Machine Learning Final Project Report:

Machine Learning Analysis of Bank Marketing Campaigns' Success

Hao Wu, Qin Yao, Wilson Wu

DePaul University

DSC 478
Professor Bamshad Mobasher

March 16, 2020

Table of Contents

1. Introduction

Marketing is an unique and essential part to the commercial banking industry. For most banks, optimizing marketing strategies could very likely result in increased performance. An accurate model to predict customer decisions may lead to higher customer satisfaction as well as improved efficiency in its routine businesses.

The marketing campaigns' data from a Portuguese banking institution combined with social and economic indicators will be used in this project. Our goals are: use basic statistical analysis, data exploration, and data visualization techniques to provide a deeper understanding how features including clients' profile, marketing strategies, social and economic indexes affect the campaigns' success; apply various machine learning techniques to generate different classifiers and find the best optimized model to predict the success of telemarketing calls for selling bank long-term deposits; use unsupervised techniques to discover the interesting pattern hidden within the dataset.

This report was divided into ten sections: graphical analysis and statistical analysis were explored in section 5; different predictive models with comparison of performance were given in section 6; data clusters with evaluation were shown in section 7; conclusions and future works were drawn in section 8.

42. Dataset

The data was related with direct marketing campaigns of a Portuguese banking institution. The marketing campaigns were based on phone calls. Often, more than one contact to the same client was required, in order to assess if the product (bank term deposit) would be ('yes') or not be ('no') subscribed[1]. The data was enriched by the addition of five new social and economic features/attributes.

Independent Variables:
bank client data:
1 - age (numeric)
2 - job : type of job (categorical:
'admin.','blue-collar','entrepreneur','housemaid','management','retired','self-employed','services','student','technician','unemployed','unknown')
3 - marital : marital status (categorical: 'divorced','married','single','unknown'; note: 'divorced' means divorced or widowed)

4 - education (categorical:
'basic.4y','basic.6y','basic.9y','high.school','illiterate','professional.course','university.degree','unknown')
5 - default: has credit in default? (categorical: 'no','yes','unknown')
6 - housing: has housing loan? (categorical: 'no','yes','unknown')
7 - loan: has personal loan? (categorical: 'no','yes','unknown')
related with the last contact of the current campaign:
8 - contact: contact communication type (categorical: 'cellular','telephone')
9 - month: last contact month of year (categorical: 'jan', 'feb', 'mar', ..., 'nov', 'dec')
10 - day_of_week: last contact day of the week (categorical: 'mon','tue','wed','thu','fri')
11 - duration: last contact duration, in seconds (numeric). Important note: this attribute highly affects the output target (e.g., if duration=0 then y='no'). Yet, the duration is not known before a call is performed. Also, after the end of the call y is obviously known. Thus, this input should only be included for benchmark purposes and should be discarded if the intention is to have a realistic predictive model.
other attributes:
12 - campaign: number of contacts performed during this campaign and for this client (numeric, includes last contact)
13 - pdays: number of days that passed by after the client was last contacted from a previous campaign (numeric; 999 means client was not previously contacted)
14 - previous: number of contacts performed before this campaign and for this client (numeric)
15 - poutcome: outcome of the previous marketing campaign (categorical: 'failure','nonexistent','success')
social and economic context attributes
16 - emp.var.rate: employment variation rate - quarterly indicator (numeric)
17 - cons.price.idx: consumer price index - monthly indicator (numeric)
18 - cons.conf.idx: consumer confidence index - monthly indicator (numeric)
19 - euribor3m: euribor 3 month rate - daily indicator (numeric)
20 - nr.employed: number of employees - quarterly indicator (numeric)
Dependent variable (desired target):
21 -y - has the client subscribed a term deposit? (binary: 'yes','no')

3. Methodology

Data Exploration: examine the means, standard deviations and other statistics associated with the numerical attributes, and visualize all interesting features related to our target data.

Predictive Modeling (supervised): use supervised machine learning techniques to generate predictive models, which includes Logistic Regressions, Ridge Classification, Naive Bayes, Decision Tree, LDA, k-NN, SVM, Random Forest. Perform feature selection, model selection and other methods to optimize the models. Compare the performance of all models.

Exploratory Data Analysis (unsupervised): apply K-mean clustering to discover interesting patterns.

4. Data Preprocessing

The missing values in this dataset were all in categorical attributes and were coded with the “unknown” label. So we decided to treat the missing values as a possible class label. Dummy variables were created for all categorical attributes. The dataset was splitted into 80% training set and 20% testing set for training and testing the upcoming prediction models. Because the features are very different in scale and would contribute unequally to the analysis, we performed min-max scaler for the dataset.

5. Data Exploration - Qin

In this dataset, there were 41,188 clients across 20 different features, both categorical and numeric[1]. The first step was to load the dataset into a data frame for easy manipulation and exploration using the pandas package, then getting some statistical information of the data frame.

The next step was using the seaborn package to explore and clean the numeric variables such as "age," "previous," "duration." Making box plots for each one that finds out whether variables had outliers or not. Next, we focus on categorical variables such as 'job type,' 'marital status,' 'education.' Plots for each were produced that looked at their relative frequency as well as the normalized relative frequency

In appendix A-1, we could see that the frequencies for each level of the categorical features in the dataset. First, we saw that the desired target was unbalanced, only around 10% of the observations corresponding to clients that subscribed to a term deposit. Most clients did not subscribe to a term deposit. For the day that the client contacted, we could see there was an equilibrium between levels presented in the day.of.week. Also, in the month graph, we got that the last contact of most of the clients was in May. In the poutcome graph, most of the clients have a nonexistent previous marketing campaign, and many people who were married had jobs in the administrative sector with university degrees.

In appendix A-2, we could find out the relationship between each categorical variable and term deposit subscription.We saw that retired people and people who did not have a loan and marry subscribed a term deposit. Besides, their poutcome was a success, and clients who had university degrees had a higher subscription.

In the following step, we would be analyzed in a few variables by statistical description. We knew that age was less than 30 or greater than 60 had a higher success rate. However, only 3% of clients had the age of 60 and higher. Most of the people, who were contacted, had tertiary or secondary education. Top contacted clients whose job types were 'blue-collar,' 'management,' and 'technician.' The success rate was highest for the student. Very few clients were contacted, and many of their jobs were defaulter. As seen for the default variable, fewer clients who had a loan were contacted. Most of the people were contacted through cellular. The attribute pdays was a clear distinction in quartile ranges of pdays for target variable yes and no. 75% of clients contacted through the campaign were not previously contacted.

The attribute duration seemed to be an essential feature as there was a clear distinction in quartile ranges of duration for target variable yes and no. 75% call duration was less than or equal to 319. The duration had a mean of 258.28 and a standard deviation of 259.28.

75% of previous values equal to 0, and 99% values were less and equal to 2.0 and previously had a mean of 0.17 and a standard deviation of 0.49. Most of the clients contacted have the previous outcome as 'unknown.'

We saw the correlations (Figure 1), two-by-two, for all the numerical features in the dataset. We saw a random behavior in most variables. This random behavior was described by a correlation close to zero or between the interval -0.6 and 0.9. There were three strong positive correlations: emp.var.rate and euribor3m, euribor3m and nr.employed, and emp.var.rate and nr.employed.

Figure 1: Correlation Matrix

6. Predictive Modeling

6.1. Random Forest Classifier – Qin

The reason we did a random forest is the low correlation. We used the package Sklearn.ensemble.RandomForestClassifier for the Random Forest classifier. The random forest combines hundreds or thousands of decision trees, trains each one on a slightly different set of the observations, splitting nodes in each tree considering a limited number of the features. The final predictions of the random forest are made by averaging the predictions of each tree [2]. In this method, the individual trees were purposely overfitted, and the validation set was used to optimize tree-level parameters. We made class predictions as well as predicted scores to calculate the ROC AUC. Once we had the testing predictions and training prediction, we calculated the ROC AUC.

The final test ROC AUC for the random forest was 0.754 (Figure 2). The training scores in both models were 1.0 ROC AUC with the max depth was 25. We plot the ROC curve for the single decision tree and the random forest. A curve to the top and left was a better model.

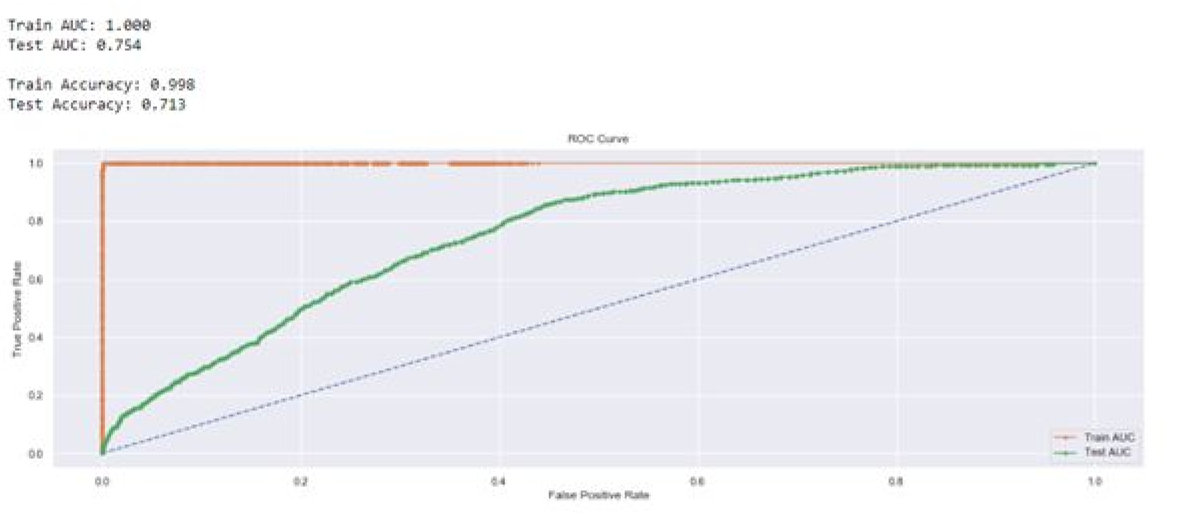


Figure 2: Random Forest

And then, we took the diagnostic measure of the model. Here was the confusion matrix for the training predictions (figure 3). The graph below showed that predictions in the top left and bottom right corners were correct in the model, and the predictions in the lower left and upper right of the model were missed.

Figure 3: Random training set

Finally, we took the diagnostic measure of the model. Here was the confusion matrix for the testing predictions (figure 4). The figure below showed predictions in the top left and bottom right corners were correct in the model, and the predictions in the lower left and upper right of the model were missed.

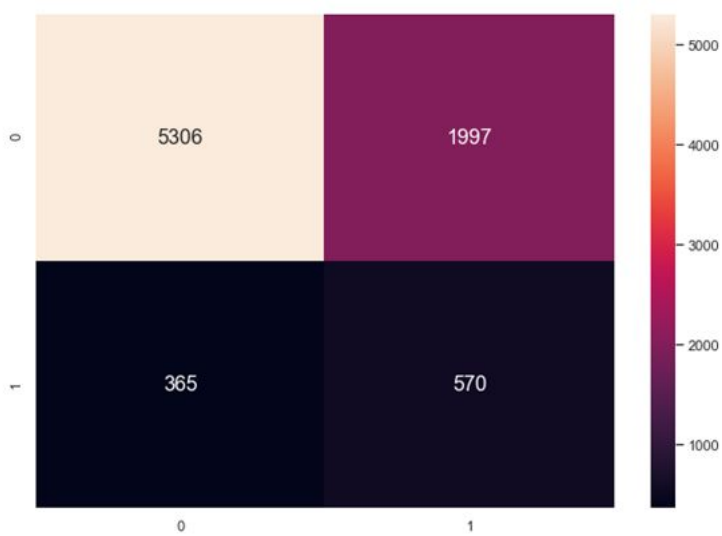


Figure 4: Random testing set

6.2. Ridge, k-Nearest Neighbors, Decision Tree Classifiers – Hao

6.2.1 Background

Ridge Classifier – This classifier first converts binary targets to {-1, 1} and then treats the problem as a regression task, optimizing the same objective as above. The predicted class corresponds to the sign of the regressor's prediction.

K-Nearest Neighbors (k-NN) Classifier – It is a non-parametric method. The output is a class membership. An object is classified by a plurality vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors.

Decision Tree Classifier – It is a non-parametric supervised learning method. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features.

We implemented above three techniques to build classifiers to predict if the client has subscribed a term deposit. After data cleaning, we performed min-max scaler as the preprocessing step because the features are very different in scale and would contribute unequally to the analysis. We optimized the classifiers by model selection and feature selection to see how much improvement compared with the default model from scikit-learn. To avoid overfit and get accurate performance of models, we also used cross-validation. Finally, we compared the performance of each predictors in the testing set (20% of full data set) by accuracy to achieve our goal, which is to find a best classifier for the Bank Marketing Data Set to predict the success of bank marketing campaigns.

6.2.2 Results

K-Nearest Neighbors (k-NN) Classifier
The default setting of k-NN classifier and its performance on the testing set:

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
           metric_params=None, n_jobs=None, n_neighbors=5, p=2,
           weights='distance')

Accuracy:0.879

Classification report
              precision    recall  f1-score   support

           0       0.91      0.96      0.93      7288
           1       0.46      0.27      0.34       950

   micro avg       0.88      0.88      0.88      8238
   macro avg       0.68      0.61      0.64      8238
weighted avg       0.86      0.88      0.87      8238
```

In order to optimize our model, we performed model selection and feature selection. According to the results in Appendix B-1, we took k as 17 and weight as 'uniform'. Total 17 features were kept as a subset and used for our best model.

The optimized setting of k-NN classifier and its performance on the testing set:

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
           metric_params=None, n_jobs=None, n_neighbors=17, p=2,
           weights='uniform')

Accuracy:0.906

Classification report
              precision    recall  f1-score   support

           0       0.93      0.96      0.95      7288
           1       0.62      0.48      0.54       950

   micro avg       0.91      0.91      0.91      8238
   macro avg       0.78      0.72      0.75      8238
weighted avg       0.90      0.91      0.90      8238
```

The prediction accuracy was increased from 0.879 to 0.906 which is a 3.1% increase. The confusion matrix and ROC curve in Figure 5, 6 also showed a good performance of the k-NN predictor.

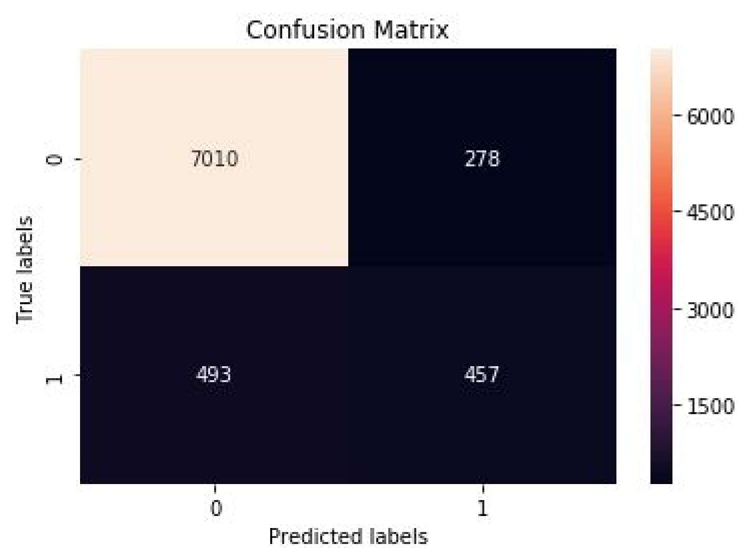


Figure 5: Confusion matrix of optimized k-NN classifier

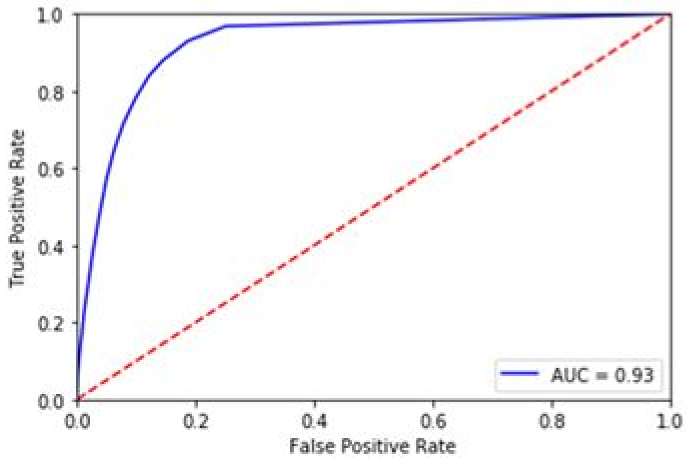


Figure 6: ROC curve of k-NN classifier

Decision Tree Classifier

The default setting of decision tree classifier and its performance on the testing set:

```
DecisionTreeClassifier(class_weight=None, criterion='entropy', max_depth=None,
            max_features=None, max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, presort=False, random_state=None,
            splitter='best')

Accuracy:0.882

Classification report
              precision    recall  f1-score   support

           0       0.94      0.93      0.93      7288
           1       0.49      0.50      0.50       950

   micro avg       0.88      0.88      0.88      8238
   macro avg       0.71      0.72      0.72      8238
weighted avg       0.88      0.88      0.88      8238
```

In order to optimize our model, we performed model selection and feature selection. According to the results in Appendix B-2, we setted criterion as ‘gini’, max_depth as 7, min_samples_leaf as 23, and min_samples_split as 2. Total 17 features were kept as a subset and used for our best model.

The optimized setting of decision tree classifier and the performance on the testing set:

```
DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=7,
            max_features=None, max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=23, min_samples_split=2,
            min_weight_fraction_leaf=0.0, presort=False, random_state=None,
            splitter='best')

Accuracy:0.909

Classification report
              precision    recall  f1-score   support

           0       0.94      0.96      0.95      7288
           1       0.62      0.53      0.57       950

   micro avg       0.91      0.91      0.91      8238
   macro avg       0.78      0.74      0.76      8238
weighted avg       0.90      0.91      0.91      8238
```

The prediction accuracy was increased from 0.882 to 0.909 which is a 3.1% increase. The confusion matrix and ROC curve in Figure 7, 8 also showed a good performance of the decision tree predictor.

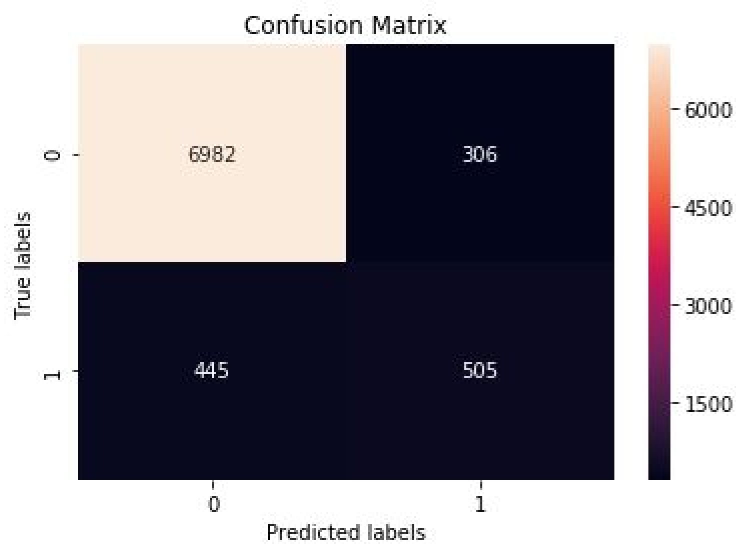


Figure 7: Confusion matrix of optimized decision tree classifier

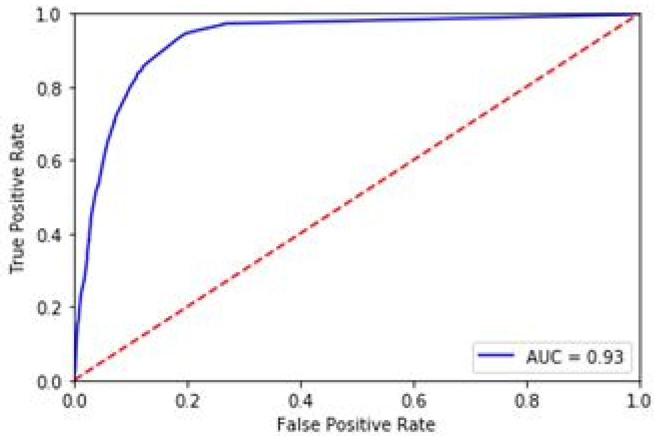


Figure 8: ROC curve of decision tree classifier

Ridge Classifier

We performed a similar method to get an optimized ridge classifier which took 1.06 as the best alpha. Also, 40 features were selected for the final predictor. However, the increase compared with the default model was very small. The results can be referred to Appendix B-3. We finally got 90.2% accuracy when we used the optimized ridge classifier to predict if the clients in the testing set have subscribed to a term deposit. The confusion matrix in Figure 9 indicated the ridge classifier was also a good choice for our prediction task.

The optimized setting of ridge classifier and the performance on the testing set:

```
RidgeClassifier(alpha=1.0621052631578947, class_weight=None, copy_X=True,
        fit_intercept=True, max_iter=None, normalize=False,
        random_state=None, solver='auto', tol=0.001)


Accuracy:0.902

Classification report
              precision    recall  f1-score   support

           0       0.92      0.98      0.95      7288
           1       0.65      0.32      0.43       950

   micro avg       0.90      0.90      0.90      8238
   macro avg       0.79      0.65      0.69      8238
weighted avg       0.89      0.90      0.89      8238
```

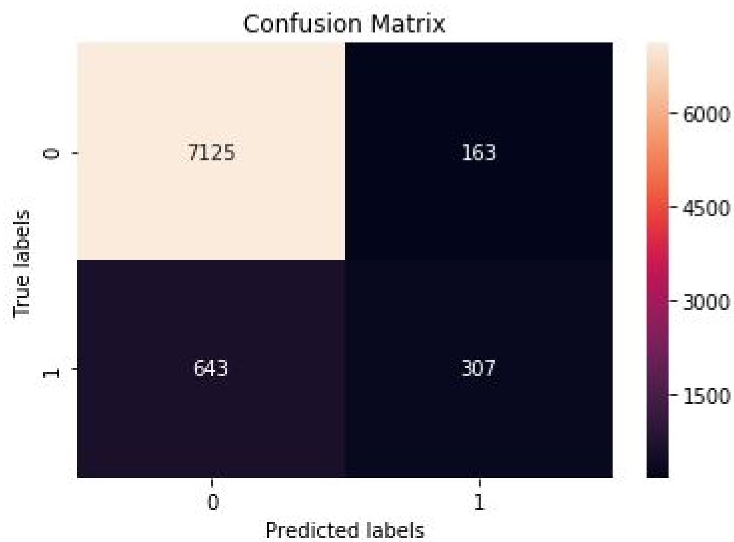


Figure 9: Confusion matrix of optimized ridge classifier

Comparison of Three Classifiers

Overall, all three classifiers did good jobs for our prediction task. The decision tree model with 90.9% prediction accuracy in the testing set was slightly better than the others. According to the confusion matrices, the decision tree model had best false-negative rate and worst false-positive rate among the three models. Considering this, we can pick different models for different aims.

6.3. LogisticRegression, Naive Bayes, LDA, SVM Classifiers – Wilson

Logistic Regression

For the Logistic regression result. We compared the training accuracy and the testing accuracy for the normal without any regularization, L1 regularization, L2 regularization and Elastic-Net. Moreover, for our regularization, we also tested four different inverse of regularization strength numbers from C = [10, 1, 0.1, 0.001]. First, Our normal Logistic regression, we got training accuracy in 91.04 % and test accuracy in 91.05 %. Second, in L1 regularization, we picked C=10 to achieve best accuracy which is 91.14% in training accuracy and 91.11% in testing accuracy. Third, in L2 regularization, we picked C=0.001 to achieve best accuracy, and we got training accuracy in 90.87% and test accuracy in 91.06%. Last, in Elastic-Net, we also added one more penalty, l1_ratio, to find the sweet spot for L1 and L2. There were five different values for l1_ratio [1,.5,.1,.01,.001], because l1_ratio should be between 1 to 0. To get our best accuracy, we picked C= 0.001 and l1_ratio = 0.5, so we got training accuracy in 90.58% and test accuracy in 90.7%. Based on all our four models, although there was not a huge difference between all of them, because they all surrounded at 90% to 91%, we still finally chose L1 regularization to be our better performance model which resulted in 91.11% testing accuracy. Furthermore, the Precision and recall both had high percent performance (Figure 10) and on the confusion matrix, general performed well, but for false positive, which was mean the client didn't subscribed a term deposit, but classify as subscribed, seems still had amount of number to be solved in future (Figure 11). Finally, in our L1 regularization ROC curve, it also showed that even if we raised up the thresholds, but we still could not fix the False Positive, it seemed to be flatten after the true positive rate over 0.9 (Figure 12).

```
-------------------------------------------------------------------------

Training accuracy: 0.9114415781487102
              precision    recall  f1-score   support

           0       0.97      0.93      0.95     30683
           1       0.41      0.67      0.51      2267

    accuracy                           0.91     32950
   macro avg       0.69      0.80      0.73     32950
weighted avg       0.94      0.91      0.92     32950

Predict Training Data with Actual Training Data:
 [[28505  2178]
 [  740  1527]]

-------------------------------------------------------------------------

Test accuracy: 0.911143481427531
              precision    recall  f1-score   support

           0       0.97      0.93      0.95      7645
           1       0.43      0.67      0.52       593

    accuracy                           0.91      8238
   macro avg       0.70      0.80      0.74      8238
weighted avg       0.93      0.91      0.92      8238

Predict Testing Data with Actual Testing Data:
 [[7108  537]
 [ 195  398]]
```

Figure 10: L1 regularization classification report

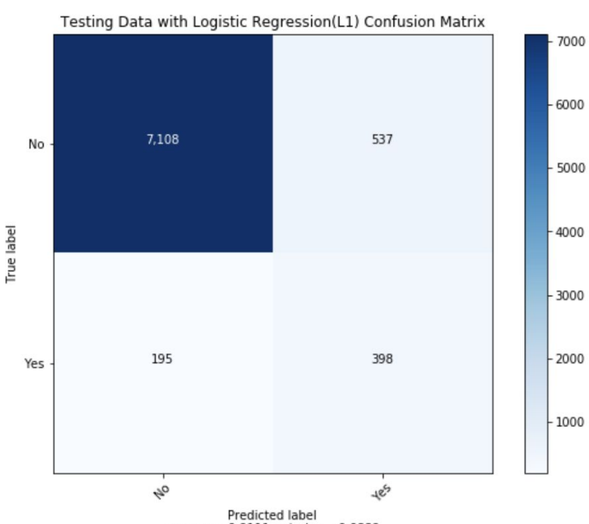


Figure 11: L1 regularization confusion matrix

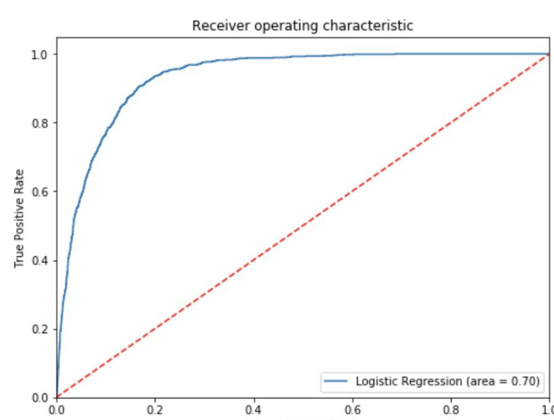


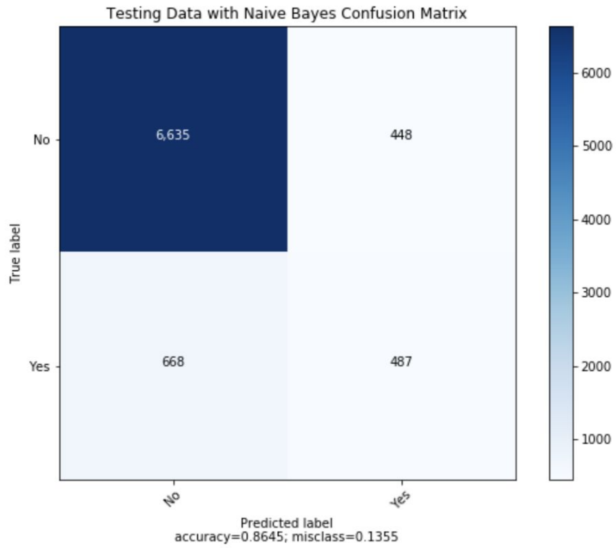
Figure 12: L1 regularization ROC curve

Naive Bayes

For the Naive Bayes result. There were not many penalties in the scikit-learn package so we just used the default setting and calculated the probability to classify the data. Our model performed 86% accuracy in testing data and training dataset. It was not really well performance which compared to Logistic and LAD, but we will see after using dimensionality reduction techniques, PCA, Naive Bayes model has some improvement. In the Precision and recall both still had high percent performance (Figure 13) and on the confusion matrix, general performed well, but for false positive, which meant the client didn't subscribed a term deposit, but classify as subscribed, seems still have amount of number to be solved in future and also similarly to the false negative. (Figure 14) Finally, in our Naive Bayes ROC curve, it seemed to be flatten after the true positive rate over 0.8 (Figure 15).

```
Naive Bayes

-------------------------------------------------------------------------
Training accuracy: 0.8640060698027314
              precision    recall  f1-score   support

           0       0.91      0.94      0.92     28318
           1       0.52      0.42      0.46      4632

    accuracy                           0.86     32950
   macro avg       0.71      0.68      0.69     32950
weighted avg       0.85      0.86      0.86     32950

Predict Training Data with Actual Training Data:
 [[26541  1777]
 [ 2704  1928]]

-------------------------------------------------------------------------
Test accuracy: 0.864530225782957

              precision    recall  f1-score   support

           0       0.91      0.94      0.92      7083
           1       0.52      0.42      0.47      1155

    accuracy                           0.86      8238
   macro avg       0.71      0.68      0.69      8238
weighted avg       0.85      0.86      0.86      8238

Predict Test Data with Actual Test Data:
 [[6635  448]
 [ 668  487]]
```

Figure 13: Naive Bayes classification report



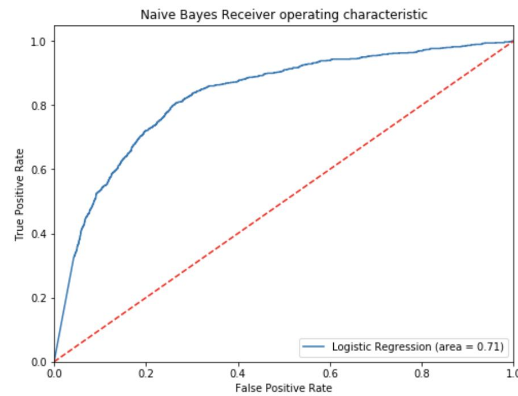


Figure 14: Naive Bayes confusion matrix **Figure 15:** Naive Bayes ROC curve

Linear Discriminant Analysis

For the Linear Discriminant Analysis result. Linear Discriminant Analysis focused on separate differences in the classes of data by creating new axes. Our model performed 90.9 % accuracy in training data and 90.8 % accuracy in testing dataset. Furthermore, the Precision and recall both had high percent performance (Figure 16) and on the confusion matrix, general performed well, but for false positive, which meant the client didn't subscribe to a term deposit, but classify as subscribed, and seemed still can be improved in future (Figure 17). Finally, in our Linear Discriminant Analysis ROC curve, it also showed that even if we raise up the thresholds, we still could not fix the False Positive, it seemed to be flatten after the true positive rate over 0.9 (Figure 18).

```
LinearDiscriminantAnalysis

-----------------------------------------------------------------------
Training accuracy: 0.9098330804248862

              precision    recall  f1-score   support

           0       0.96      0.94      0.95     29906
           1       0.51      0.62      0.56      3044

    accuracy                           0.91     32950
   macro avg       0.74      0.78      0.75     32950
weighted avg       0.92      0.91      0.91     32950

Predict Training Data with Actual Training Data:
 [[28090  1816]
 [ 1155  1889]]

-----------------------------------------------------------------------

Test accuracy: 0.9085943190094683

              precision    recall  f1-score   support

           0       0.96      0.94      0.95      7458
           1       0.51      0.62      0.56       780

    accuracy                           0.91      8238
   macro avg       0.74      0.78      0.75      8238
weighted avg       0.92      0.91      0.91      8238

Predict Test Data with Actual Test Data:
 [[7004  454]
 [ 299  481]]
```

Figure 16: Linear Discriminant Analysis classification report

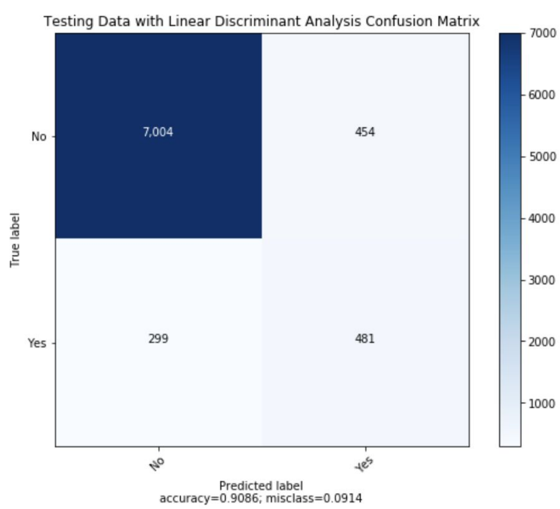


Figure 17: LDA confusion matrix

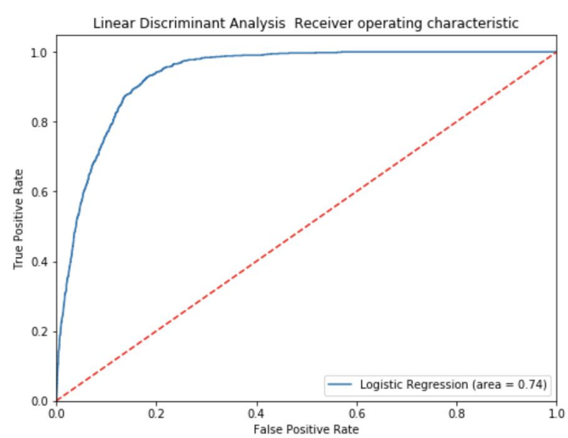
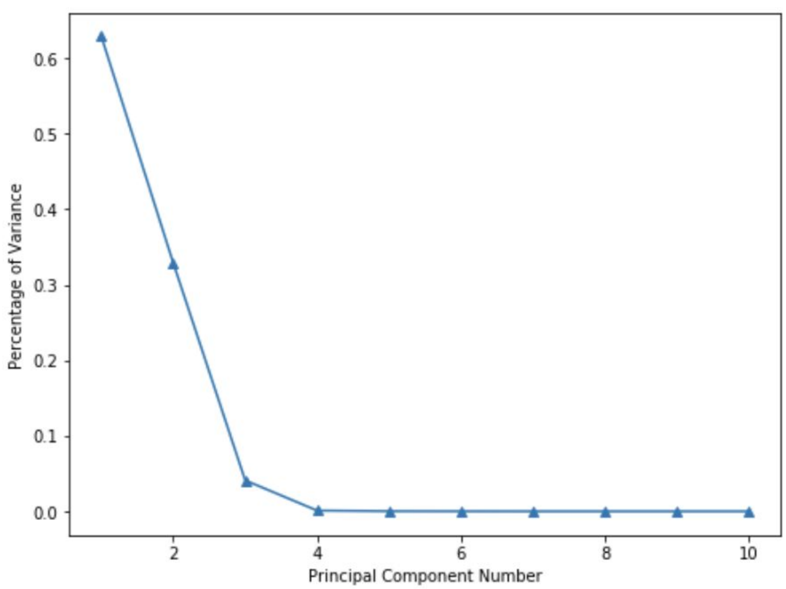


Figure 18: LDA ROC curve

Principal Components Analysis

Since we had 63 columns after we created the dummy variables in our variables, we decided to use principal components analysis, which was dimensionality reduction technique. First we did min-max normalization on our data, then we could compare with and without normalization performance. In normalization, Principal Component Number had around 30 components to satisfy 95% representation of data, but surprisingly, without normalization, Principal Component Number just had 3 components that could satisfy 99% representation of data (Figure 19, Figure 20). Because of the dimensionality much less than the other, we decided to use original data to PCA in 3 components. Second, we used the PCA matrix to redo our three classifications again. Each of the models had quite similar accuracy to using original data. For Logistic Regression, we used L1 regularization Logistic Regression and inverse of regularization strength to 10, and we got 90.6% accuracy on training data and 90.8% accuracy on test data, so we discovered that the original dataset performs better accuracy. For Naive Bayes, we got 90.6% accuracy on training data and test data. Furthermore, in the Naive Bayes model, we improved 4% accuracy on both training and testing. In the Precision and recall both raised up performance (Figure 21) and on the confusion matrix, we lowered down the true negative, which meant the client did subscribe to a term deposit, but classify as not subscribed (Figure 22). Yet, in our Naive Bayes ROC curve, it raised its threshold and became flatten after the true positive rate over 0.9, just like L1 regularization Logistic Regression (Figure 23). Last model, Linear Discriminant Analysis had similar results but not better. We got 90.4% accuracy on training data and 90.6% accuracy on test data. Based on our discovery, principal components analysis could help us use less dimensions and still predict decent accuracy for our data.



`Top three PCA can perform: 99.8662700492688 percentage of Variance`

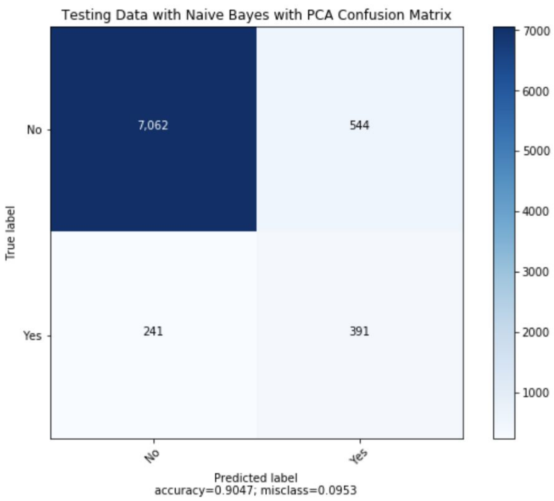
Figure 19: Principal Components Analysis

```
[[-167.333  -32.279  -52.989]
 [-142.697  -13.216   74.222]
 [-245.357   -4.822   75.045]
 ...
 [  14.921  972.316   -8.067]
 [  33.78   -42.534  -18.058]
 [  70.876  -51.641  -54.889]]

 Training data PCA can perform: 99.8662700492687 percentage of variance.

[[ 109.258  -32.37    72.513]
 [  22.209  -42.231  -18.081]
 [-209.266  -14.93    73.816]
 ...
 [-119.407  -19.853   73.406]
 [1062.155 -105.222  -58.979]
 [ -71.649  -37.078  -17.683]]

 Test data PCA can perform: 99.86856515530388 percentage of variance.
```

Figure 20: Principal Components Analysis matrix on data set

```
Naive Bayes with PCA

------------------------------------------------------------------------
Training accuracy: 0.9045220030349014

              precision    recall  f1-score   support

           0       0.97      0.93      0.95     30475
           1       0.41      0.61      0.49      2475

    accuracy                           0.90     32950
   macro avg       0.69      0.77      0.72     32950
weighted avg       0.93      0.90      0.91     32950

Predict Training Data with Actual Training Data:
 [[28287  2188]
 [  958  1517]]

------------------------------------------------------------------------
Test accuracy: 0.9047098810390871
              precision    recall  f1-score   support

           0       0.97      0.93      0.95      7606
           1       0.42      0.62      0.50       632

    accuracy                           0.90      8238
   macro avg       0.69      0.77      0.72      8238
weighted avg       0.92      0.90      0.91      8238

Predict Test Data with Actual Test Data:
 [[7062  544]
 [ 241  391]]
```

Figure 21: Naive Bayes classification report (PCA)



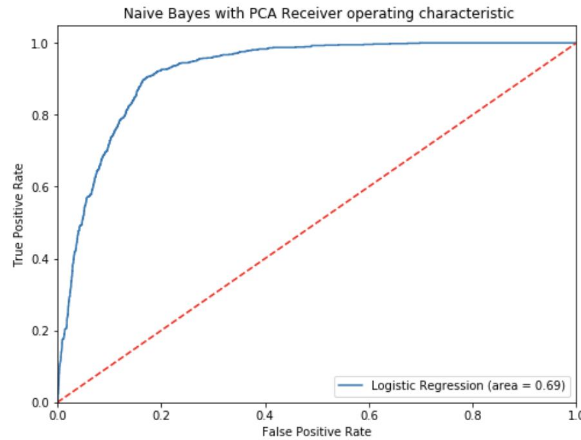


Figure 22: Naive Bayes confusion matrix(PCA) **Figure 23:** Naive Bayes ROC curve(PCA)

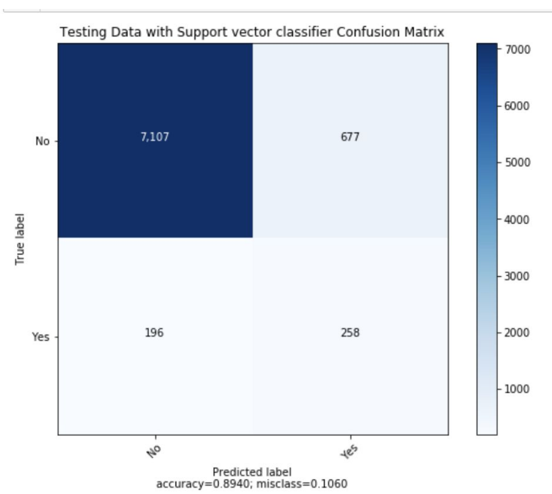
Support Vector Machines

For the Support Vector Machines result. SVM focused on separate differences in the classes of data by using separating hyperplane.Due to not being sure about our data pattern in high dimensions, We compared the training accuracy and the testing accuracy for the linear kernel SVM and radial basis function kernel SVM. Moreover, we also tested 5 different inverse of regularization strength numbers from C = [10, 1, 0.1, 0.001] in our radial basis function kernel.

First, in linear kernel SVM, C doesn't affect any accuracy, precision and recall so we chose the inverse of regularization strength number equal one. Our model performed 88.6 % accuracy in both training data and testing dataset. Similarly, in the radial basis function kernel SVM, we also chose the inverse of regularization strength number equal one, Our model performed 95.9 % accuracy in training data and 89.4% accuracy in testing dataset. Comparing two models, we discovered that SVM with a radial basis function kernel has better performance on training and test data. Furthermore, the Precision and recall both had high percent performance (Figure 24) and on the confusion matrix, general performed well, but for false positive has 677 wrong classified, which meant the client didn't subscribe a term deposit, but classify as subscribed, it had an amount of number so we have to think how to improve in future (Figure 25). Finally, in our SVM ROC curve, it raised its threshold and became flatten after the true positive rate over 0.8 (Figure 26).

```
Training accuracy: 0.9599393019726858

              precision    recall  f1-score   support

           0       1.00      0.96      0.98     30293
           1       0.68      0.95      0.79      2657

    accuracy                           0.96     32950
   macro avg       0.84      0.95      0.89     32950
weighted avg       0.97      0.96      0.96     32950

Predict Training Data with Actual Training Data:
 [[29109  1184]
 [  136  2521]]

Test accuracy: 0.894027676620539

              precision    recall  f1-score   support

           0       0.97      0.91      0.94      7784
           1       0.28      0.57      0.37       454

    accuracy                           0.89      8238
   macro avg       0.62      0.74      0.66      8238
weighted avg       0.93      0.89      0.91      8238

Predict Test Data with Actual Test Data:
 [[7107  677]
 [ 196  258]]
```

Figure 24: radial basis function kernel SVM classification report



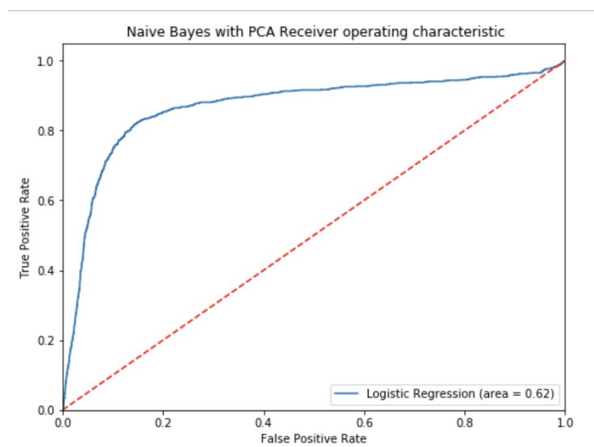


Figure 25: rbf kernel SVM confusion matrix **Figure 26:** rbf kernel SVM ROC curve(PCA)

7. Exploratory Data Analysis

7.1. K-mean Clustering – Hao

7.1.1 Background

K-mean Clustering – The clustering aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean, serving as a prototype of the cluster.

We used K-mean clustering as exploratory data analysis technique to help us understand the market segmentation. We tried to group the clients in the campaigns that are similar to each other whether in terms of background or attributes. It helped us get a meaningful intuition of the structure of the data. The Elbow method was implemented in this process to determine the best K. To evaluate the clusters, we performed Silhouette analysis on the clusters.

7.1.2 Results

According to the results of the Elbow method using inertia, showed in Figure 6, we chose k=3 as the best k value.

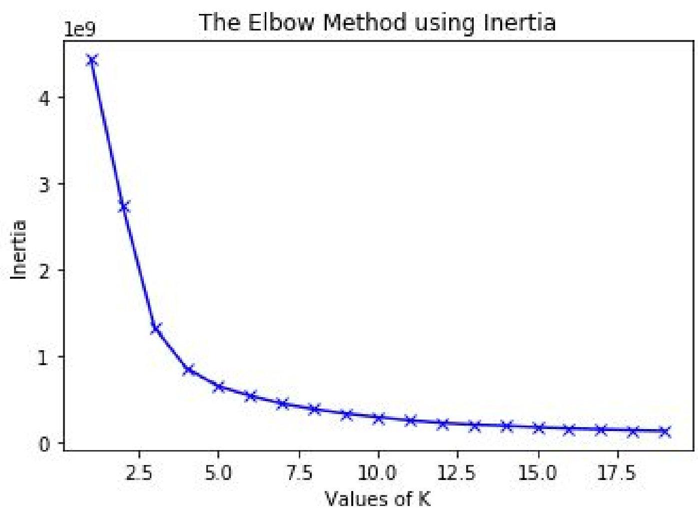


Figure 6: The Elbow method using Inertia

Three clusters were generated by k-mean clustering. The size of each cluster was unbalanced, over 84% of the data was distributed in cluster 0. By analyzing each centroid, we found the main differences between cluster 0 to the others were that it had less number of days that passed by after the client was last contacted from a previous campaign, and the contact type was all telephone. The main differences between cluster 1 to cluster 2 were that cluster 2 had higher social and economic indicators in the last contact months.

```
Size of Cluster 0 =  34859
Size of Cluster 1 =  4820
Size of Cluster 2 =  1509
```

	age	duration	campaign	pdays	previous	emp.var.rate	cons.price.idx	cons.conf.idx	euribor3m	nr.employed	job_admin.	job_blue-collar	job_entrepreneur	jc
0	41.87	314.54	1.82	6.00	1.66	-2.10	93.34	-38.33	0.99	5,029.25	0.31	0.08	0.02	
1	39.95	180.55	2.62	999.00	0.12	0.17	93.58	-40.55	3.73	5,172.46	0.25	0.23	0.04	
2	40.00	802.90	2.44	997.77	0.10	0.15	93.60	-40.84	3.69	5,170.95	0.24	0.24	0.04	

We evaluated the clusters by Silhouette analysis and found the overall mean Silhouette value was about 0.69. The result indicated that the object is well matched to its own cluster and poorly matched to neighboring clusters. The performance of each cluster was showed in Figure 7.

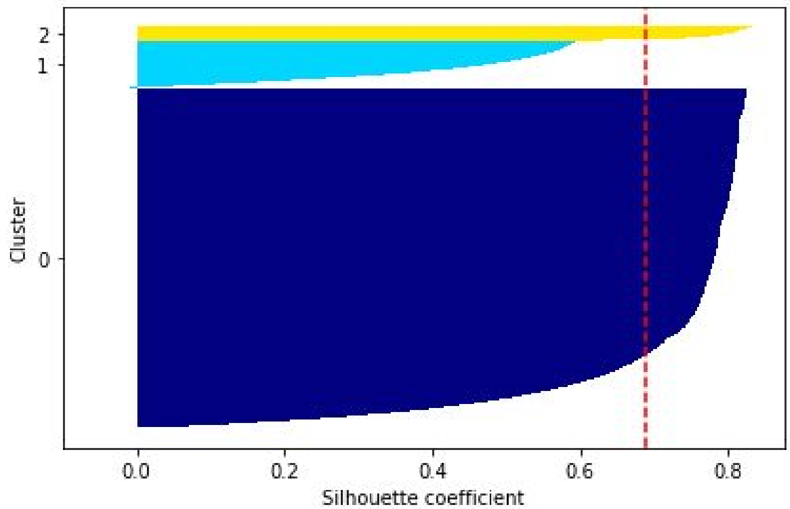


Figure 7: The Silhouette coefficient of clusters

8. Conclusion and Future Work

From statistical analysis, we know that 'duration' is the most important feature, and 'education' is the least important feature. Furthermore, May is the month with the highest number of clients contacted but the least success rate. Highest success rate is observed for the end month of the financial year as well as the calendar year. So we can say that our dataset has some kind of seasonality. In order to increase the successful rate in next campaigns, the banker should focus more on the target clients with below features: retired, without a loan, married, has a university degree, job types are 'blue-collar,' 'management,' or 'technician'.

After comparing various classification techniques, we concluded that all models had a good performance but the logistic regression with L1 regularization had the best performance among all of them that can predict the bank marketing campaigns' success with 91.11% accuracy.

From this project, we learned how banks can improve their marketing activities by focusing their revenues on certain quality customers, and how to identify market conditions that will help increase customer subscriptions about fixed products offered.

In the future work, we would like to try deep learning techniques to perform both supervised or unsupervised tasks in this dataset. Deep learning is also a class of machine learning algorithms [3]. However, it utilizes a hierarchical level of artificial neural networks to carry out the process of machine learning. The hierarchical function of deep learning systems enables machines to process data with a nonlinear approach.

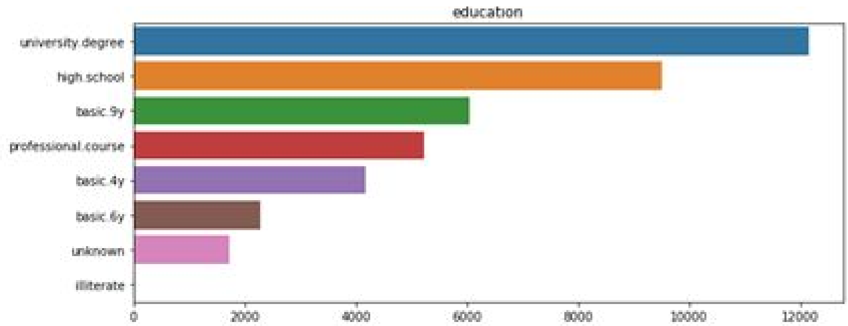
9. Reference

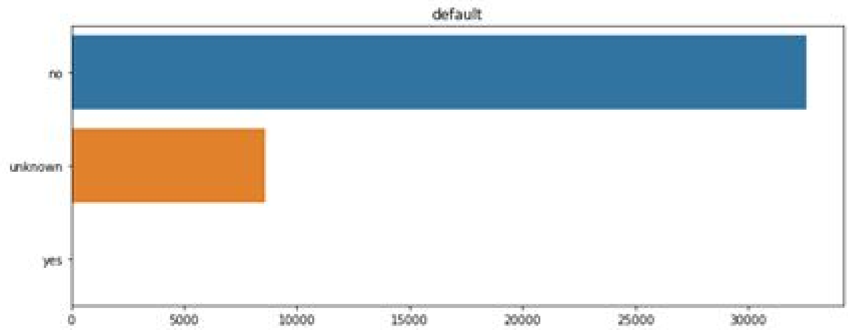
[1] UCI Machine Learning Repository: Bank Marketing Data Set.
https://archive.ics.uci.edu/ml/datasets/Bank%2BMarketing

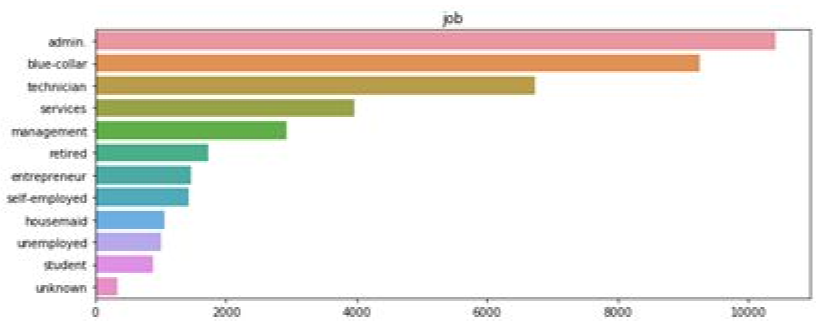
[2] An Implementation and Explanation of the Random Forest in ....
https://towardsdatascience.com/an-implementation-and-explanation-of-the-random-forest-in-python-77bf308a9b76

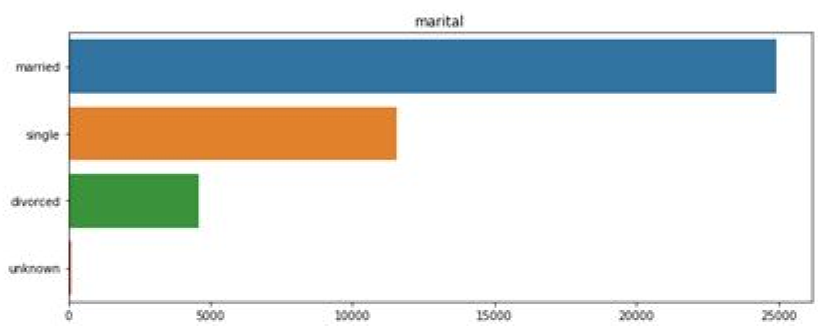
[3] Deng, L.; Yu, D. (2014). "Deep Learning: Methods and Applications". Foundations and Trends in Signal Processing. 7 (3–4): 1–199. doi:10.1561/2000000039

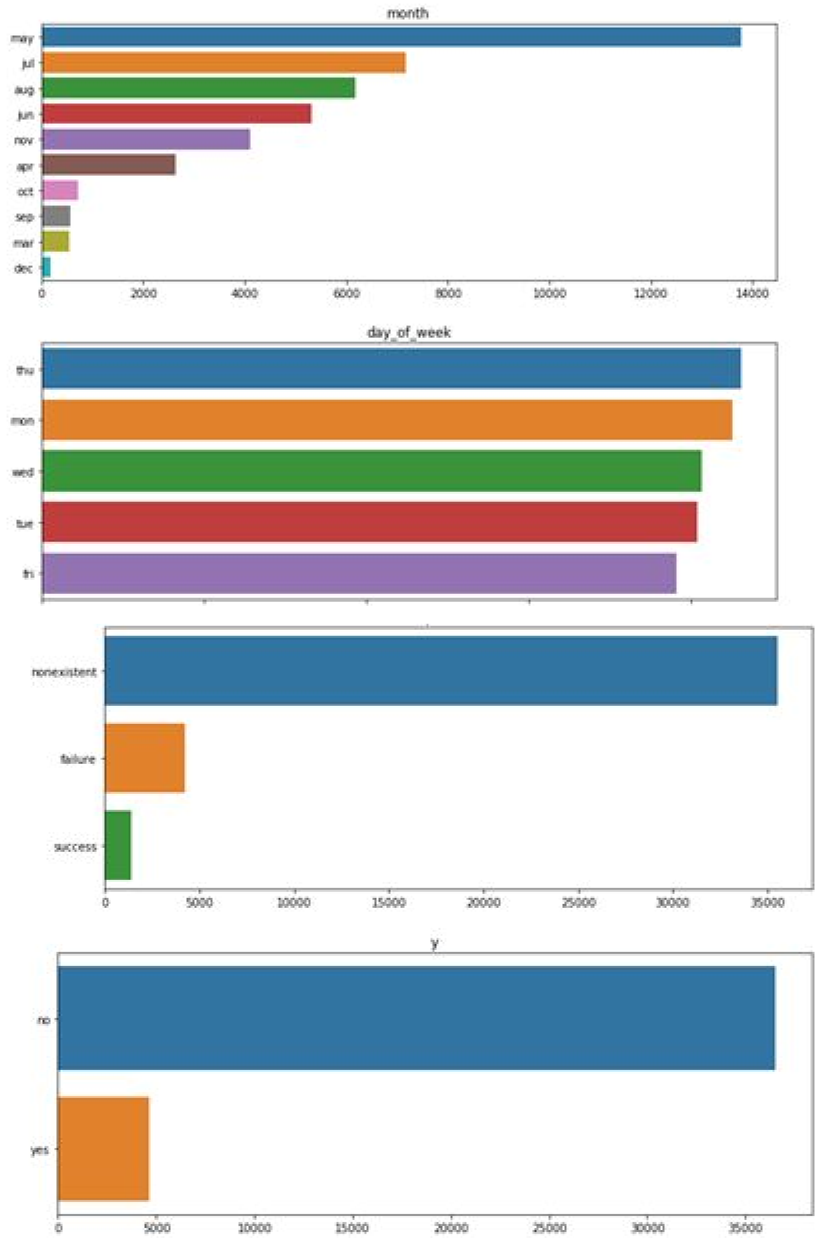
10. Appendix


education
university.degree
high.school
basic.9y
professional.course
basic.4y
basic.6y
unknown
illiterate
0
2000
4000
6000
8000
10000
12000

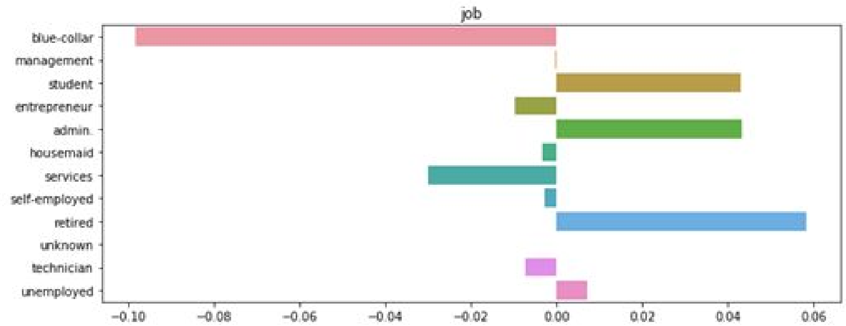

default
no
unknown
yes
0
5000
10000
15000
20000
25000
30000

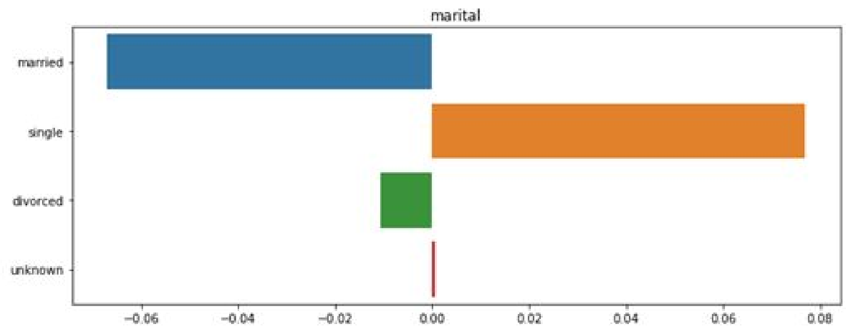

job
admin.
blue-collar
technician
services
management
retired
entrepreneur
self-employed
housemaid
unemployed
student
unknown
0
2000
4000
6000
8000
10000

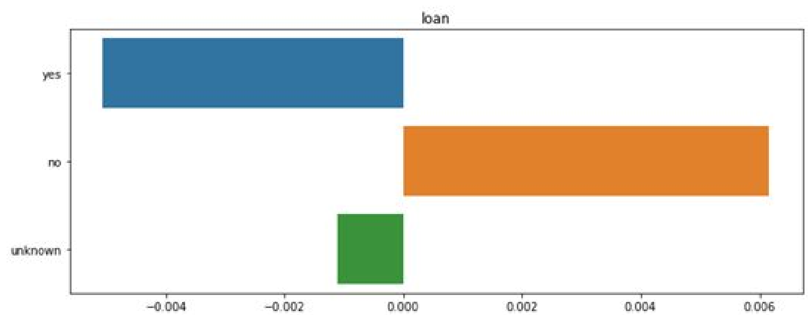

marital
married
single
divorced
unknown
0
5000
10000
15000
20000
25000

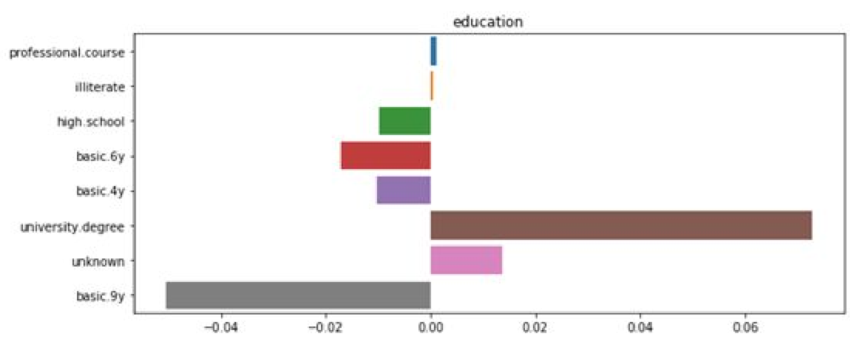

month
may
jul
aug
jun
nov
apr
oct
sep
mar
dec
0
2000
4000
6000
8000
10000
12000
14000
day_of_week
thu
mon
wed
tue
fri
nonexistent
failure
success
0
5000
10000
15000
20000
25000
30000
35000
y
no
yes
0
5000
10000
15000
20000
25000
30000
35000

A-1


job
blue-collar
management
student
entrepreneur
admin.
housemaid
services
self-employed
retired
unknown
technician
unemployed
−0.10
−0.08
−0.06
−0.04
−0.02
0.00
0.02
0.04
0.06


marital
married
single
divorced
unknown
−0.06
−0.04
−0.02
0.00
0.02
0.04
0.06
0.08


loan
yes
no
unknown
−0.004
−0.002
0.000
0.002
0.004
0.006


education
professional.course
illiterate
high.school
basic.6y
basic.4y
university.degree
unknown
basic.9y
−0.04
−0.02
0.00
0.02
0.04
0.06

```
95% of people having age are less than equal to 58.0
96% of people having age are less than equal to 59.0
97% of people having age are less than equal to 59.0
98% of people having age are less than equal to 62.0
99% of people having age are less than equal to 71.0
100% of people having age are less than equal to 98.0
IQR 15.0
```

```
Total number for Clients age < 30 is: 5669, Success rate: 0.1626389133886047
Total number for Clients age 30-45 is: 22585, Success rate: 0.0959929156519814
Total number for Clients age 40-60 is: 11741, Success rate: 0.09181500723958777
Total number for Clients age >60 is: 1193, Success rate: 0.3956412405699916
```

A-3

	Job	Total people	Success rate
0	Blue-collar	9254	0.068943
1	Management	2924	0.112175
2	Technician	6743	0.108260
3	Admin	10422	0.129726
4	Services	3969	0.081381
5	Retired	1720	0.252326
6	Self-employed	1421	0.104856
7	Entrepreneur	1456	0.085165
8	Unemployed	1014	0.142012
9	Housemaid	1060	0.100000
10	Unknown	330	0.112121
11	Student	875	0.314286

A-4

default

```
df.default.value_counts()
```

```
no         32588
unknown     8597
yes            3
Name: default, dtype: int64
```

A-5

contact

```
df.contact.value_counts()
```

```
cellular     26144
telephone    15044
Name: contact, dtype: int64
```

month

```
df.month.value_counts()
```

```
may    13769
jul     7174
aug     6178
jun     5318
nov     4101
apr     2632
oct      718
sep      570
mar      546
dec      182
Name: month, dtype: int64
```

A-6

```
95% of pdays are less than equal to 999.0
96% of pdays are less than equal to 999.0
97% of pdays are less than equal to 999.0
98% of pdays are less than equal to 999.0
99% of pdays are less than equal to 999.0
100% of pdays are less than equal to 999.0
IQR 0.0
```

A-7

previous

In [32]:
```
df.previous.describe()
```

Out[32]:
```
count    41188.000000
mean         0.172963
std          0.494901
min          0.000000
25%          0.000000
50%          0.000000
75%          0.000000
max          7.000000
Name: previous, dtype: float64
```

In [33]:
```
for x in range(95, 101 , 1):
    print("{}% of previous values less than equal to {}".fo
iqr = df.previous.quantile(0.75) - df.previous.quantile(0.2
print('IQR {}'.format(iqr))
```

```
95% of previous values less than equal to 1.0
96% of previous values less than equal to 1.0
97% of previous values less than equal to 1.0
98% of previous values less than equal to 2.0
99% of previous values less than equal to 2.0
100% of previous values less than equal to 7.0
IQR 0.0
```

A-8

poutcome

```
df.poutcome.value_counts()
```

```
nonexistent    35563
failure         4252
success         1373
Name: poutcome, dtype: int64
```

A-9

duration

In [30]:
```
df.duration.describe()
```

Out[30]:
```
count    41188.000000
mean       258.285010
std        259.279249
min          0.000000
25%        102.000000
50%        180.000000
75%        319.000000
max       4918.000000
Name: duration, dtype: float64
```

In [31]:
```
for x in range(95, 101 , 1):
    print("{}% of calls have duration less than equal to {}".format(x, df.dura
iqr = df.duration.quantile(0.75) - df.duration.quantile(0.25)
print('IQR {}'.format(iqr))
```

```
95% of calls have duration less than equal to 752.6500000000015
96% of calls have duration less than equal to 820.5199999999968
97% of calls have duration less than equal to 911.0
98% of calls have duration less than equal to 1052.260000000002
99% of calls have duration less than equal to 1271.1299999999974
100% of calls have duration less than equal to 4918.0
IQR 217.0
```

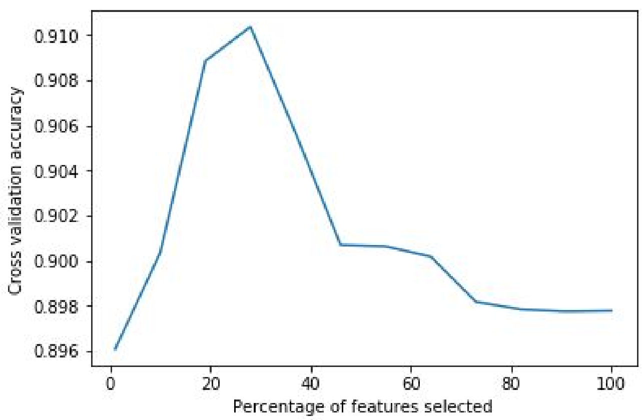
A-10

K	uniform	distance
1	0.8688	0.8688
2	0.8884	0.8688
3	0.8863	0.8759
4	0.8904	0.8773
5	0.8901	0.8792
6	0.8911	0.8801
7	0.8932	0.8837
8	0.8937	0.8837
9	0.894	0.8861
10	0.8929	0.8869
11	0.8939	0.8877
12	0.8946	0.8882
13	0.8944	0.8898
14	0.8943	0.8895
15	0.8939	0.8905
16	0.8951	0.8905
17	0.8952	0.8909
18	0.8937	0.8906
19	0.894	0.8906
20	0.894	0.8912

Optimal percentile of features:28

Optimal number of features:17

	Features	Scores
0	1	291.441870
1	4	409.696935
2	5	437.507573
3	8	723.376154
4	9	405.683607
5	15	272.223981
6	18	312.275662
7	35	252.497151
8	43	242.012623
9	44	421.132153
10	45	180.276193
11	47	211.326077
12	50	703.212583
13	51	234.198403
14	53	620.547864
15	54	470.948131
16	61	169.229836
17	62	3186.827393

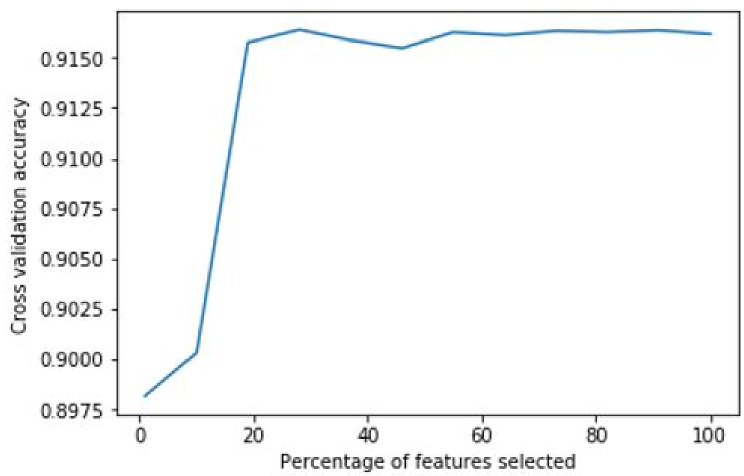


B-1

```
Fitting 5 folds for each of 3000 candidates, totalling 15000 fits
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
Wall time: 31min 51s
[Parallel(n_jobs=1)]: Done 15000 out of 15000 | elapsed: 31.9min finished
({'criterion': 'gini',
  'max_depth': 7,
  'min_samples_leaf': 23,
  'min_samples_split': 2},
 0.916176024279211)
```

```
Optimal percentile of features:28

Optimal number of features:17
```

	Features	Scores
0	1	291.441870
1	4	409.696935
2	5	437.507573
3	8	723.376154
4	9	405.683607
5	15	272.223981
6	18	312.275662
7	35	252.497151
8	43	242.012623
9	44	421.132153
10	45	180.276193
11	47	211.326077
12	50	703.212583
13	51	234.198403
14	53	620.547864
15	54	470.948131
16	61	169.229836



Optimal percentile of features:64

Optimal number of features:40

	Features	Scores
0	1	291.441870
1	2	12.299755
2	3	124.243389
3	4	409.696935
4	5	437.507573
5	6	59.105282
6	7	8.640747
7	8	723.376154
8	9	405.683607
9	10	25.212756
10	11	133.267115
11	12	8.599567
12	15	272.223981
13	17	35.010179
14	18	312.275662
15	20	5.759002
16	22	3.254310
17	23	26.434300
18	24	75.335470
19	27	9.254294
20	28	61.666072
21	32	53.222436
22	33	8.965226
23	34	66.775188
24	35	252.497151
25	43	242.012623
26	44	421.132153
27	45	180.276193
28	46	3.523233
29	47	211.326077
30	48	35.428337
31	50	703.212583
32	51	234.198403
33	52	3.755589
34	53	620.547864
35	54	470.948131
36	56	12.917707
37	60	36.261668
38	61	169.229836
39	62	3186.827393

Ridge Classifier

alpha	acc_train	acc_5cv
0.010	0.9079	0.9076
1.062	0.9078	0.9078
2.114	0.9079	0.9074
3.166	0.9079	0.9075
4.218	0.9078	0.9075
5.271	0.9079	0.9073
6.323	0.9079	0.9073
7.375	0.9079	0.9071
8.427	0.9075	0.9069
9.479	0.9075	0.9067
10.531	0.9072	0.9067
11.583	0.9071	0.9062
12.635	0.9072	0.9062
13.687	0.9071	0.9062
14.739	0.9069	0.9063
15.792	0.9067	0.9063
16.844	0.9066	0.9064
17.896	0.9066	0.9064
18.948	0.9067	0.9065
20.000	0.9066	0.9066