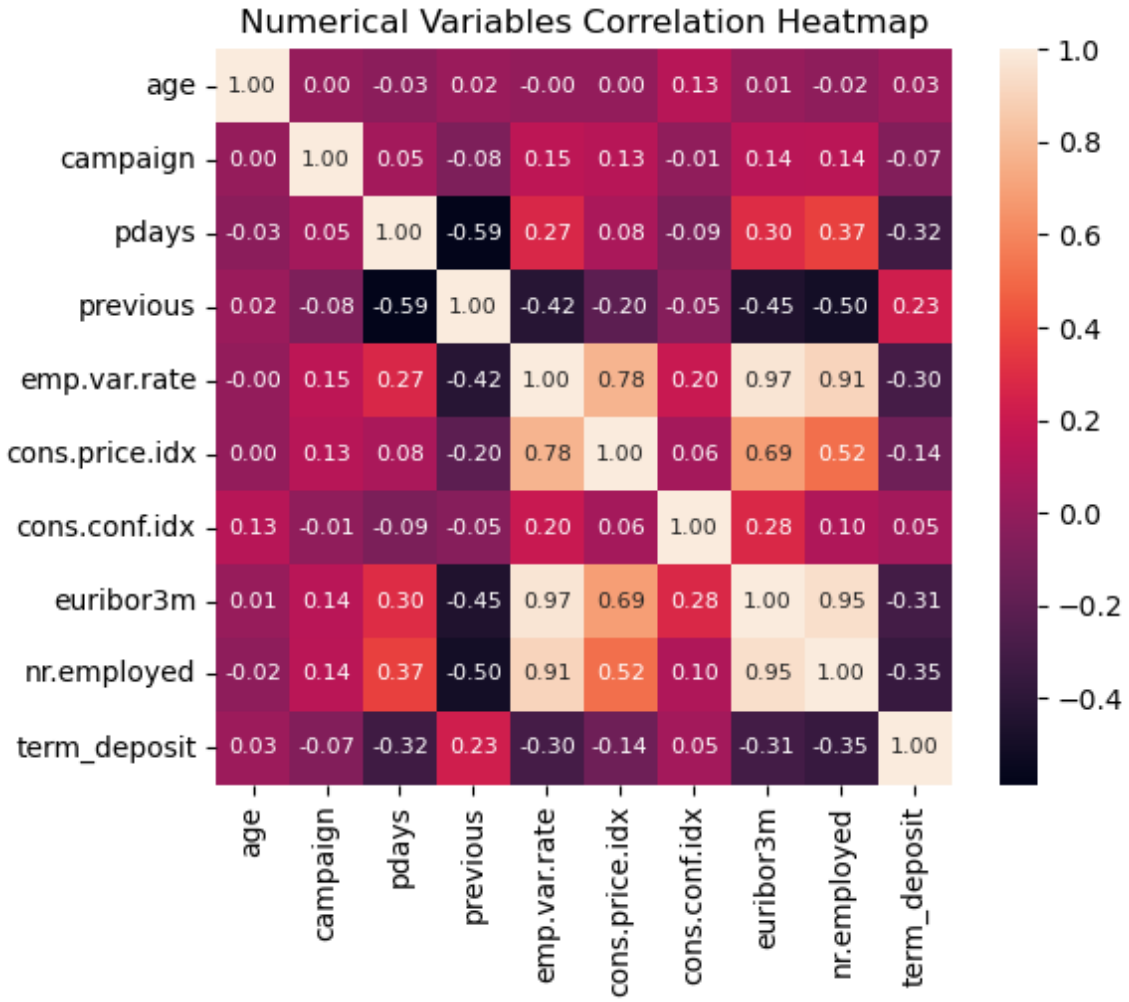


Figure 2 Correlation heatmap between the numerical predictor variables.

It can be seen from the heatmap that there is very low correlation between most of our predictor variables, with the main exception being the tendency for there to be a strong positive correlation between economic indicators such as the Euribor 3-month rate & employment variation rate, which had a correlation of 0.97. The variables most strongly correlated with the term deposit success variable were the number of employees, the number of days that passed by after the client was last contacted from a previous campaign, & the Euribor 3-month rate, which all had negative correlations with the variable of interest. This implies clients were more likely to subscribe to a term deposit account when they had few employees, they were contacted frequently & Euribor rates were low.

Q2:

a) Fit a logistic regression model on the dataset. Choose a probability threshold of 10%, 20%, 35% and 50%, to assign an observation to the Term Deposit = 1 class. Compute a confusion matrix for each of the probability thresholds. How do the true positive and false positive rates vary over these probability thresholds? Which probability threshold would you choose?

A logistic regression model was fit to the transformed variables as outlined above, yielding a set of 38 regression coefficients (the number of regression coefficients is inflated due to the high number of categorical variables).
The model yields the following confusion matrices for each of the given probability thresholds.

Threshold = 10%		Actual Subscribed	Actual Not Subscribed
Predicted	Subscribed	0.078057	0.210620
	Not Subscribed	0.034597	0.676726

Threshold = 20%		Actual Subscribed	Actual Not Subscribed
Predicted	Subscribed	0.058124	0.072934
	Not Subscribed	0.054530	0.814412

Threshold = 35%		Actual Subscribed	Actual Not Subscribed
Predicted	Subscribed	0.037851	0.031563
	Not Subscribed	0.074803	0.855783

Threshold = 50%		Actual Subscribed	Actual Not Subscribed
Predicted	Subscribed	0.024643	0.012528
	Not Subscribed	0.088011	0.874818

Some of the relevant model performance metrics are outlined in the following table:

Metric	Formula	10% Threshold	20% Threshold	35% Threshold	50% Threshold
Accuracy	TP+TN	75.5%	87.25%	89.35%	89.95%
Senitivity	$\frac{TP}{TP+FN}$	69.29%	51.6%	33.6%	21.87%
Specificity	$\frac{TP}{TP+FN}$	76.26%	91.78%	96.44%	98.59%

The desired result for the case of the bank in question, is to have a highly sensitive model (i.e., high percentage of subscribed users correctly predicted). An inverse relationship can be determined between the sensitivity and the threshold. When the threshold is reduced, the model becomes more sensitive, as observed by the increase in True-Positives. On the other hand, when the threshold is increased, there is an increasing trend in True-Negatives. Based off the calculated performance metrics, the optimal probability threshold given the classification objective is 10%.

b) Divide the dataset into training (70%) and test (30%) sets and repeat the above question and report the performance of these models, across probability thresholds, on the test set.

Threshold = 0.1		Actual	
		Subscribed	Not Subscribed
Predicted	Subscribed	0.081978	0.249899
	Not Subscribed	0.030671	0.637452

Threshold = 0.2		Actual	
		Subscribed	Not Subscribed
Predicted	Subscribed	0.057295	0.072105
	Not Subscribed	0.055353	0.815246

Threshold = 0.35		Actual	
		Subscribed	Not Subscribed
Predicted	Subscribed	0.030995	0.026786
	Not Subscribed	0.081654	0.860565

Threshold = 0.5		Actual	
		Subscribed	Not Subscribed
Predicted	Subscribed	0.022174	0.010844
	Not Subscribed	0.090475	0.876507

The same inverse relationship trend (as seen in part **a)**) can be observed in the ratios throughout the various thresholds. When the threshold decreases, TPR and FPR increase whereas when the threshold increases, then TNR & FNR increase. Thus, the best model is again the one where the probability threshold is set at 10%. The aim is to have both the TNR and TPR high and so, these ratios will serve as the main signals on which machine learning model algorithm outperforms the rest.

c) Plot the ROC for a logistic model on a graph and compute the AUC. Explain the information conveyed by the ROC and the AUC metrics

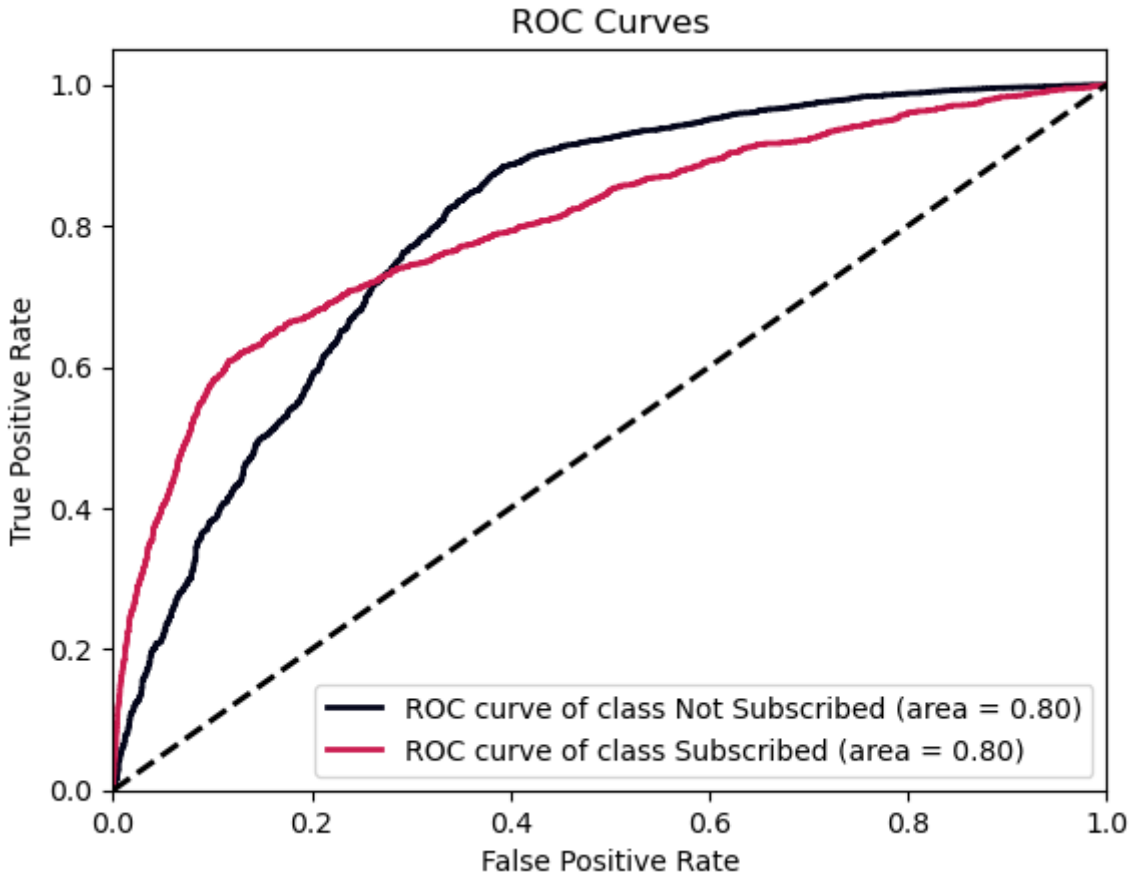


Figure 3 ROC Curve for the logistic regression model on the validation set.

By plotting the TPR and FPR at various classification thresholds, the Receiver Operator Characteristic (ROC) curve provides a visual representation of the trade-off between the true positive rate and the false positive rate for a given binary classifier. TPR and FPR are used to construct the ROC curve and AUC because they provide a more comprehensive evaluation of the classifier's performance.
A perfect binary classifier would have an AUC score of 1, indicating that it can perfectly distinguish between positive and negative samples (higher AUC value is considered to have better overall performance). The further left the point is, the better the model's trade-off between TRP & FPR is, with the broken diagonal line representing a model with predictive performance no better than random classification (corresponding AUC of 0.5).

Q3:

a) Fit three alternative machine learning models (e.g., classification tree, bagging, random forest models; KNN or SVM algorithms) in the data and comment on the performance of these models. Which model would you choose and why?

To compare the relative performance of these models we will look at 'recall' or 'sensitivity' as our main scoring strategy in evaluating the performance models. This is because 'false negatives' are more costly to the bank than 'false positives' and as such will be the focus on any performance measures. In addition to this we will look at the confusion matrices and ROC curves of each of these models to get an idea of the performance of each of the optimum cross-validated models for different probability thresholds.
The three machine learning models chosen to fit the data were:

1. Random Forest
2. XGBoost
3. Support Vector Machine

Hyperparameter tuning is a process which enables us to compare the performance of a ML model using various different inputs for the hyperparameters. This process was performed on each of the models, examining on the basis of AUC in order to attain the hyperparameters which yield the best trade-off between TPR & FPR, with the final probability threshold then chosen for our purposes based on the ROC Curve.

1. Random Forest Classifier
A random forest model is a type of ensemble model in which several decision trees are built on bootstrapped training samples. When building these decision trees, each time a split in a tree is considered, a random sample of m predictors is chosen as split candidates from the full set of p predictors. A fresh sample of m predictors is taken at each split, and typically we choose $m \approx \sqrt{p}$. This process de-correlates the trees, thereby taking the average of the resulting trees which yields less variable and hence more reliable estimates. Using a small m in building a random forest will typically be helpful when we have many correlated predictors.
The model was tested across a number of different hyperparameters, with the optimal set found to be:

- Number of trees, $B = 150$
- Max depth of trees, $d = 5$
- Split criterion, 'entropy'

With our hyperparameters chosen we proceeded to score the model on the basis of ROC-AUC with the result being '0.7985'. Based on these parameters the model was run on the test data and the following confusion matrix was are arrived at:

		Actual	
		Subscribed	Not Subscribed
Predicted	Subscribed	0.077041	0.147366
	Not Subscribed	0.035607	0.739985

The sensitivity here is 67.96%, with a corresponding specificity of 34.3%.
The model achieved an AUC score of 0.81.

2. XGBoost

Boosting is a machine learning technique that involves combining many decision trees to improve predictions. Unlike fitting a single decision tree to the data, which can lead to overfitting, Boosting learns slowly. The idea behind boosting is to train a series of models, each of which tries to correct the errors made by the previous models. The final model is an ensemble of all the models trained during the Boosting process. Boosting is particularly useful when dealing with high-dimensional data, as it can help to reduce the risk of overfitting. It is also effective when dealing with imbalanced datasets, where one class of data is more prevalent than the other, as is the case with our dataset.
The model was tested across a number of different hyperparameters, with the optimal set found to be:

- Number of trees, $B = 50$
- Max depth of trees, $d = 5$
- Shrinkage parameter, $\lambda = 0.1$

With our hyperparameters chosen we proceeded to score the model on the basis of ROC-AUC with the result being '0.800'. This indicates high accuracy but further analysis of the predictions need to be carried out keeping in mind the business requirement. Therefore, based on these parameters the model was run on the test data and the following confusion matrix was are arrived at to gather further insights:

		Actual	
		Subscribed	Not Subscribed
Predicted	Subscribed	0.075666	0.164198
	Not Subscribed	0.036983	0.723153

The Boosting model achieved a sensitivity of 67.2% with a specificity of 31.5%. This was very similar to the performance of the Random Forest, with Boosting performing marginally worse on both counts. The AUC score achieved by the model on our test set was 0.77, again, slightly worse than the Random Forest.

3. Support Vector Machine (SVM)

SVM is a popular classification approach, being considered one of the best out-of-the-box classifiers and will be used in the binary classification setting. The 'maximal margin classifier' (MMC) which seeks to perfectly separate observations using a separating hyperplane by maximising the minimal distance from the observations to the hyperplane (i.e., maximise the 'margin'). However, with complex real data it is highly unlikely that there will exist a perfect linear boundary with no violations of the hyperplane – thus it is necessary to introduce SVM which is a generalisation of the MMC, allowing high-dimensional transformations of the predictor variables added to the feature space in a computationally efficient manner.
Due to the class imbalanced observed in Q1. it was necessary to balance the classes sampled by the SVM model.

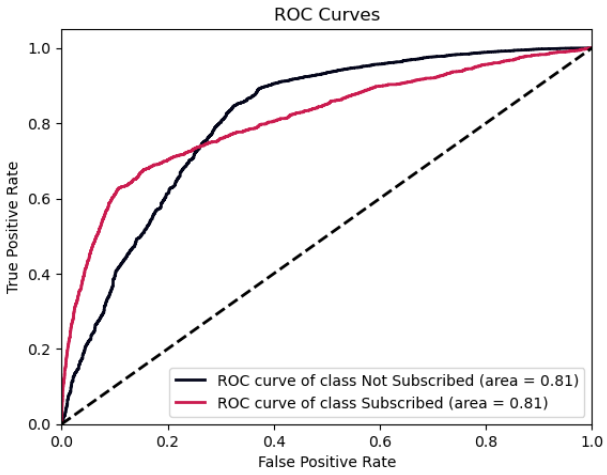
		Actual	
		Subscribed	Not Subscribed
Predicted	Subscribed	0.082221	0.252488
	Not Subscribed	0.030428	0.634863

The performance of SVM is excellent in terms of specificity but this comes at the cost of an increase in false positives of 11% over Random Forest and 9% over Boosting. While the model undoubtedly performs well, the implementation would prove to be costly due to the increased wages paid and

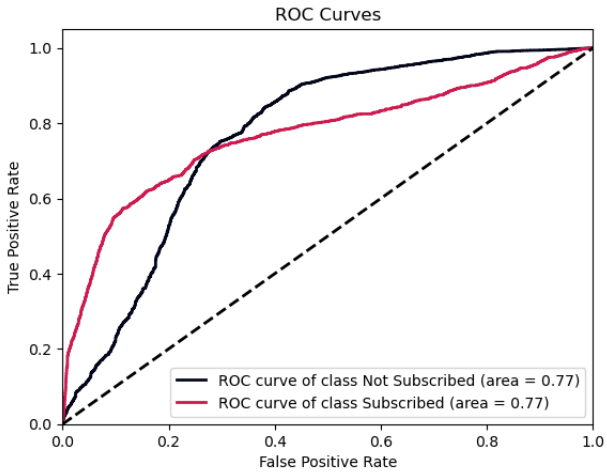
man-hours necessary as a result of the increased amount of total predicted positives. Despite the high true positive rate, we will still favour Random Forest as it will prove more cost-effective in the business setting.

As mentioned previously, the performance of the Boosting & Random Forest procedures were very similar, with Random Forests the preferred model. The SVM model had a good advantage in terms of sensitivity, but this was at a cost of a large drop in the specificity of the model.
Hence, based on our analysis the Random Forest model would be the recommended model for this business case.

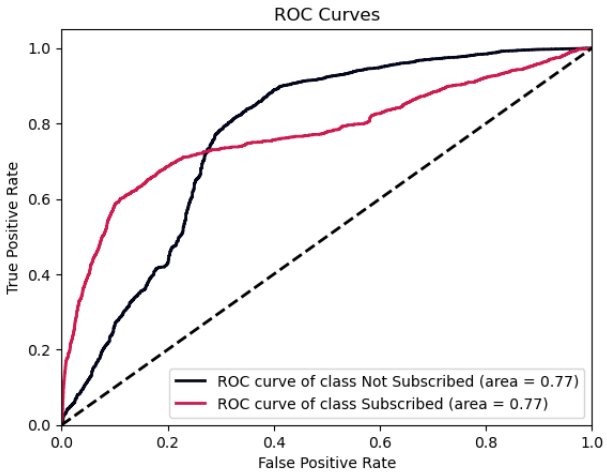
b) How do these models perform compared to the fitted logistic regression model in question 2?



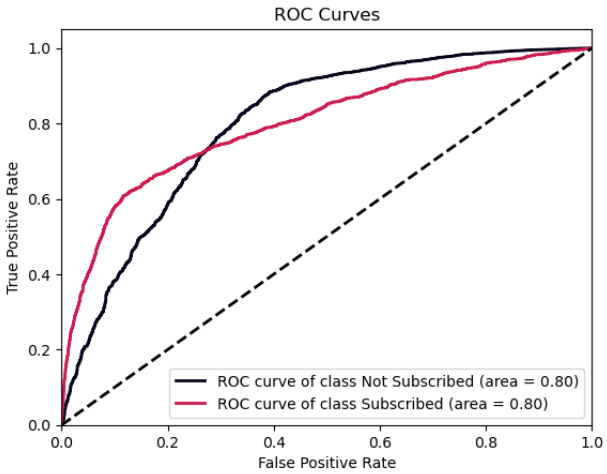
(a) ROC Curve for Random Forest model



(b) ROC Curve for Boosting model



(c) ROC Curve for SVM model



(d) ROC Curve for Logistic Regression model

The performance of each of the 3 models was quite similar to that of the logisitic regression (at a probabillity threshold of 10%) in terms of sensitivity, with SVM & the Random Forest performing slightly better, with the Boosting procedure marginally worse. The real difference between the models comes in the specificity score, with Random Forest achieving a much better score. The relatively low specificity of the SVM model is again noted here, with its score roughly 7% higher than that of the logistic regression model.
Overall the performance of the logistic regression model was slightly worse than the Random Forest model, but about in line with that of SVM & Boosting.

Q4:

Evaluate the potential, from theoretical and empirical perspectives, of the following unsupervised learning techniques to improve (or not) the predictive modeling work you have reported.

1. Principal Components Analysis
2. K-Means Clustering
3. Hierarchical Clustering

1. Principal Components Analysis (PCA)

PCA is a dimensional reduction technique which uses the singular value decomposition (SVD) of a covariance matrix, in order to in construct an orthogonal set of basis vectors based on the directions upon which the variability of the dataset are greatest. If there is a high degree of correlation between our variables, it is possible that the a significant portion of the variance can be explained using just the first few principal components. This has the potential to remove some of the statistical noise and redundant features, thereby improving the performance of our models. As has already been seen in the correlation heatmap from Q1., the correlation between our predictors in this case have tended to be very low, potentially reducing the effectiveness of PCA in this case.
Once the principal components were calculated, the proportion of variance explained by each component can then be calculated from the corresponding eigenvalue. $\frac{\lambda_i}{p}$, where λ_i represents the i^{th} eigenvalue, and p represents the dimension of the dataset.

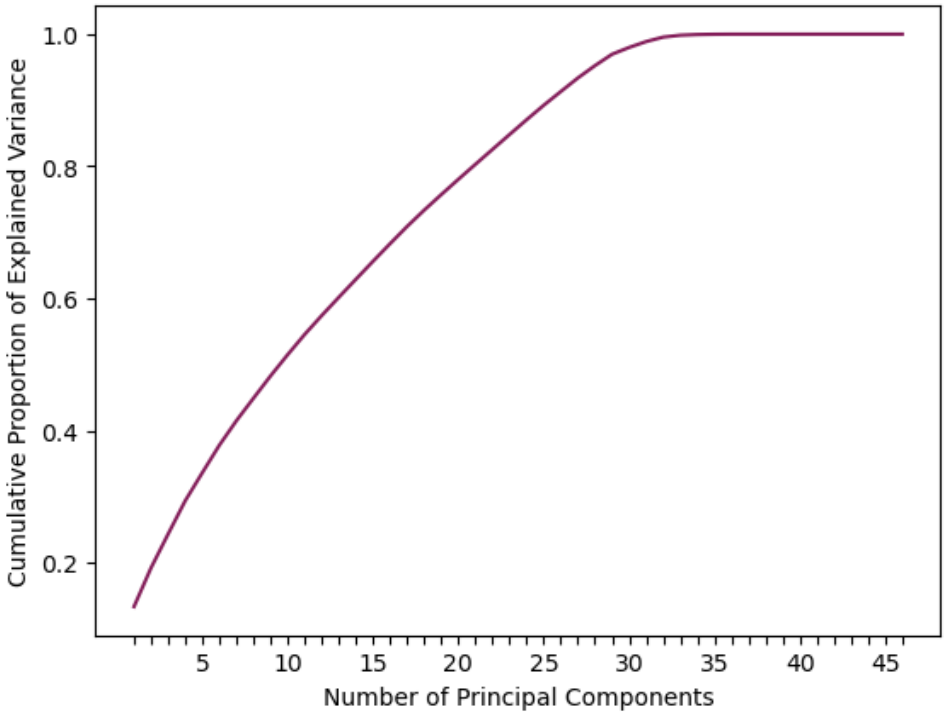


Figure 5 Cumulative proportion of explained variance for each component.

It can be inferred from this that the components do a relatively poor job at explaining the variance in the data. Due to this poor performance, there were no advantages to implementing PCA for our classification model.

2. K-Means Clustering

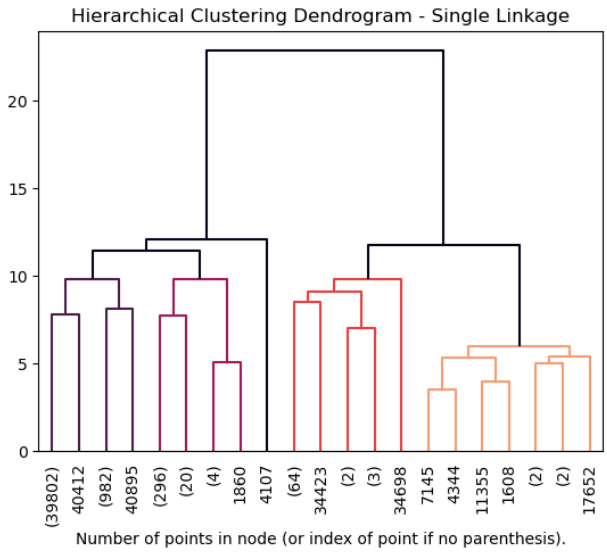
K-means clustering is a clustering algorithm which partitions a dataset into K separate clusters based on their similarity. The algorithm starts by randomly assigning K points as the initial centroids of the clusters. Then, each data point is assigned to the nearest centroid, and the centroids are recalculated based on the mean of the points assigned to them. This process continues until the centroids stay constant ie. the algorithm reaches convergence. K-means clustering is an efficient and widely used clustering algorithm for datasets with many dimensions. In this report, clustering solution from K = 2 to K = 7 was considered, creating the corresponding silhouette plots for each solution.
Looking at the silhouette plots holistically, none of the clustering solutions seem to be particularly good at partitioning the data in similar clusters, based on the silhouette indices. The model with the best average silhouette index achieved a score of 0.127, which indicates marginally similarly distributed clusters.
However, examining the pairs plot below, the clusters do seem to be better separated, indicating that perhaps some form of clustering structure has been obtained.
Looking at the relationship between membership of each of the clusters and the subscriptions to term deposits, there is evidence that our clustering solution has uncovered groups which demonstrate different subscription behaviours, with observations within cluster 1 much more likely to subscribe to an account.

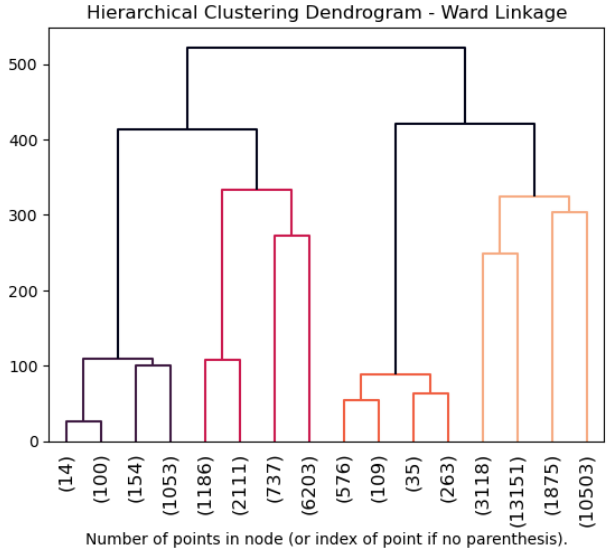
		Term Deposit	
		Subscribed	Not Subscribed
Cluster	0	1172	23130
	1	3468	13418

3. Hierarchical Clustering

Two forms of hierarchical clustering exist, describing different methods of building up our clusters, divisive & agglomerative, with divisive clustering starting from the one single cluster until all observations are contained in their own individual cluster and agglomerative starting where each data point starts in its own cluster and the algorithm merges clusters until all data points are in the same cluster. For the purpose of the analysis, an agglomerative approach was taken. The result of hierarchical clustering solution can be visualized in the form of a dendrogram, which is a tree-like diagram that displays the hierarchical relationships between the clusters. The tree can then be 'cut' off at a certain level of the similarity/difference metric in order to arrive at a K-cluster solution, for any given value of K.
Many different ways of determining which clusters to combine, linkages, exist, with each yielding a different clustering solution. Using a single linkage solution, which combines clusters based on the minimum difference between any pairs of points in each cluster, gave a solution with one extremely large cluster (39,800 observations), with a few very small clusters. This is known as the chaining effect, and is a common problem when using a single linkage approach. Therefore, alternative linkage approaches had to be explored.
Ward linkage yielded more evenly distributed clusters, with two large clusters (each with more than 10,000 observations) and two smaller clusters (each smaller than 2,000 observations), which was more suitable for the dataset.
The corresponding dendrograms for each solution can be found here.
The separation of these clusters was further examined using pairs plots of all the numerical variables, as done before for the K-means solution.
Once again examining the relationship between the variable of interest and the clustering solution, we see that each cluster does seem to have a different frequency of subscriptions. Observations in cluster 0 had about a 65% chnace of subscribing vs. the general population figure of 11.2%.

		Term Deposit	
		Subscribed	Not Subscribed
Cluster	0	864	457
	1	107	876
	2	2037	26610
	3	1632	8605



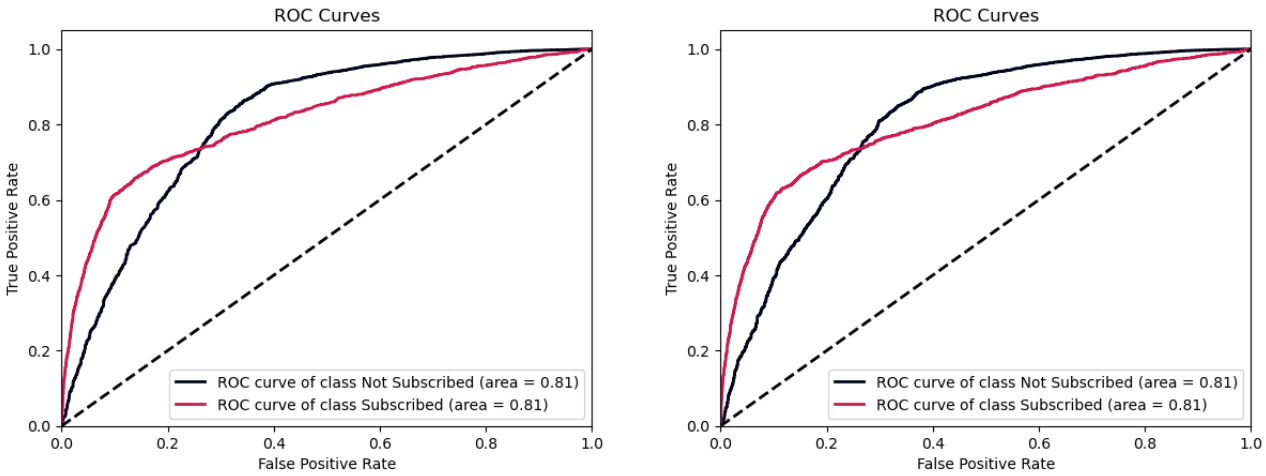


Since the performance of the Random Forest & Boosting models were so similar, it was decided that both models should be refit with the new cluster labels. Using the same process of hyperparameter tuning, each model was fit, with the resulting confusion matrices shown below.

Random Forest		Actual	
		Subscribed	Not Subscribed
Predicted	Subscribed	0.077041	0.147366
	Not Subscribed	0.035607	0.739985

XGBoost		Actual	
		Subscribed	Not Subscribed
Predicted	Subscribed	0.078093	0.159828
	Not Subscribed	0.034555	0.727523

With the new cluster labels as features, we notice that the XGBoost is now outperforming the Random Forest model in terms of both sensitivity and specificity.



The ROC curves show near identical performance for each model, with and AUC of 0.81 for both. To get an idea as to the relative importance of each of the features in the model, the following importance plot was created:

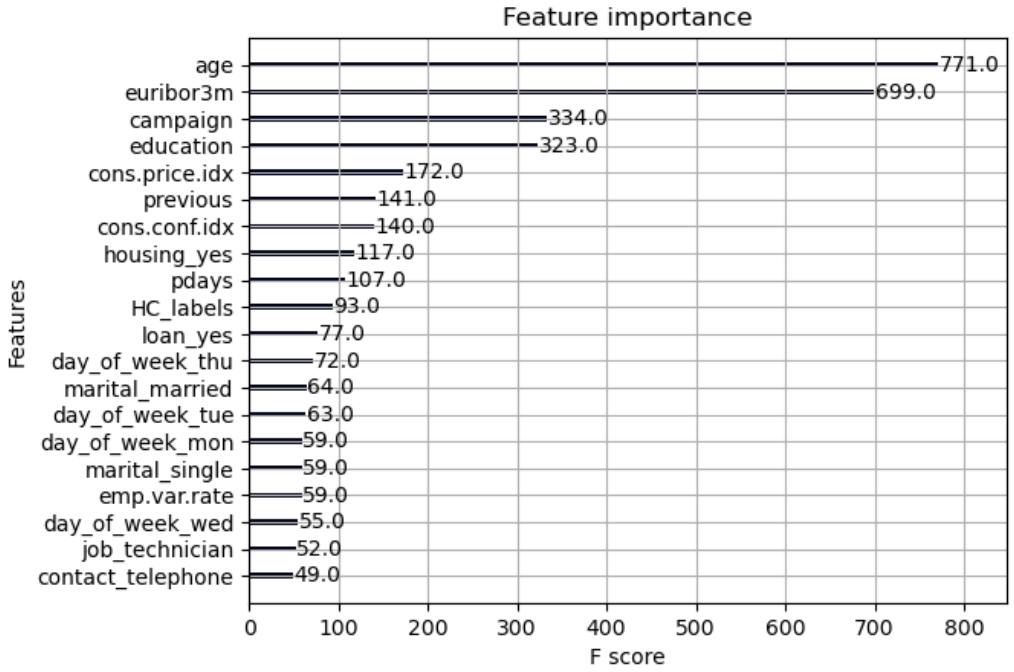


Figure 8 Plot of the feature importance score of the top 20 most important variables.

Unsurprisingly, two general features dominate, these being the age and Euribor 3-month rate. Noteworthy is the importance of hierarchical clustering labels which is in line with the enhanced performance of Boosting relative to the other models after clustering.

Appendix:

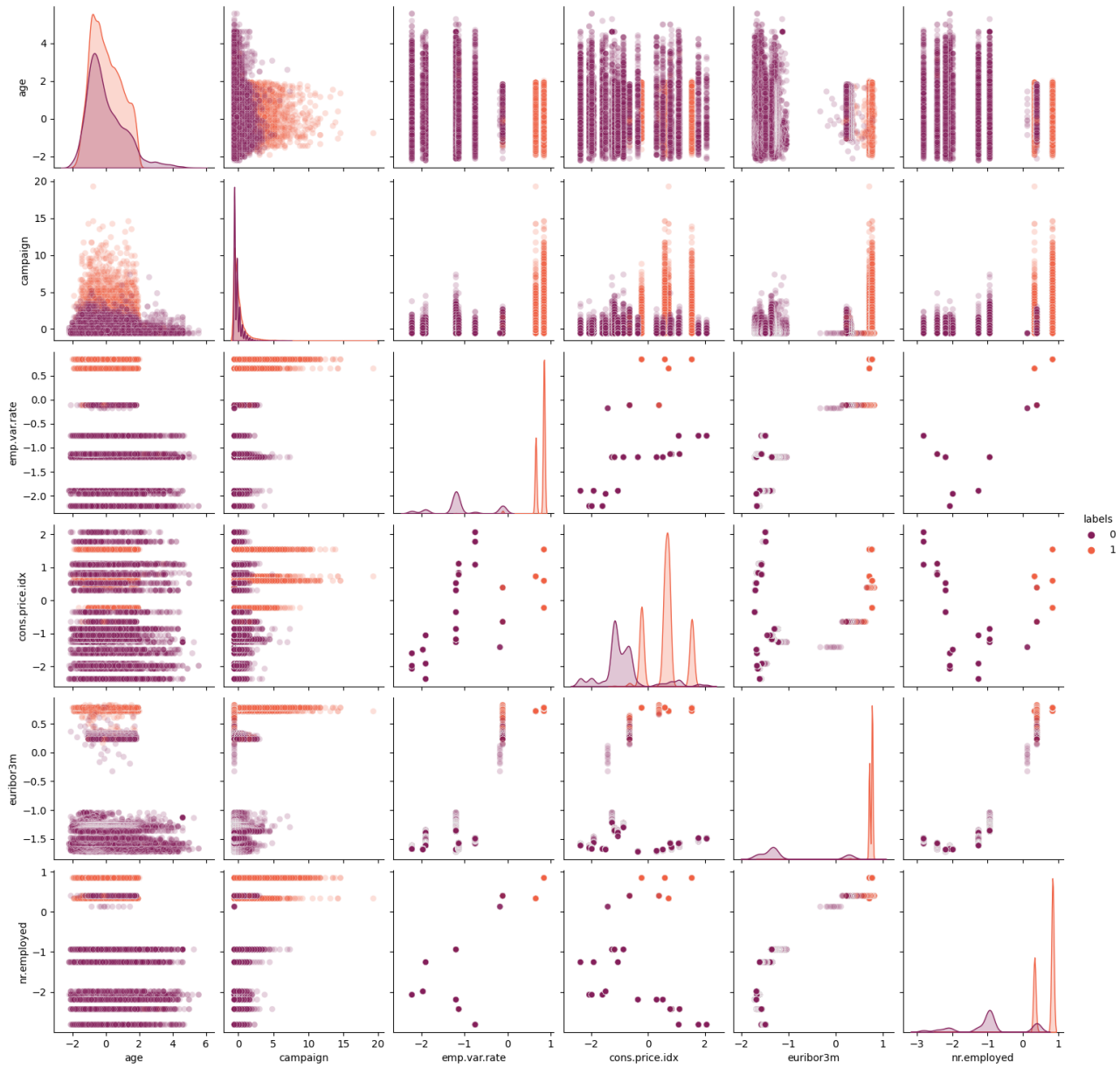


Figure 9 *Pairs plot of the variables, coloured based on cluster membership.*
**Some of the predictor variables are not plotted for readability purposes.*

Return to Report

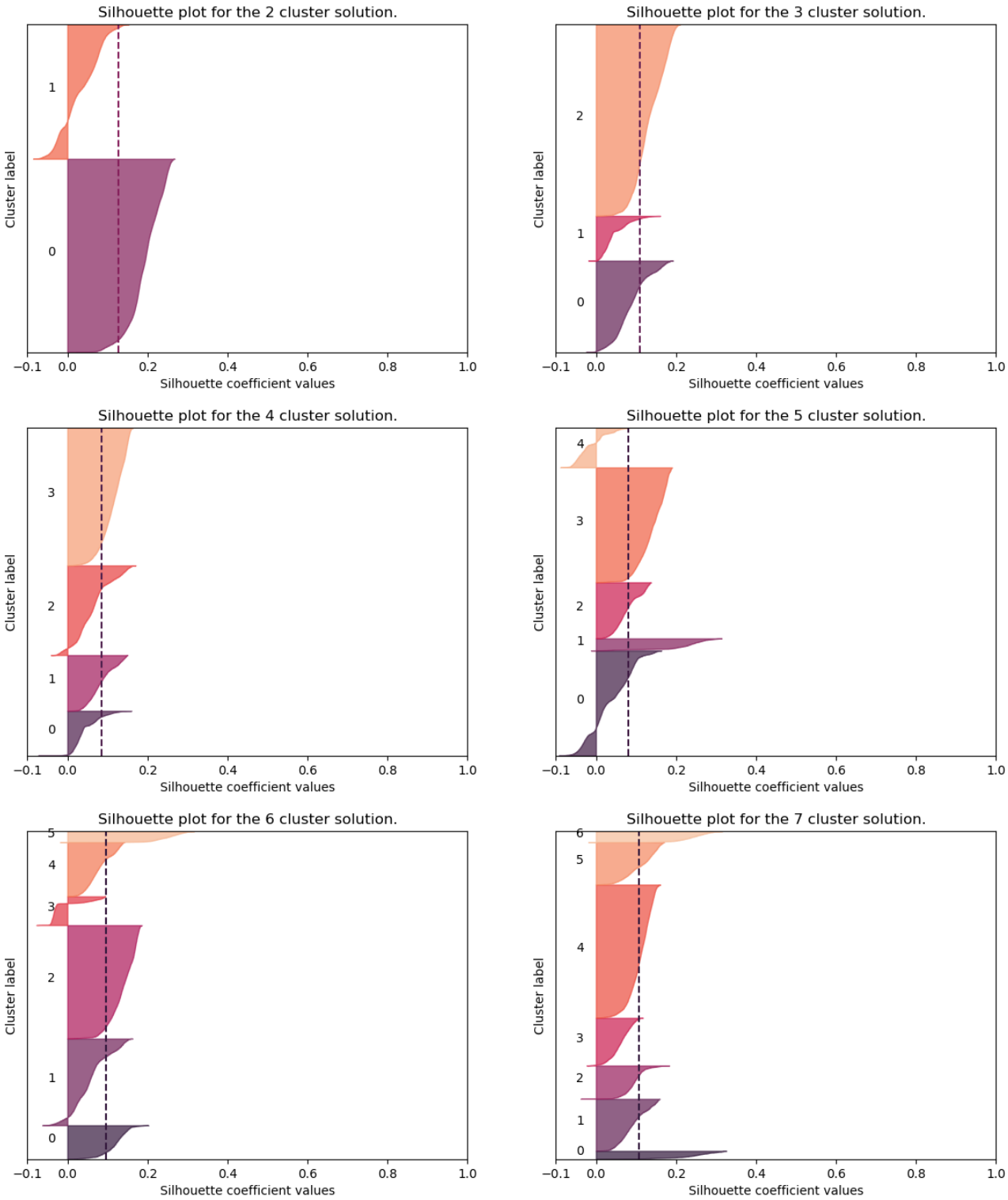


Figure 10 Silhouette plots for each clutering solution.

Return to Report

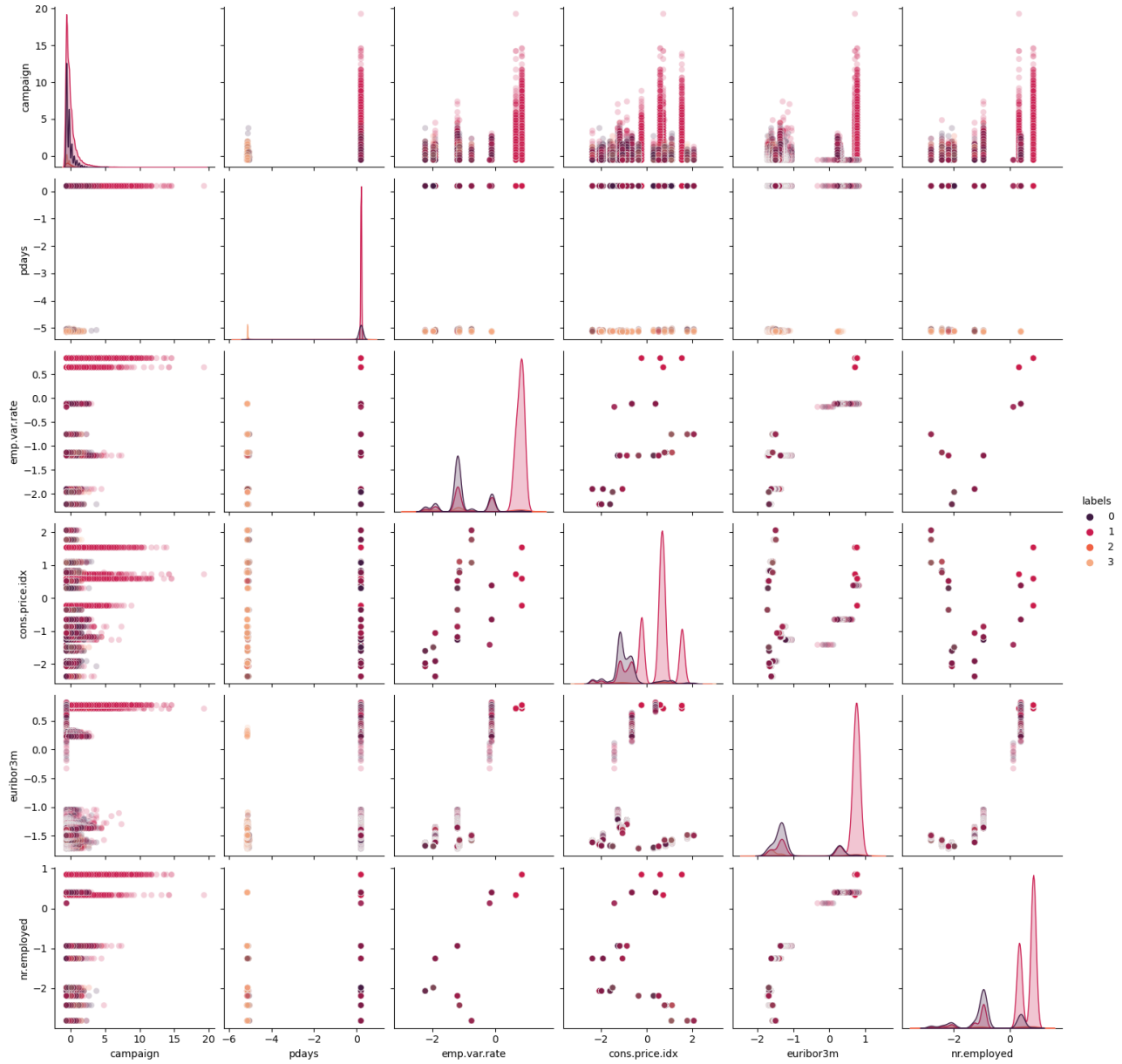


Figure 11 *Pairs plot of the variables, coloured based on cluster membership.*
**Some of the predictor variables are not plotted for readability purposes.*

Return to Report

References:

Hyperparameter Tuning
Random Forest
SVM
Silhouette Index
Dendrogram
RF Feature Importance Plot