Analysis Of Banking Data With Help Of Machine Learning

Alhamza Ibrahim, Bilal Yıldız

Problem Statementment

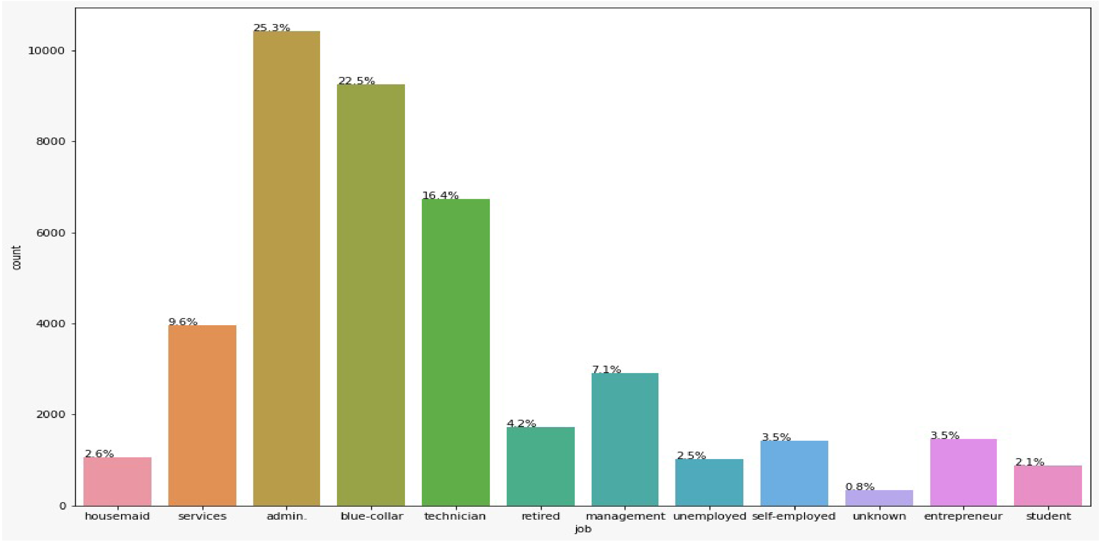
A portugese Bank wants to sell it's term deposit product to customers and before launching the product they want to develop a model which help them in understanding whether a particular customer will buy their product or not (based on customer's past interaction with bank or other Financial Institution).

Data Set

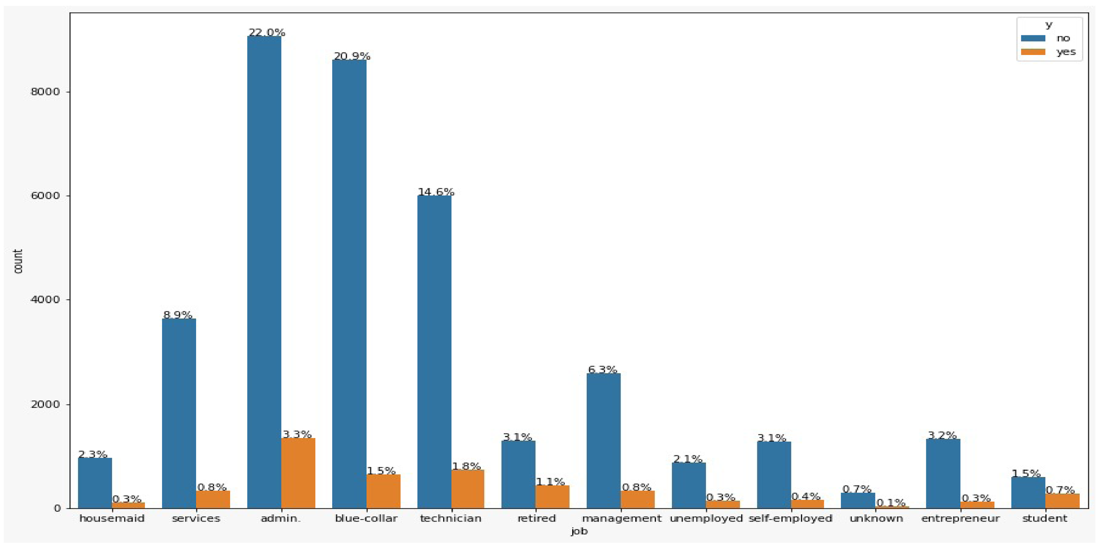
The data is related to direct marketing campaigns of a Portuguese banking institution. The data set contains 21 Columns and more than 40k rows. Most of The columns are self explainatory (age ,Job,Education,etc…) except for the last five columns which are :
[emp.var.rate: employment variation | cons.price.idx: consumer price index | cons.conf.idx: consumer confidence index | euribor3m: euribor 3 month rate | nr.employed: number of employees]

Insights from Data

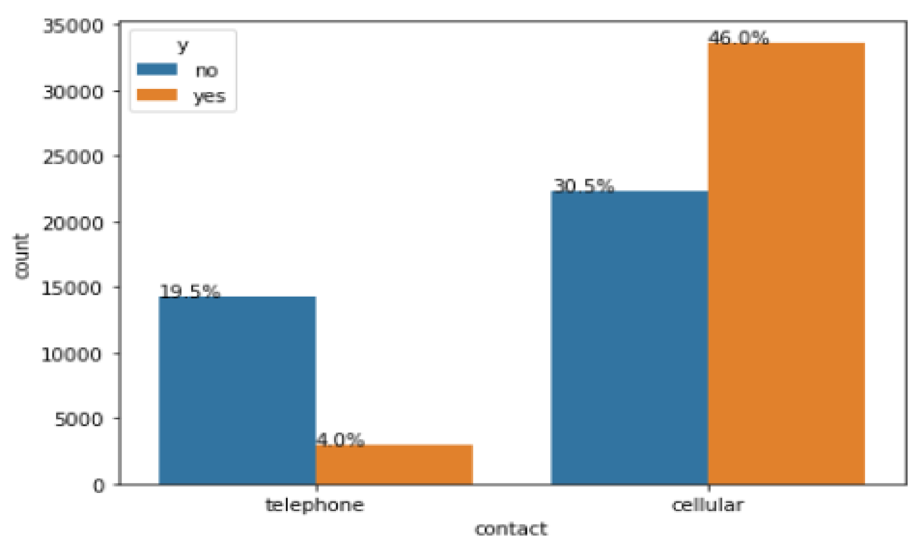
This bar chart shows the number of customers based on their Profession .We can see that the most subscribed to the product are the jobs "Admins" and "Technicians".



This chart shows the Professions and the answer of people wether they will buy the product or not we can see clearly here that admin and blue-collar are almost 50% of the whole bank costumers.

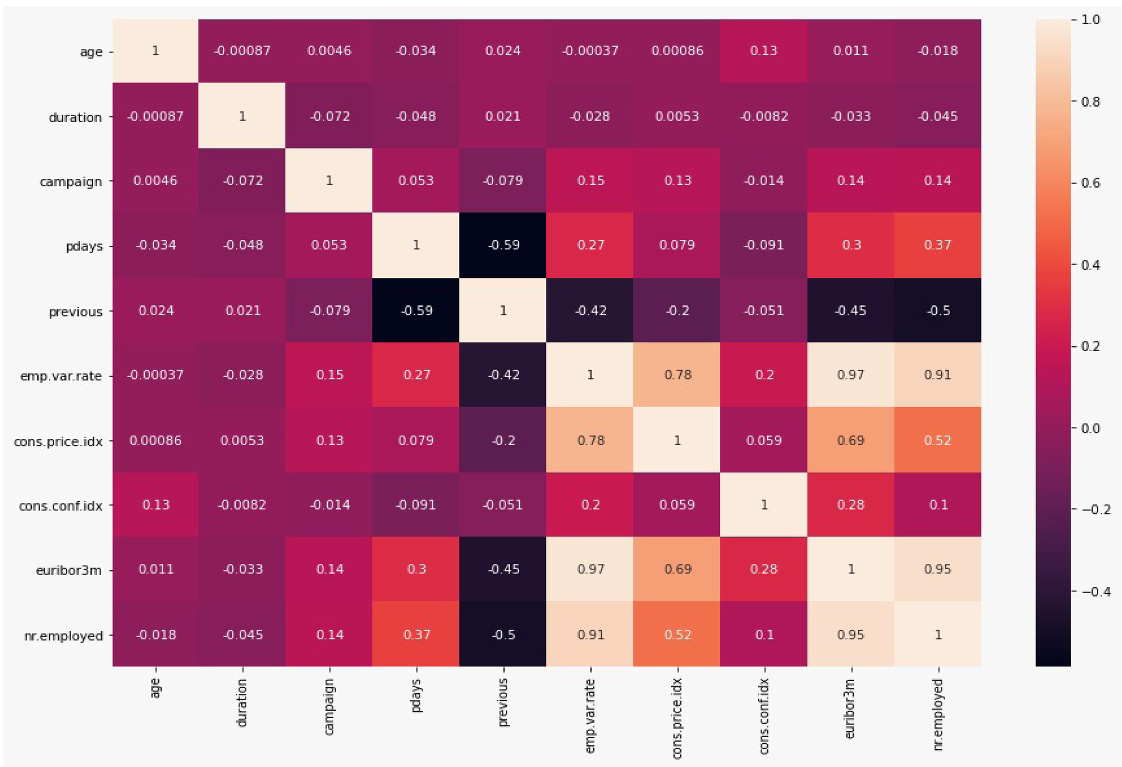


People who use cellular are almost 10 times more likely to subscribe to the product than people with telephone. (46% vs 4%)

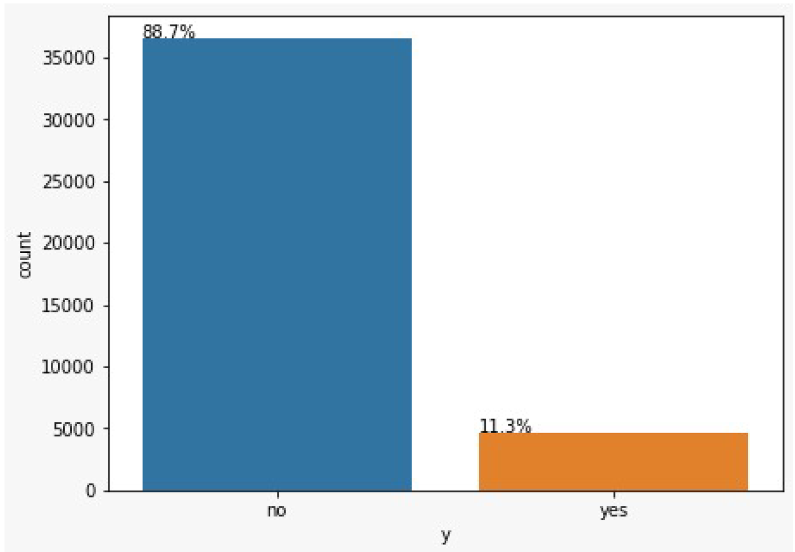


Correlation Heatmap

The featrues [emp.var.rate, euribor3m, nr.employed and cons.price.index] have very high correlation. But we can also notice that [Euribor3m with nr.employed] and [emp.var.rate with nr.employed] with the highest correlation with more than 0.9 value.



Data Pre Processing



	count	percentage
no	36548	0.887346
yes	4640	0.112654

The original DataSet Shape (Imbalanced Data)

	count	percentage
no	36548	0.5
yes	36548	0.5

Data Shape after applying oversampling

As from the plot, we can see that the majority of datapoints belong to the class [No] with 88.7% and teh minority belongs to the calss [YES] with 11.3% .The ratio of No:Yes is almost 8:1 which means the data is imbalanced

To overcome this problem we used an oversampling tecnique called SMOTE to generate synthetic instances. Now the final shape of our data became (36548*2)

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 41188 entries, 0 to 41187
Data columns (total 21 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   age             41188 non-null  int64  
 1   job             41188 non-null  object 
 2   marital         41188 non-null  object 
 3   education       41188 non-null  object 
 4   default         41188 non-null  object 
 5   housing         41188 non-null  object 
 6   loan            41188 non-null  object 
 7   contact         41188 non-null  object 
 8   month           41188 non-null  object 
 9   day_of_week     41188 non-null  object 
 10  duration        41188 non-null  int64  
 11  campaign        41188 non-null  int64  
 12  pdays           41188 non-null  int64  
 13  previous        41188 non-null  int64  
 14  poutcome        41188 non-null  object 
 15  emp.var.rate    41188 non-null  float64
 16  cons.price.idx  41188 non-null  float64
 17  cons.conf.idx   41188 non-null  float64
 18  euribor3m       41188 non-null  float64
 19  nr.employed     41188 non-null  float64
 20  y               41188 non-null  object 
dtypes: float64(5), int64(5), object(11)
memory usage: 6.6+ MB
```

We notice that there isn't any Null-containing columns but we have 11 object-containing columns (Non-Numerical) Four of them (The last 4 columns) were YES or NO columns, we have encoded them with help of Ordinal Encoder and for the rest (Multivatiant) we are have used One-Hot-Encoder.

The final data shape after converting all the columns into numecal values became (73096 rows × 48 columns) which led to another problem ….. Dimontionality Curse !!

PCA for Dimontionality reduction: we have used the PCA algorithm as a step in the model-building-pipline to overcome the Dimontionality problem

We have found some "unknown" values in six of the columns (from one to seven) so we replaced them with the most frequent values from each column with help of the "SimpleImputer"

```
imp = SimpleImputer(missing_values= "unknown" , strategy='most_frequent')
mostfrequentdf = pd.DataFrame(imp.fit_transform(mydf))
```

```
no unknowns detected
unknown detected in colum num 1
unknown detected in colum num 2
unknown detected in colum num 3
unknown detected in colum num 4
unknown detected in colum num 5
unknown detected in colum num 6
no unknowns detected
no unknowns detected
no unknowns detected
no unknowns detected
no unknowns detected
no unknowns detected
no unknowns detected
no unknowns detected
no unknowns detected
no unknowns detected
no unknowns detected
no unknowns detected
no unknowns detected
no unknowns detected
```

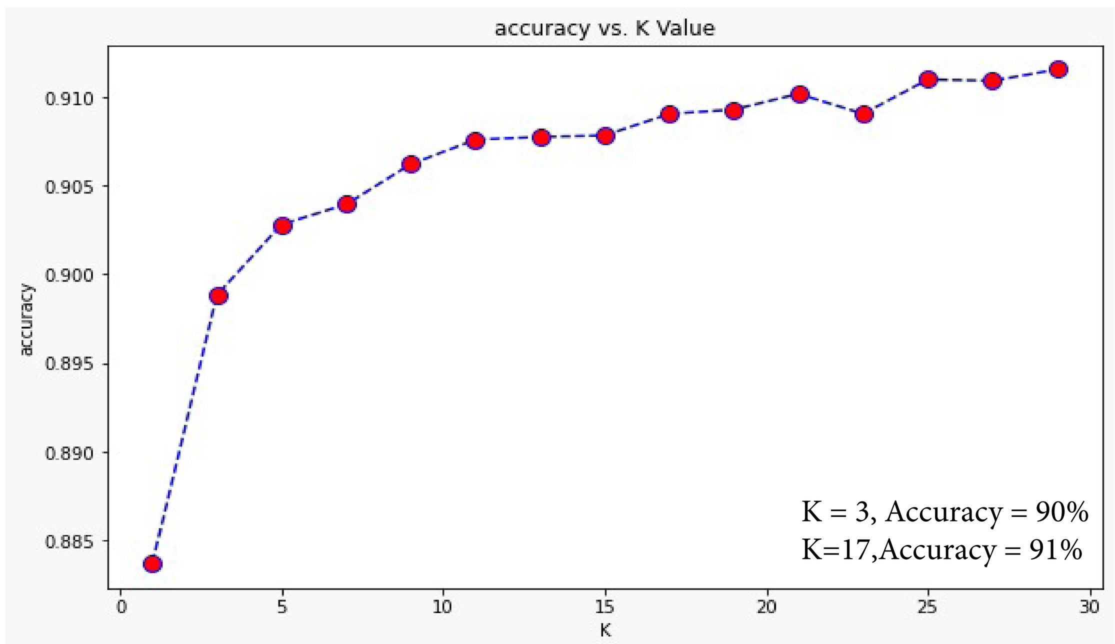
Building the models

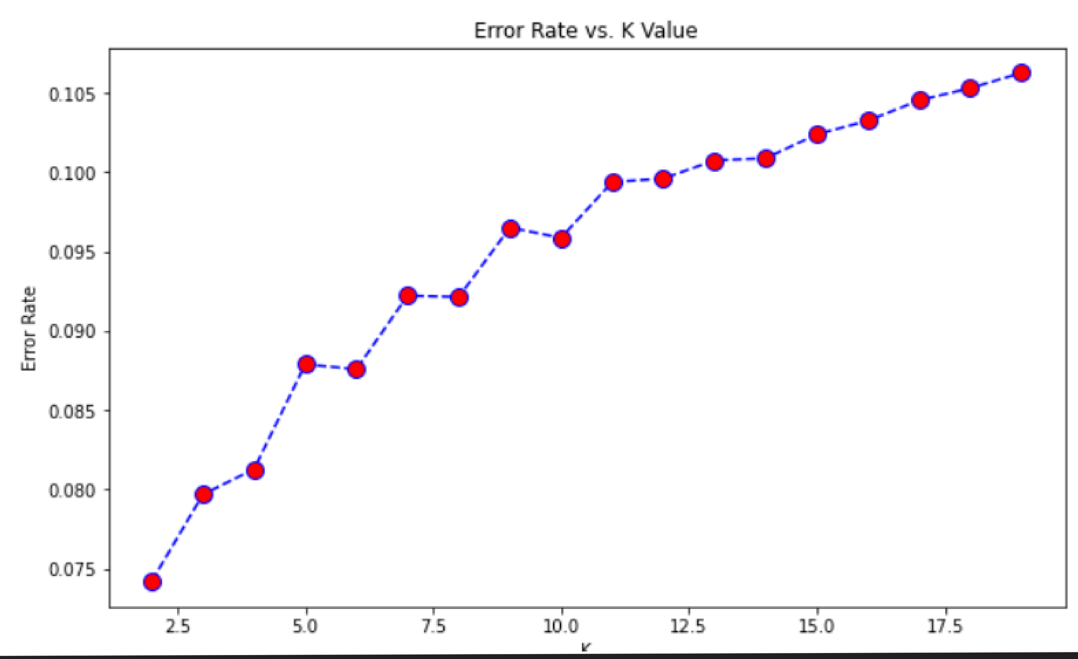
We have started with a simple yet efficient model the Logistic Regression model and the accuracy was (90%) not bad for the first shot !!
The model pipline included MinMax normalization, dimontionality reduction with PCA and the Logistic Regression model as shown below :

```
from sklearn.linear_model import LogisticRegression
steps = [('norm', MinMaxScaler()), ('pca', PCA(n_components = 'mle')), ('m', LogisticRegression(max_iter= 3000))]
lrmodel = Pipeline(steps=steps)
```

Secondly We used the Random Forest Algorithm .The fine-tuned model managed to achieve an accuracy of (91%) .We noticed an improvment so we decided to use another ensembling model the AdaBoost Classifier and the results were better as expected (93%). We tried the support vector classifier from the Support Vector Machine and the results were so disappointing with (85%) accuracy only

As for our last model We used the K-Nearest Neighbor algorithm .We let the algorithm run with different values for K (in range 1 => 30) and the improvement in the accuracy was insignificant compared to the increasment of the K value as shown in the graphic below





Conclusion

ML Algorithm	Result
Logistic Regression	0.90
Random Forest Classifier	0.91
Ada Boost Classifier	0.93
SVC (Support Vector Classifier)	0.85
K Neighbors Classifier	0.91

Since the problem is classification problem We have used the sklearn.accuracy as the Evaluation metric.
According to the result we decided to go with the AdaBoost Classifier(The best accuracy 93%).

DataSet Link : https://archive.ics.uci.edu/ml/datasets/bank+marketing