Introduction

The data is related to direct marketing campaigns of a Portuguese banking institution. The marketing campaigns were based on phone calls. Often, more than one contact with the same client was required, in order to assess if the product (bank term deposit) would be ('yes') or not ('no') subscribed. The dataset helps to recognize the attributes that influence the decision of customers to subscribe for the bank term deposit.

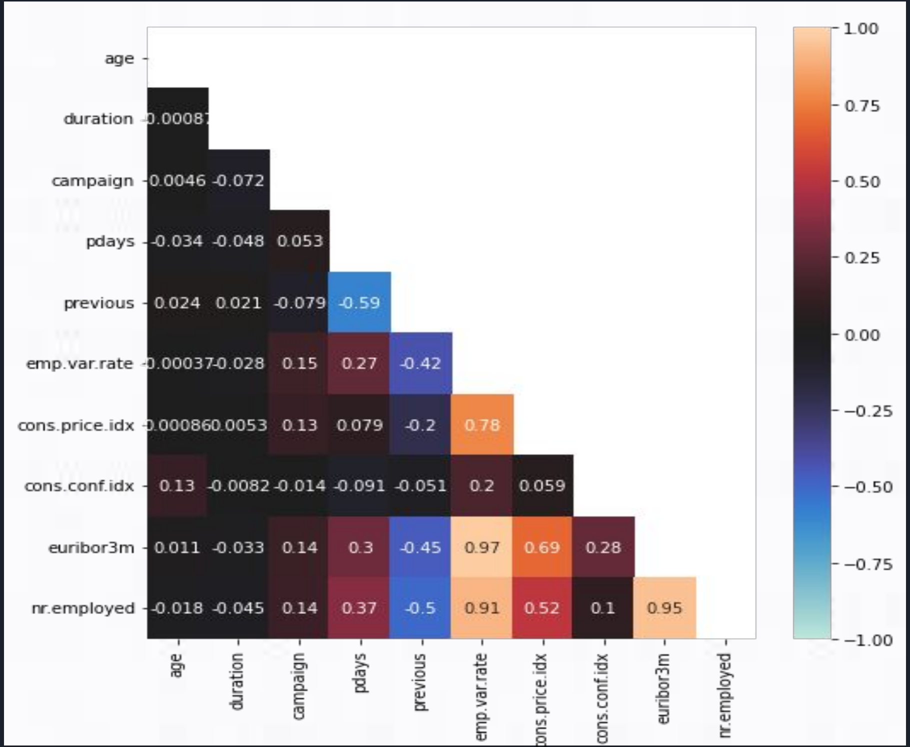
The raw dataset comprises of 21 attributes and 41,188 records.

Our goal was to analyze the bank marketing dataset and to predict if a customer would subscribe to a bank term deposit.

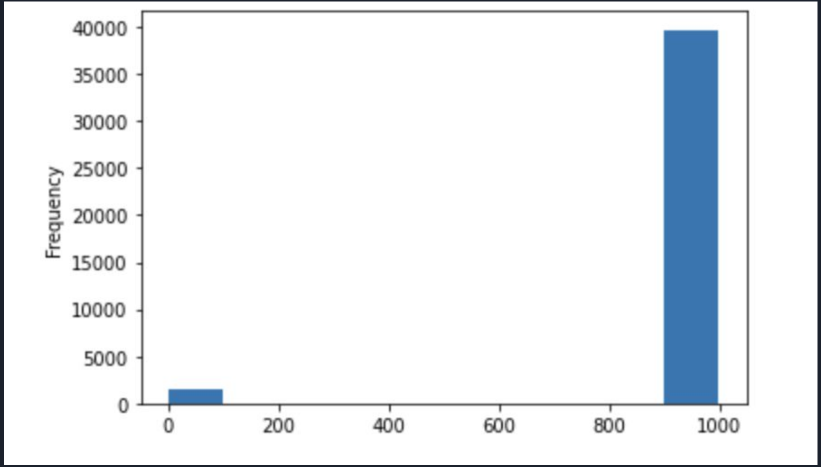
Data Pre-processing

Below are some steps we explored to clean our dataset for better accuracy:

Correlation Plot



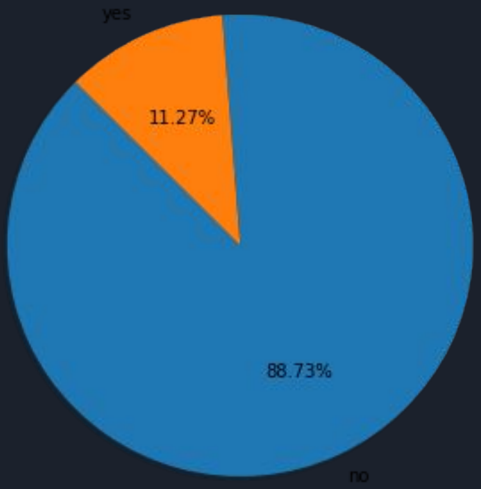
From the histogram below, you can notice that the pdays (no. of days since last contacted) have mostly the value as approximately 1000.



Exploratory Data Analysis

SMOTE

We plotted the target variable to check if our data was imbalanced. As seen in the figure below, our data was highly imbalanced so we decided to use SMOTE technique.

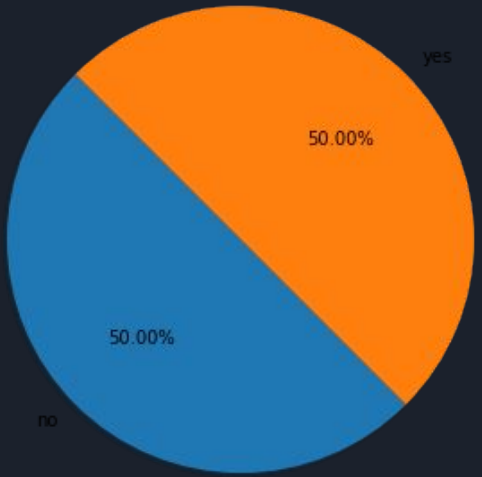


There were 3251 records classified as class 1 and 25580 classified as class 0.

Exploratory Data Analysis

Post SMOTE

After we oversampled the data using the SMOTE approach, the training datasets dimensions before the model-building phase were 51160 instances and 46 attributes.



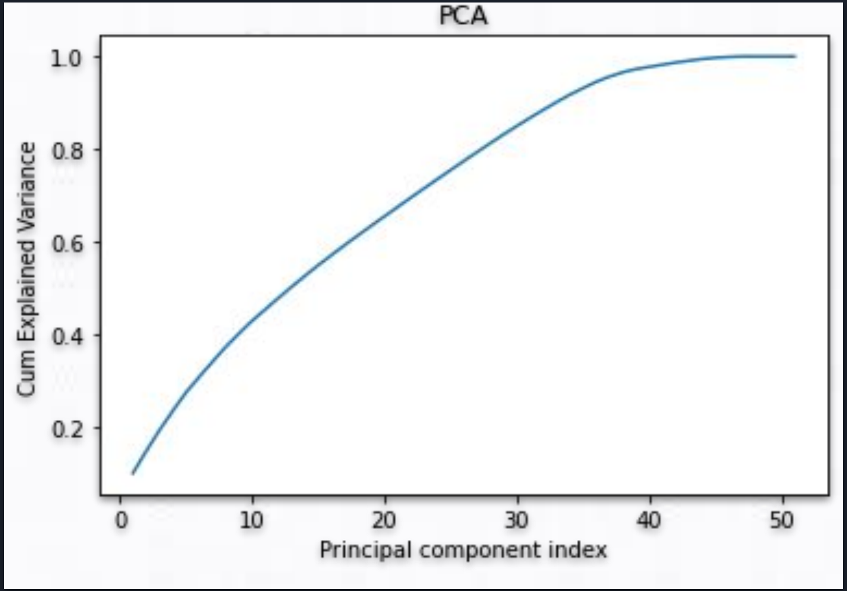
The class is now balanced with 25580 records for 1’s as well as 0’s each.

Exploratory Data Analysis

Dimension Reduction: PCA

The chart shows that the initial 45 components captured a significant amount of variance before eventually declining . While reducing the dimensions, it’s important to avoid losing too much variance.

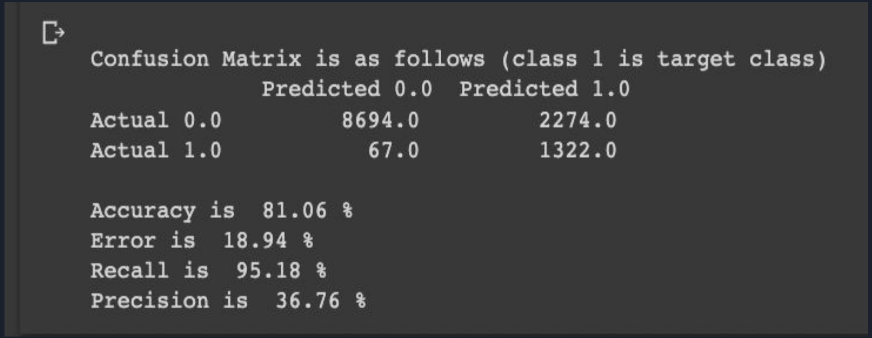
We chose to select the first 45 principal components, which captured 99% of the total explained variance, after considering the tradeoff variance between these two factors.



Logistic Regression

Initial results showed a training error of 12.62 percent and a test error of 18.72 percent.

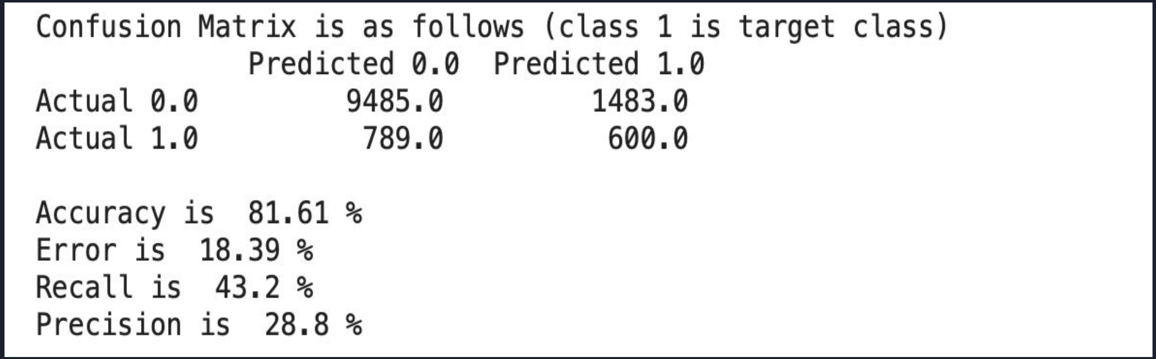
We played about with the learning parameters to try and solve the overfitting issue, and we found that tolerance = 0.0000005, learning rate = 0.00001, and max iterations = 10000 were the optimum settings for doing so.



```
Confusion Matrix is as follows (class 1 is target class)
             Predicted 0.0  Predicted 1.0
Actual 0.0          8694.0         2274.0
Actual 1.0            67.0         1322.0

Accuracy is  81.06 %
Error is  18.94 %
Recall is  95.18 %
Precision is  36.76 %
```

Naive Bayes

As Naive Bayes is a non-parametric model, no hyperparameter tuning could be done to improve its performance.

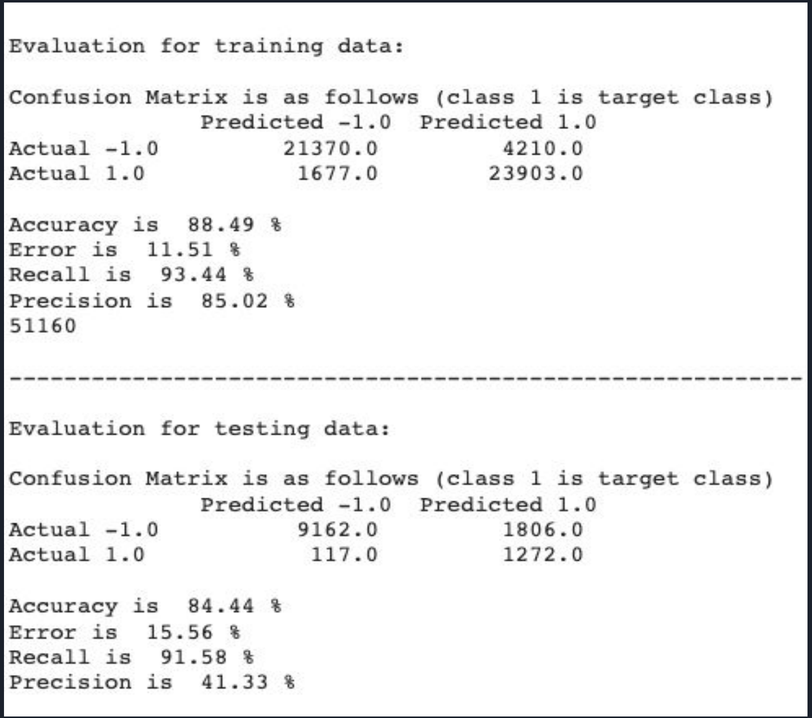


```
Confusion Matrix is as follows (class 1 is target class)
             Predicted 0.0  Predicted 1.0
Actual 0.0         9485.0         1483.0
Actual 1.0          789.0          600.0

Accuracy is  81.61 %
Error is  18.39 %
Recall is  43.2 %
Precision is  28.8 %
```

Hard Margin SVM

Initial results show a training error of 11.51 percent which determined our model was underfitting. It meant that there was a large bias and low variance.

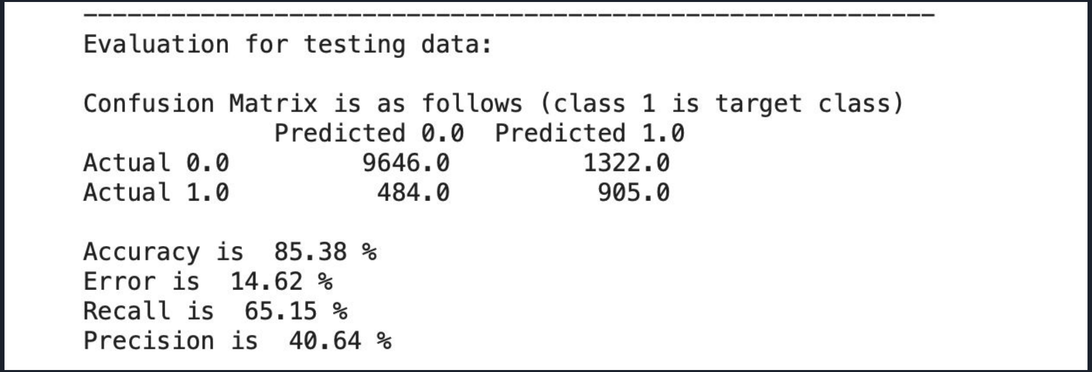
Therefore, we addressed the issue of underfitting, and the values for learning rate = 0.00001, lamda = 0.001, and n iters = 200 produced the best results for attaining the bias-variance tradeoff.



```
Evaluation for training data:

Confusion Matrix is as follows (class 1 is target class)
              Predicted -1.0  Predicted 1.0
Actual -1.0         21370.0         4210.0
Actual 1.0           1677.0        23903.0

Accuracy is  88.49 %
Error is  11.51 %
Recall is  93.44 %
Precision is  85.02 %
51160

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

Evaluation for testing data:

Confusion Matrix is as follows (class 1 is target class)
              Predicted -1.0  Predicted 1.0
Actual -1.0          9162.0         1806.0
Actual 1.0            117.0         1272.0

Accuracy is  84.44 %
Error is  15.56 %
Recall is  91.58 %
Precision is  41.33 %
```

KNN

As K-NN is a non-parametric model, no hyperparameter tuning could be done to improve its performance. The best test accuracy achieved was 85.38% for the value of k=3.



```
----------------------------------------------------------
Evaluation for testing data:

Confusion Matrix is as follows (class 1 is target class)
             Predicted 0.0  Predicted 1.0
Actual 0.0         9646.0         1322.0
Actual 1.0          484.0          905.0

Accuracy is  85.38 %
Error is  14.62 %
Recall is  65.15 %
Precision is  40.64 %
```

COMPARISON OF THE MODELS

Metrics	Logistic Regression	Naive Bayes	SVM	KNN
Accuracy	0.80	0.8161	0.8444	0.8538
Error	0.189	0.1839	0.1556	0.1462
Recall	0.95	0.432	0.9158	0.6515
Precision	0.36	0.288	0.4133	0.4064
F1 Score	0.5304	0.3456	0.5695	0.5006

Conclusion

The above table gives us evidence that KNN was the best performing model with accuracy 85.38% followed by SVM which was 84.44%. However, while running the models, we observed that KNN took approximately 2 hrs 15 mins each time we ran the code whereas **Hard Margin SVM** gave us the quickest results within approximately 3 mins. Therefore, after all, the analysis, we believe **Hard Margin SVM** was able to determine if the customer subscribed for the bank term deposit.