Abstract

Our goal was to analyze the bank marketing dataset and to predict if a customer would subscribe to a bank term deposit. We tried to solve the classification problem by determining the factors that led customers to subscribe/not subscribe to the schema.

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.

Data Description

Link to UCI repository:

UCI Machine Learning Repository: Bank Marketing Data Set

The raw dataset comprises 21 attributes and 41,188 records. The different data fields in the dataset are as follows:

Categorical Variables	**Numerical Variables**
Job	Age
Contact	Campaign
Day of the week	Consumer Confidence Index
Default values	Consumer Price Index
Education	Duration
Housing Loan	Employment Variation Rate
Personal Loan	European Bank 3 month rate
Marital Status	Number of Employees
Month of the Year	Pdays (No. of days since last contacted)
Y (subscribed or not subscribed)	Previous (No. of contacts performed)
Poutcome (previous campaign outcome)	

Exploratory Data Analysis

Data exploration and visual representation are crucial tools for telling a story and making it simple to understand by showing trends and outliers. Removing extraneous noise offers us a clear picture and makes it possible for us to make sense of the data and develop conclusions.

To acquire unique features in the model we check for unique values column wise to understand the nature of the dataset.

```
df.nunique()
```

```
age                78
job                12
marital             4
education           8
default             3
housing             3
loan                3
contact             2
month              10
day_of_week         5
duration         1544
campaign           42
pdays              27
previous            8
poutcome            3
cons.price.idx     26
cons.conf.idx      26
euribor3m         316
nr.employed        11
y                   2
dtype: int64
```

As we need to clean the data it's crucial to analyze null values in each of the columns to make sure that there are no null values as it would lead to poor analysis.

```
df.isna().sum()
```

```
age               0
job               0
marital           0
education         0
default           0
housing           0
loan              0
contact           0
month             0
day_of_week       0
duration          0
campaign          0
pdays             0
previous          0
poutcome          0
cons.price.idx    0
cons.conf.idx     0
euribor3m         0
nr.employed       0
y                 0
dtype: int64
```

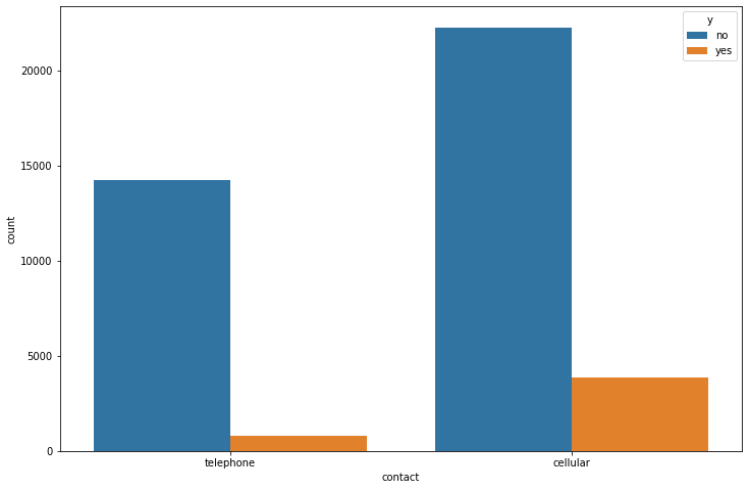
As we can see there are no null values. Hence, we don't have to do any changes in this dataset.

Let's use pairplot to plot some data:

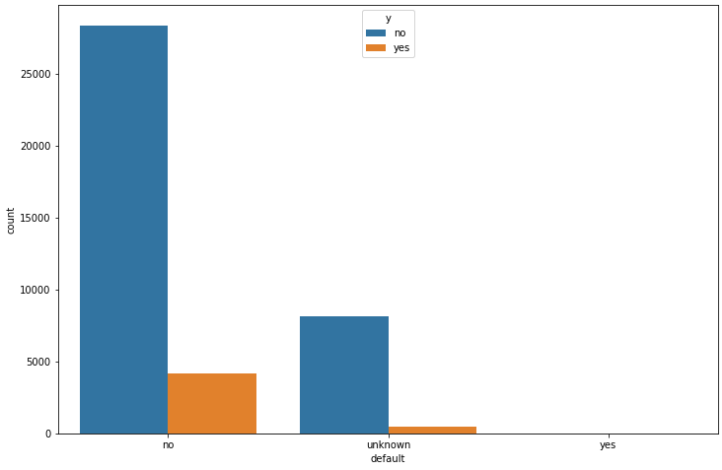
The variances in each plot are shown in the image below. The graphs are in matrix format, where the x-axis is represented by the row name and the y-axis by the column name.

Below are some of the graphical representations to visualize the dataset:

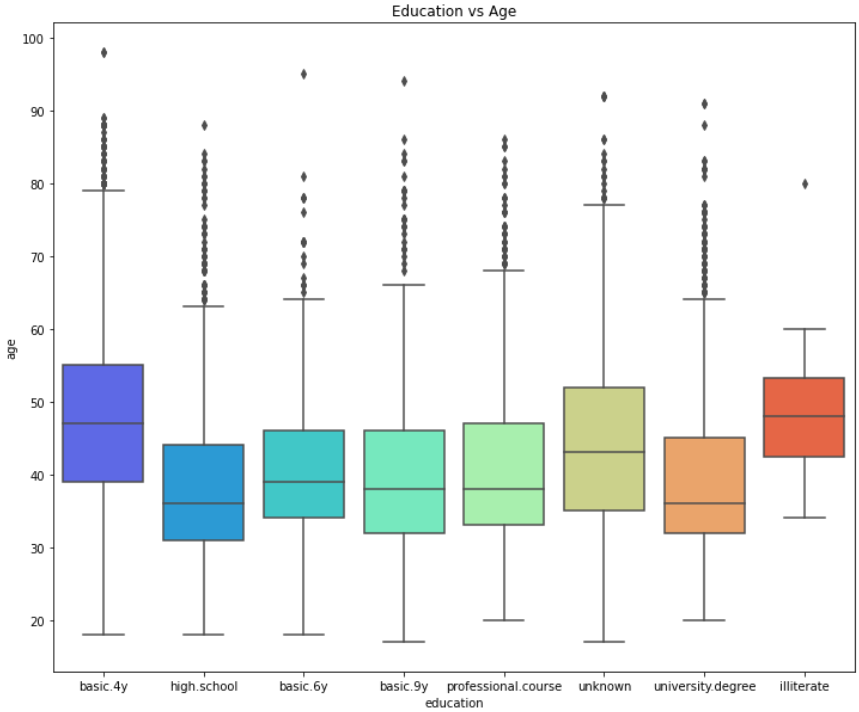
1. We plotted the contact method of clients that were contacted. We observed that 14.7% of customers subscribed when contacted via cellular and 5.2% of customers subscribed when contacted via telephone.



2. We plotted the count of clients that fell into the defaulter's category. We observed that negligible customers belonged to the default category.



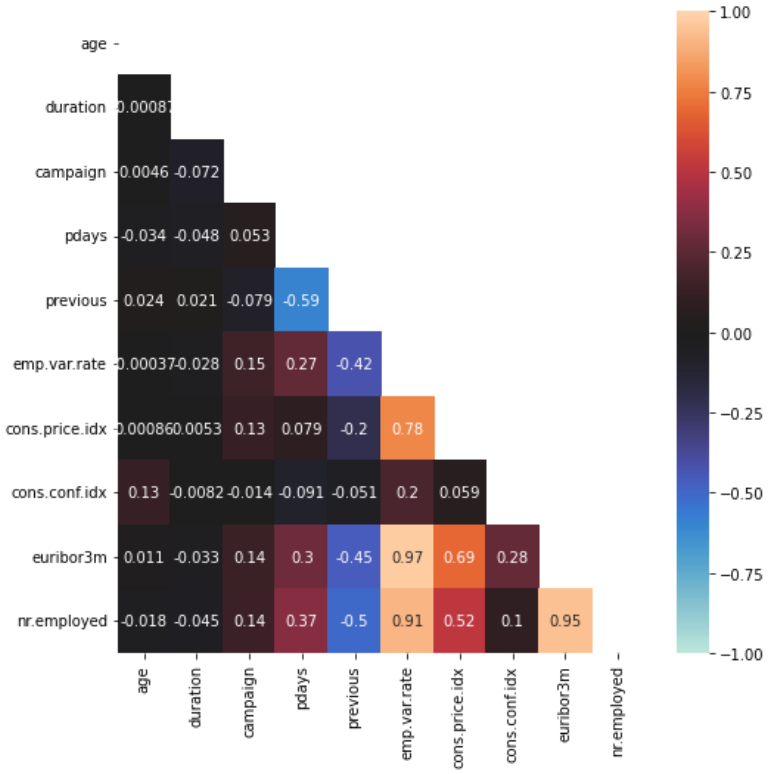
3. We plotted a box plot for the age of the customers with respect to their education. We observed that the median age group of targeted customers is between 35-40 years.



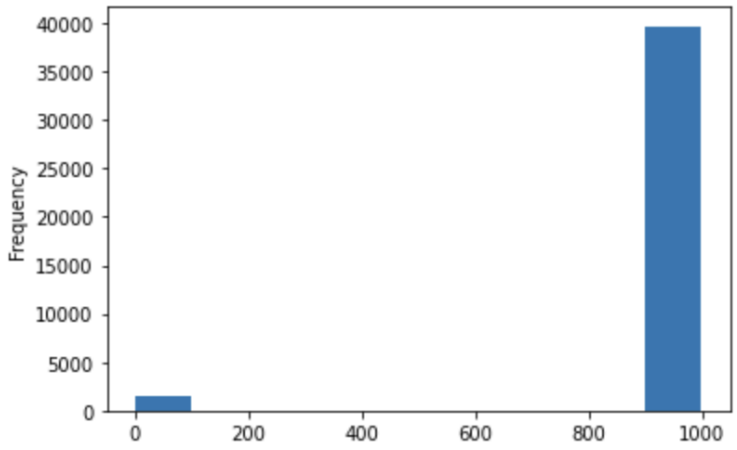
Data Pre-processing

Correlation Plot

From the heatmap below, you can notice that emp.var.rate is highly correlated to euribor 3m and nr.employed amongst all the variables. Therefore, we would like to remove emp.var.rate as they express multicollinearity.



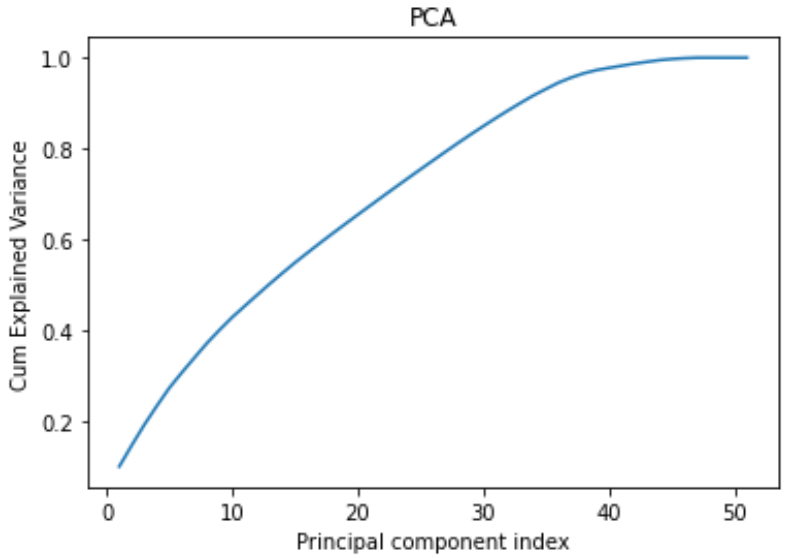
From the histogram below, you can notice that the pdays have mostly the value as approximately 1000. Hence, it won't play a significant role in the classification of the target.



Data Reduction

To reduce the computational complexity of our dataset, especially for the K Nearest Neighbours and Hard Margin SVM that requires a lot of computing, we are going to perform the principal component analysis before applying any learning algorithms to our dataset.

First we normalized data by StandardScaler(), performed the Principal Component Analysis, found the dimensions with the greatest variance to maximize the distinction of different data points out of it. We calculated cumulative explained variance and visualized it using a line chart in order to get a better idea of how many principal components to choose.

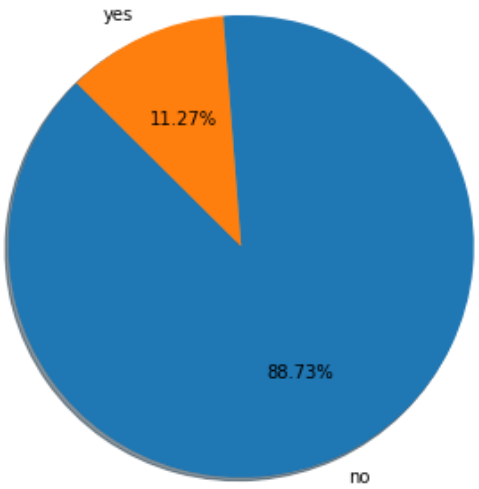


The aforementioned chart shows that the initial 45 components captured a significant amount of variance before eventually declining . While reducing the dimensions, it's important to avoid losing too much variance. We chose to select the first 45 principal components, which captured 99% of the total explained variance, after considering the tradeoff variance between these two factors.

Feature Engineering

Dummy variables were created for the categorical variables in the dataset to ensure that data is encoded in binary format to train classification models during the model construction phase.

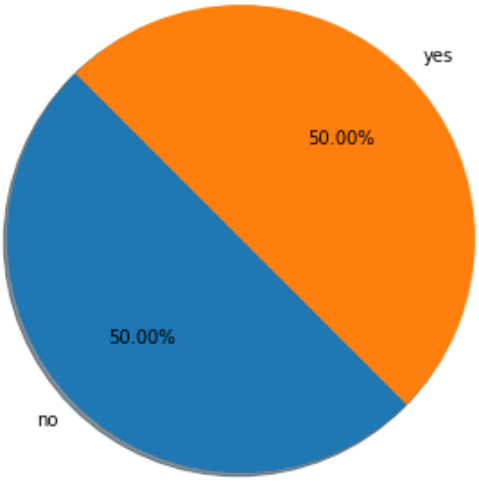
We plotted the target variable to check if our data was imbalanced. As seen in the figure below, our data was highly imbalanced so we decided to use SMOTE technique.



There were 3251 records classified as class 1 and 25580 classified as class 0. After we oversampled the data using the SMOTE approach, the training dataset's dimensions before the model-building phase were 51160 instances and 46 attributes.

Post SMOTE: The class is now balanced with 25580 records for 1's as well as 0's each.

Graphical Representation of the class distribution after Smote



Methods

1. Logistic Regression

   It's a parametric classification model that depends on a particular model linking the predictor factors with the target variable, making the output variable categorical. After applying a threshold cutoff to the probabilities for classification into either class, the output is an estimate of the odds of belonging to each class. It is possible to model the outcome variable as a linear function of the predictors.

2. Naive Bayes

   Performed Gaussian Naïve bayes algorithm since its a classification model. Basically, it is the calculation of the probabilities for each hypothesis simplified to make their calculation tractable. Instead of attempting to compute each attribute value individually, P(d1, d2, d3|h), they are believed to be conditionally independent given the goal value and calculated as P(d1|h) * P(d2|H), and so on. A fitted naive Bayes model's file contains a list of probabilities.

3. SVM

   SVM uses the concept of choosing the optimum hyperplane to divide the features into various domains. The likelihood of successfully classifying the Support Vector points in their respective regions or classes increases with the distance of the points from the hyperplane. The support vector points are crucial in figuring out the hyperplane because if the vectors' positions change, so does the hyperplane's location.

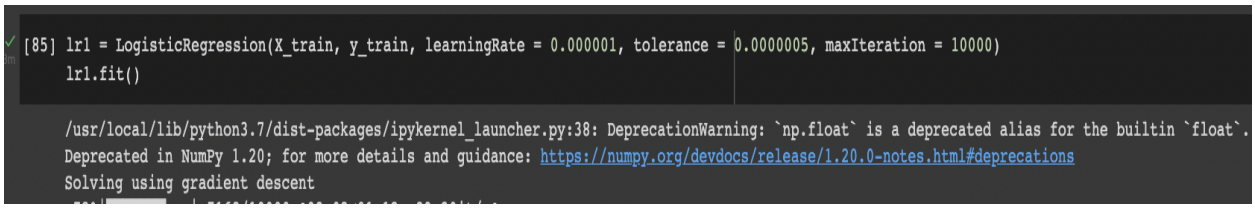
4. KNN

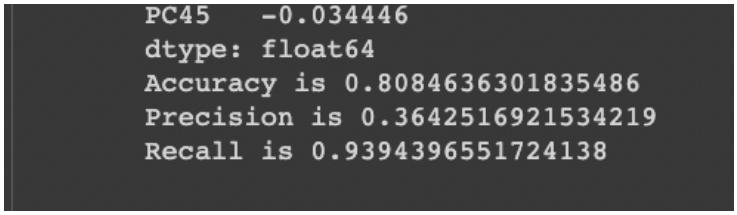
A new record from the test data is classified using similar records from the training data, which is the main concept behind the k-NN Classification method. The Euclidean distance metric was used along with the weighted distance to each neighbor (so that closer neighbors have a greater influence than neighbors which are further away). The new record is categorized as a member of the majority class of the k-neighbors using a majority decision rule after the related records are identified using the k-nearest neighbors.

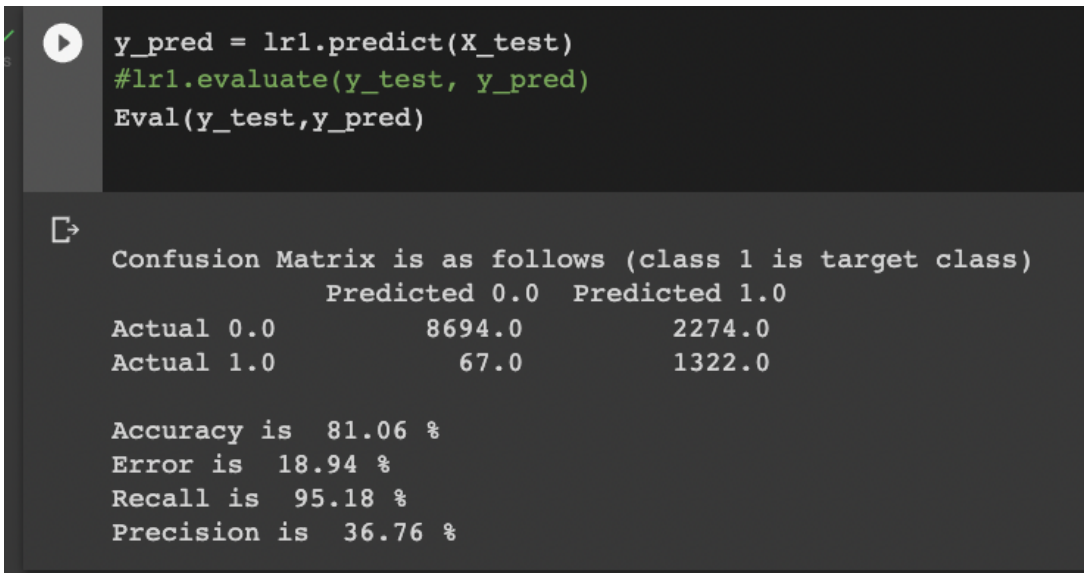
Results

1. Logistic Regression

   Initial results showed a training error of 12.62 percent and a test error of 18.72 percent with tolerance = 0.0000005, learning rate = 0.000001, and maxIterations =10000. This allowed us to see that our model was overfitting, which meant that the variance and bias were low. We played about with the learning parameters to try and solve the overfitting issue, and we found that tolerance = 0.65, learning rate = 0.00001, and max iterations = 400 were the optimum settings for doing so. As a consequence, the logistic regression model had a training accuracy of 81.93 percent and a test accuracy of 80.81 percent.







2. 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 %
```

3. Hard Margin SVM

First results show a training error of 11.51 percent and a test error of 15.56 percent with learning rate = 0.00001, lamda = 0.001, and n iters = 200. This allowed us to see that our model was underfitting, which meant that there was a large bias and low variance, indicating that the model did not generalize the data well. We played around with the learning settings in order to address 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. The SVM model had a 11.51 percent training error and a 15.56 percent test error as a result.

```
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 %
```

4. 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 %
```

Discussion

Our goal for this project was to classify if a customer subscribed for the bank term deposit. We performed a few EDA steps. Generated a correlation plot to check for multicollinearity and drop columns that were correlated to avoid redundancy. Secondly, created dummy variables to convert non-numeric columns to binary numeric. Then, used PCA to reduce the dimensions of the dataset and not overfit the model. Moreover, the dataset was highly imbalanced with the class of interest 'yes' at only 11.27%. Therefore, we performed smote to handle the issue before training the dataset with different classification models.

Below is the table of the models that we used and a comparison of their respective accuracy, error, recall, precision, and F1 Score:

	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

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 2hrs 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.