School of Computer Science and Engineering
May, 2021

Machine Learning

PROJECT REPORT

BANK MARKETING

Under the guidance of

BALAMURUGAN R

Submitted by

Gontla Bhargava Sai Sathvik-18BCI0087

CONTENTS

LIST OF FIGURES AND TABLES

ABSTRACT

Bank marketing refers to the various ways in which a bank can help a customer, such as operating accounts, making transfers, paying standing orders and selling foreign currency. Marketing is important for growing market share as well as sales in banking and insurance.

Marketing is essential for any business. Since the Banking sector is moving towards customer-centric, Marketing is very important for that. The marketing of bank services is the activity of presenting, advertising and selling of bank's products in the best possible way in

in order to satisfy consumers' requirements. In this project we will be using a bank marketing dataset from UCI machine learning repository. The data is related to direct marketing campaigns of a Portuguese banking institution. The marketing campaigns were based on phone calls This dataset is multivariate and has a total of 17 attributes. The primary objective of this project is to predict whether the client will subscribe to a term deposit or not. We will be using various supervised machine learning techniques such as logistic regression, decision tree classifier,support vector machine, etc and then we will compare accuracy of all the techniques among them. This will give us the best model with the highest accuracy. We will also find out the metricscores of the model such as precision, recall and f1 score. This will give us the best model with

the highest accuracy.

INTRODUCTION

In the last few years, machine learning (ML) has grown into one of the most significant IT and Artificial intelligence (AI) branches. This is a specific sub-group of AI based on the idea that the machine can learn by identifying patterns and make predictions in various data problems with minimum human intervention. Machine learning is a data analysis method that is widely used in various business and industrial sectors.The main reason for that because ML can build predictive models to produce better predictions and achieve the desired level of accuracy, leading to better outcomes.

The aim of the project is to find how to use machine learning techniques for analysis and making predictions using existing dataset in banking marketing. To find how they can be used together in a process of converting raw data to effective decision making knowledge. Building the predictive models will help to predict whether the client will subscribe for a term deposit.

This report will describe the different stages of preparation and implementation of the predictive models, staring with a literature review on machine learning techniques; in particular, linear regression and decision trees and how machine learning techniques are used in banking marking.

Once the literature review of these techniques has been revised, a methodology will be composed on how to pursue the investigation. The methodology is needed to establish how the implementation of the models will continue.

After the establishment of the methodology, the methods of data cleaning and preparationmethods on the raw data will be described and explained. The project will identify probabilities and visualize the results in order to improve the solutions and achieve desired outcomes. At the same time, a good understanding of banking marketing dataset will be provided so that the scopeof the analysis can be clearly defined.

LITERATURE SURVEY

S. No.	Title of the Paper And year	Algorithms Used	Performance Measures	Dataset being used	Gaps identified
1	Predicting Customer Response to Bank Direct Telemarketing Campaign (2017)	Multilayer Perceptron Neural Network (MLPNN), Decision Tree (C4.5), Logistic Regression and Random Forest (RF).	Evaluation of the classifiers was performed using classification accuracy and ROC. The RF classifier produced 86.80% as well as 92.7% respectively to place first among the classifiers.	UCI Machine Learning Repository database	Better Models Can be used.
2	Evaluation of Classification and Ensemble Algorithms for Bank Customer Marketing Response Prediction (2016)	Logistic Regression, Decision Tree, Naïve Bayes and the Random Forest ensemble after Cross Industry Standard for Data Mining (CRISP-DM)	A mix of ROC AUC and Classification Error rates. Random Forrest performed the best with AUC of 74.2% in balanced and unbalanced dataset.	Unnamed Portuguese bank dataset with 17 features.	Better Ensemble Models Can be used.
3	Evaluation of Machine Learning Frameworks on Bank Marketing and Higgs Datasets (2015)	comparison of Logical Regression and Linear SVM on Weka, Scikit-Learn and Spark frameworks	Standard Accuracy Measurements. Results show best results are obtained by Logistic regression on Scikit-Learn framework.	UCI Machine Learning Repository database	Many advanced classification models could have been tested.
4	Mining a Marketing Campaigns Data of Bank (March 2019)	Naive-Bayes Algorithm and One-R Algorithm.	Confusion Matrix and accuracy evaluation. Results show one-r to perform best with an accuracy of 89.3875%	UCI Machine Learning Repository database	Many advanced classification models(ensemble) could have been Used.

5	**Predicting the Success of Bank Telemarketing using various Classification Algorithms (2017)	Best subset algorithms of LASSO, logistic regression and random forrest followed by SVM, DT, RF and ANN classifiers	AUC and Accuracy used as performance measures. Results show best performance by RF algorithm of 90.63% accuracy on full model and 90.56% accuracy on the subset selected by random forest.	UCI Machine Learning Repository database	Advanced classification models(enseble models) could have been used.
6	Imbalanced customer classification for bank direct marketing (2017)	resampling algorithms of SMOTE, Tomek links, cluster under sampling and Classifiers namely Linear discriminant, logistic regression, k-nearest neighbours, C4.5, and MLPNN on raw and resampled data.	Evaluation is done using confusion matrix. Logistic Regression and secondarily Linear Discriminant on SMOTE oversampled data proved the most effective practices in case of limited resources.	UCI Machine Learning Repository database	Advanced Ensemble models can be used.
7	Direct marketing campaigns in retail banking with the use of deep learning and random forests (2019)	Deep belief network, RF, CART algorithms after Feature selection with Boruta algorithm	Precision Recall Metrics used for evaluation. After evaluation CART classifier applied after boruta provides the best results suited to the given business model	Web data from one of the leading ecommerce companies in Korea	Better Ensemble Models Can be used.
8	Customer Segmentation in Private Banking Sector Using Machine Learning Techniques (2012)	SVM and ANN models	Model evaluation was done using confusion matrix, SVM model using RBF kernel function clearly outruns the “affluent” detection of the MLP using gradient descent algorithm.	The database consists of 2,783 observations representing active cardholders at an important commercial bank from Romania.	Enseble classifiers could have been used to increase accuracy and efficiency

9	Visualization and Analysis in Bank Direct Marketing Prediction (2019) *	Oversampling by SMOTE, ADASYN, ROS, ADOMS, SPIDER and AHC followed by RF, SVM, K-neatest neighbour, Naïve Bayes.	It is found that RF classifier gives the best accuracy After SMOTE (89.98%) and on RAW data(90.02%)	UCI Machine Learning Repository database	Better classification models could have been used.
10	Bank Direct Marketing Based on Neural Network (2013)	Multilayer perceptron neural network (MLPNN) and Ross Quinlan new decision tree model (C5.0).	Evaluation is done using accuracy, sensitivity, specificity. Results show MLPNN to be better algorithm with accuracy of 90.32%	UCI Machine Learning Repository database	Neural network simply decimates the interpretability of your features to the point where it becomes Meaningless for the sake of performance, instead better classifier could have been used.
11	** Research on Bank Marketing Behaviour Based on Machine Learning (2020)	Under sampling followed by C5.0 decision tree	89.3% accuracy calculated after analysing confusion matrix	UCI Machine Learning Repository database	smaller dataset. Better models could have been used.
12	** Knowledge creation in banking marketing using machine learning techniques (2019)	data preparation and key simple and multiple linear regression, and C5.0 and CART decision tree.	C5.0 algorithm proved to be the best algorithm after Entropy attribute selection method with accuracy of 91.44%	UCI Irvine machine learning repository	Better Models Can be used.
13	Bank Direct Marketing Analysis of Data Mining Techniques (2014)	MLPNN, TAN, LR and C5.0	Results show that MLPNN shows best results with 90.49% accuracy	UCI Machine Learning Repository database	Better classification Models Can be used.
14	A data-driven approach to predict the success of bank telemarketing. (2014)	LR, DT, SVM, NN	NN gave the best value of 0.794 AUC	UCI Irvine machine learning repository	Ensemble classification Models Can be used.

15	** Zhang, C., (2016) Machine Learning on Bank Marketing Data [Online]	Logistic Regression, Radom Forest, Gradient Boosting, Support Vector Classifier and Neural Network. (with and without oversampling)	best results were shown by gradient boosting. Sensivity of NN and gradient boosting model increased on oversampling but that’s but the correct metric for this business model	UCI Irvine machine learning repository	Better Pre-processing can be done.

Base Papers

S. No.	**Title of the Paper And year**	**Algorithms Used**	**Performance Measures**	**Dataset being used**	**Gaps identified**
5	**Predicting the Success of Bank Telemarketing using various Classification Algorithms (2017)	Best subset algorithms of LASSO, logistic regression and random forrest followed by SVM, DT, RF and ANN classifiers	AUC and Accuracy used as performance measures. Results show best performance by RF algorithm of 90.63% accuracy on full model and 90.56% accuracy on the subset selected by random forest.	UCI Machine Learning Repository database	Advanced classification models(enseble models) could have been used.
11	** Research on Bank Marketing Behaviour Based on Machine Learning (2020)	Under sampling followed by C5.0 decision tree	89.3% accuracy calculated after analysing confusion matrix	UCI Machine Learning Repository database	smaller dataset. Better models could have been used.

12	** Knowledge creation in banking marketing using machine learning techniques (2019)	data preparation and key simple and multiple linear regression, and C5.0 and CART decision tree.	C5.0 algorithm proved to be the best algorithm after Entropy attribute selection method.	UCI Irvine machine learning repository	Better Models Can be used.
15	**Zhang, C., (2016) Machine Learning on Bank Marketing Data [Online]	Logistic Regression, Radom Forest, Gradient Boosting, Support Vector Classifier and Neural Network. (with and without oversampling)	best results were shown by gradient boosting. Sensitivity of NN and gradient boosting model increased on oversampling but that's but the correct metric for this business model	UCI Irvine machine learning repository	Better Pre-processing can be done.

METHODOLOGY

The aim of this project is to predict if the client will subscribe to a term deposit (target variable y). We carried out our classification goal by first dividing the main dataset into 3 different sets of dataset. The purpose behind this step is that the main dataset contains around 3 groups of attributes which are clients attributes, bank attributes and social and economic context attributes.

Data Preprocessing

We have obtained the datasets from the UCI machine learning repository which is related with direct marketing campaigns of a Portuguese banking institution. Bank Marketing contains 41188 instances and 17 attributes. Its features are both numerical and categorical. Hence the data needs to be vectorized. For this dataset, feature extraction and feature selection has to be done properly.

Train Test Split

Sklearn's train_test_split is used to split the dataset into training and testing set. It takes the features and the target variables as parameters. Along with them, it has other parameters to define train size and test size, random state, shuffle and stratify. All these parameters are used to divide training and testing sets.

Machine Learning Algorithms

1. Logistic Regression: Logistic Regression is a machine learning model used to model a binary dependent variable (here, bank term deposit). We apply Logistic Regression and fit the model using the Training Data. First import the LogisticRegression from sklearn and create an object logmodel. Fit the model by using the training data and then evaluate the model using the testing data.

2. Support Vector Machine: Support Vector Machine (or SVM) is a classification model used to divide various points into different zones and then predict the output within a zone.We use Support Vector Machine to fit the model using the Training Data. First import the SVC from sklearn.svm and create an object named svc with sigmoid kernel.Fit the model by using the training data and then evaluate the model using the testing data.

3. Decision Tree Classifier: Decision Tree Classification is used to go from various variables and their ranged or discrete values to predict a
discrete output. We now use Decision Tree to classify. First import the DecisionTreeClassifier from sklearn.tree and create an object named dtree with criterion as entropy. Fit the model by using the training data and then evaluate the model using the testing data.

Ensemble Learning Models

In order to boost the accuracy of the models, we are going to apply these 2 following ensemble techniques:
1. Random Forest
2. XGBClassifier

1. Random Forest: Random forests are ensembles of decision tree classifications and hence generate multiple decision trees to have the most accurate results. We use the Random Forest Ensemble Method to classify. First import the RandomForestClassifier from sklearn.ensemble and create an object named rfc with criterion as entropy with 200 estimators. Fit the model by using the training data and then evaluate the model using the testing data.

2. XGB Classifier: XGBClassification or Gradient Boosting works as an ensemble of weak prediction models and then chooses the best to generate results. Next we use the

XGBClassifier Ensemble Method to classify. First import the XGBClassifier from xgboost and create an object named xgb. Fit the model by using the training data and thenevaluate the model using the testing data.

Evaluation

Comparison of the frameworks mentioned in above is based on the following parameters:

- CPU Utilization: It denotes the percentage of CPU that was used while the machine learning algorithms were being executed on the datasets. When run on multi-core processors, high CPU utilization infers processes being executed on all cores of the processor.
- Training Time: As supervised machine learning algorithms are considered, first the model is trained using a training dataset. Time taken to train the model is called training time. This varies on the implementations of the algorithms.
- Accuracy : It is the absolute accuracy of the predictions from the trained models against provided class labels.
- Root mean squared error (RMSE): It is a frequently used factor indicating the difference between the value predicted by a model or an estimator and the values actually observed.

From above metrics we can compare the accuracies provided by both the algorithms on a dataset

RESULT

Our results show that the accuracy of Logistic Regression, Support Vector Machine, and XGB classifier are all similar and the highest. Support vector machine performs the worst with least accuracy of 87% .

DISCUSSION

The Objective of the experiment is to compare the performance of machine learning algorithms when deployed on different frameworks in terms of the accuracy and RMSE. The dataset contains 41188 instances of bank clients data and 20 different features out of which 10 are nominal and the rest are numeric features. It is a binary classification problem to predict whether the client will subscribe to that bank's term deposit or not.

Bank_client

This is our main dataset deduced from the main dataset. Bank_client contains age, job, martial, education, default, housing and loan as it attributes. The min age of a customer is 17 years and max age is 98 years.

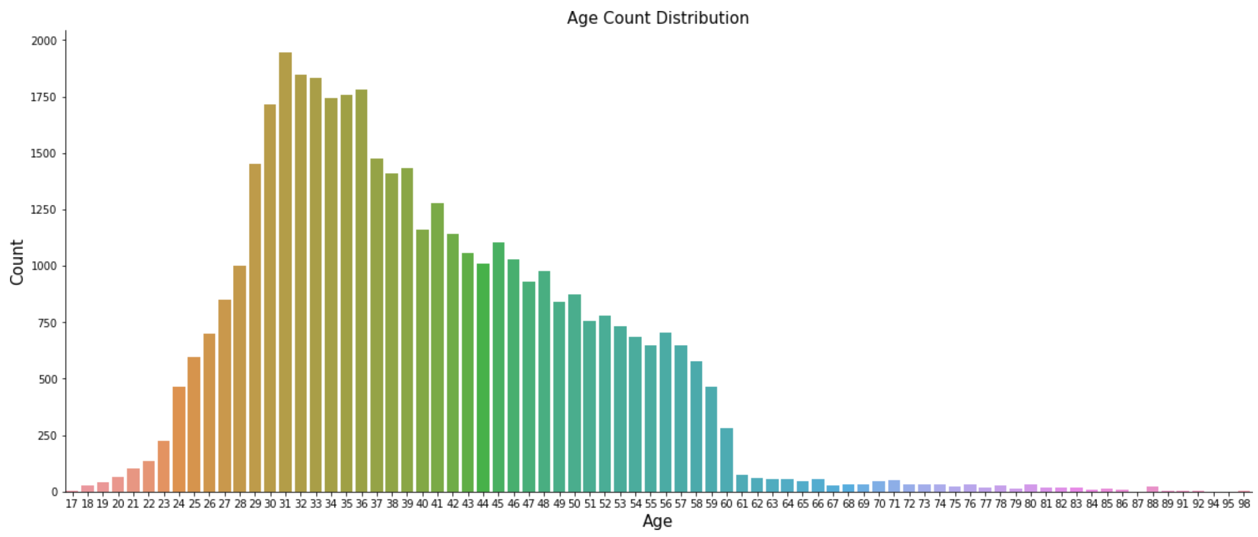


Fig 1.1 Age Count Distribution

This is the age count distribution barplot which signifies that the majority of customers lie under 31 years of age group.

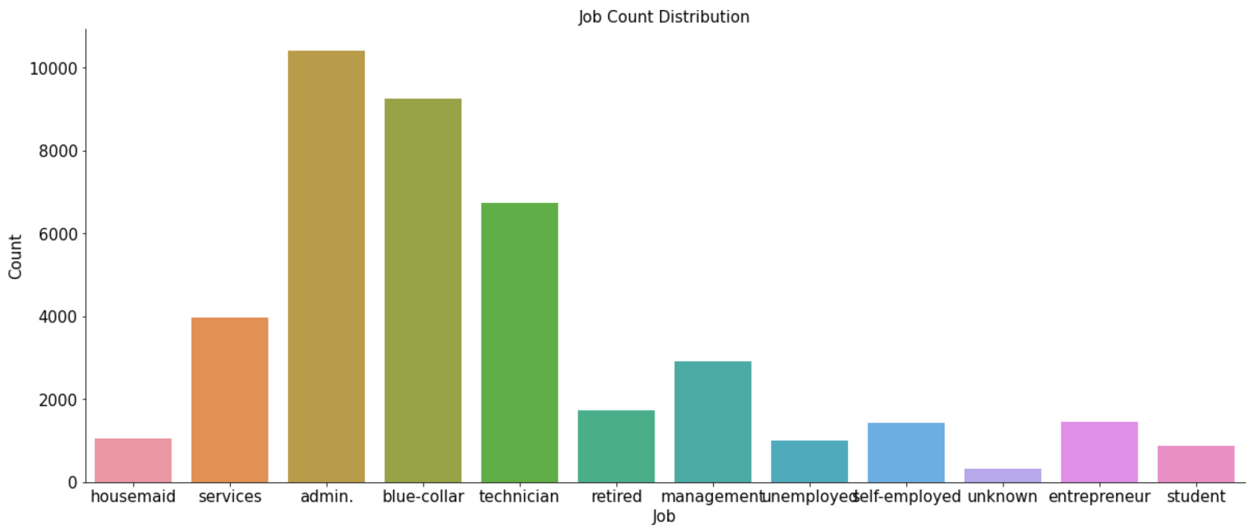


Fig 1.2 Job Count Distribution

This is the job count distribution barplot which signifies that the majority of customers lie under the administrative job group (administrative workers) and minority lies under unknown category .

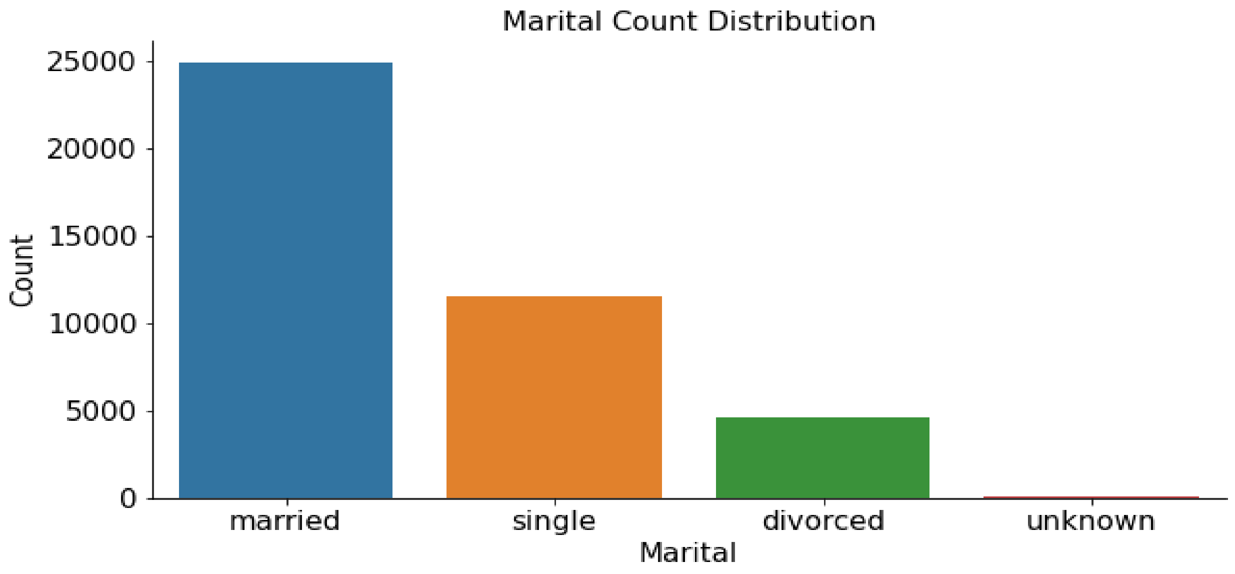


Fig 1.3 Marital Count Distribution

This is the marital count distribution barplot which signifies that the majority of customers are married.

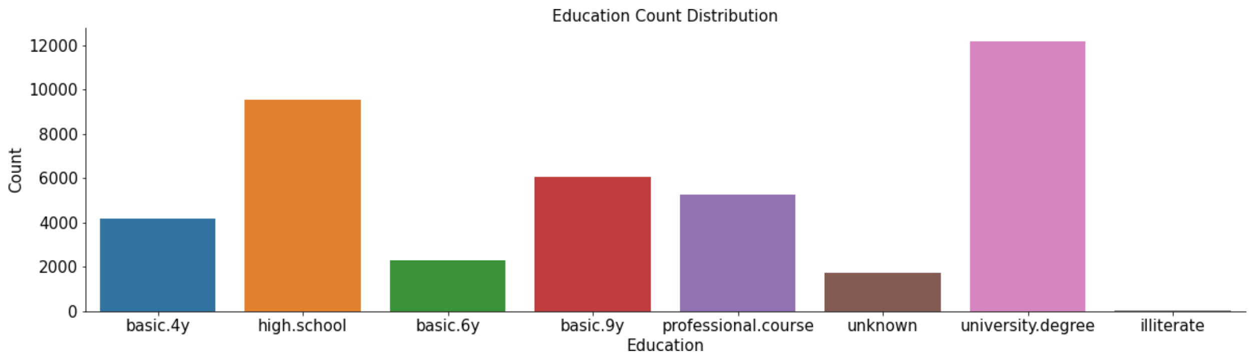


Fig 1.4 Education Count Distribution

This is the marital count distribution barplot which signifies that the majority of customers have university degrees.

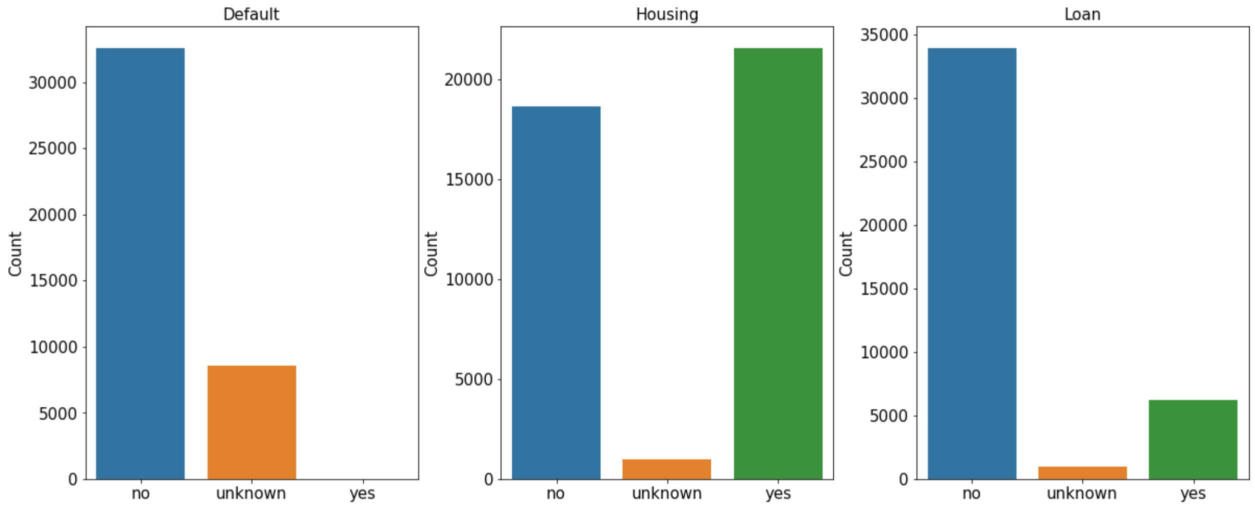


Fig 1.5 Default, Housing and Loan Count Distribution

From the default, housing loan and personal loan distribution barplot it has been deduced that the majority of the customers have no credit, no personal loans but they have house loans.

At last attributes of the bank_client dataset are label encoded so as to convert values of all attributes to numerical values.

Bank_related

Bank_related contains contact, month, day_of_week and duration as it attributes.

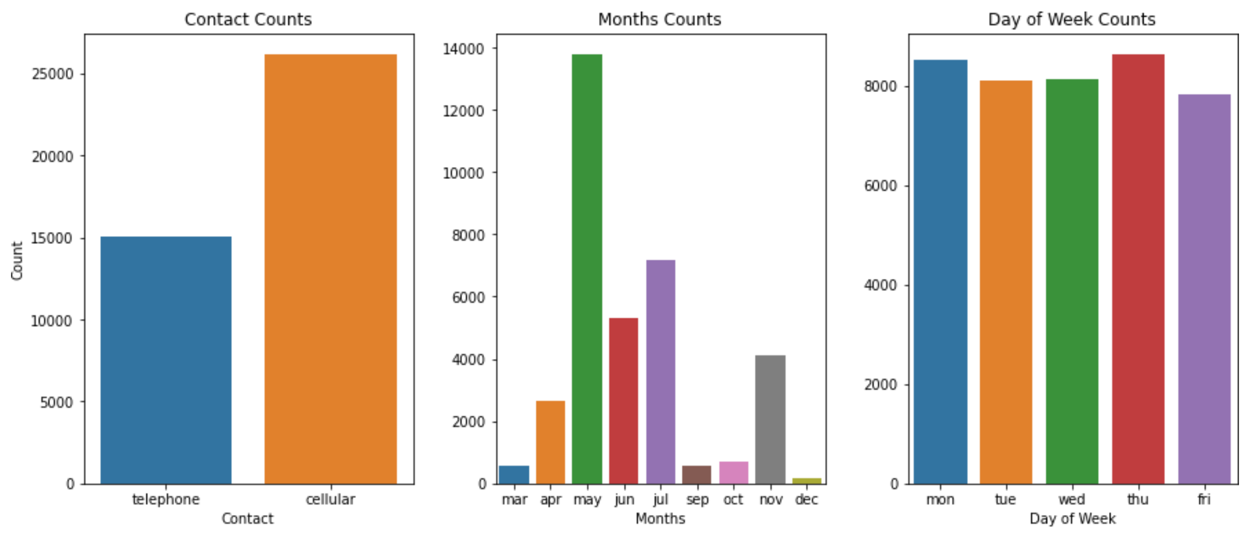


Fig 2.1 Contact, Months and Days of the week Count Distribution

From the contact, Months and day of week counts distribution barplot it has been deduced that the majority of the campaigns is done through cellular calls in the month of may and mostly on everyday.

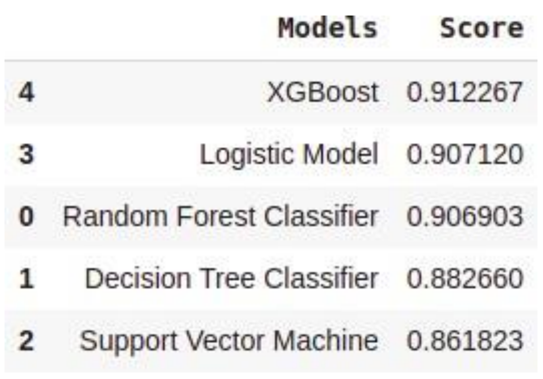
At last attributes of the bank_related dataset are label encoded so as to convert values of all attributes to numerical values.

Accuracy and RMSE comparison

S.No	Models	Accuracy	RMSE	Precision	F1 Score	Recall
1.	Logistic Regression	91.0%	0.29	65.0%	48.5%	38.7%
2.	Support Vector Machine	86.4%	0.36	38.1%	38.6%	39.1%
3.	Decision Tree Classifier	88.8%	0.33	48.9%	50.6%	52.4%
4.	Random Forest	91.1%	0.29	61.7%	55.7%	50.8%
5.	XGBoost Classifier	91.6%	0.28	65.5%	56.1%	49.1%

Table 1.1 Accuracy and RMSE Comparison

Cross Value Score



	Models	Score
4	XGBoost	0.912267
3	Logistic Model	0.907120
0	Random Forest Classifier	0.906903
1	Decision Tree Classifier	0.882660
2	Support Vector Machine	0.861823

ROC CURVE

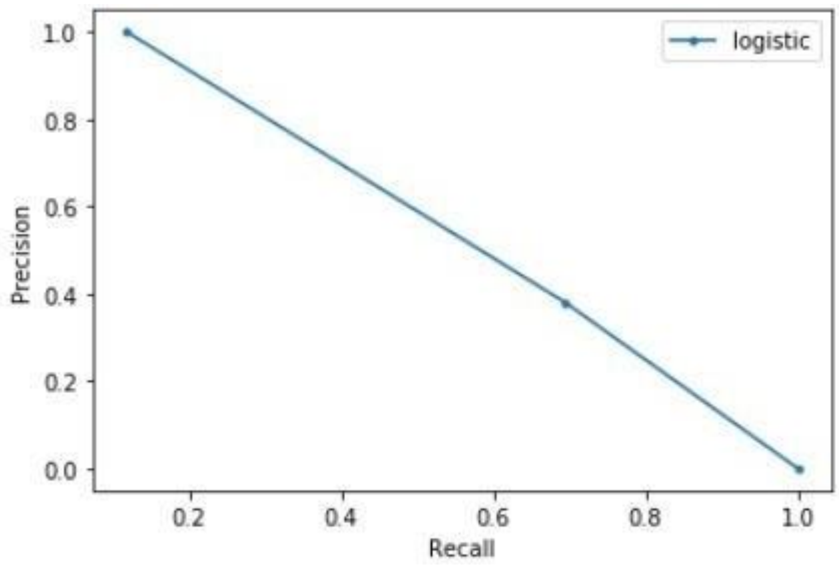


Fig 3.1 Logistic Regression ROC

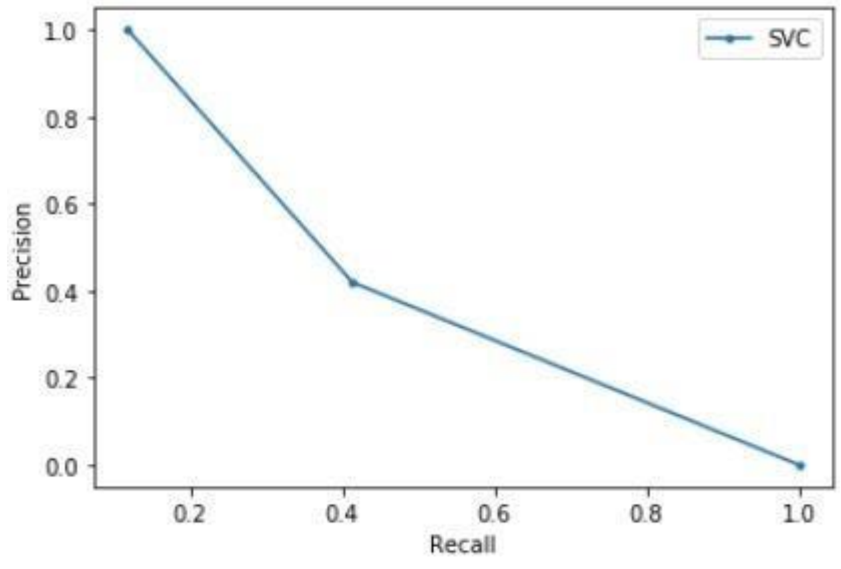


Fig 3.2 Support Vector ROC

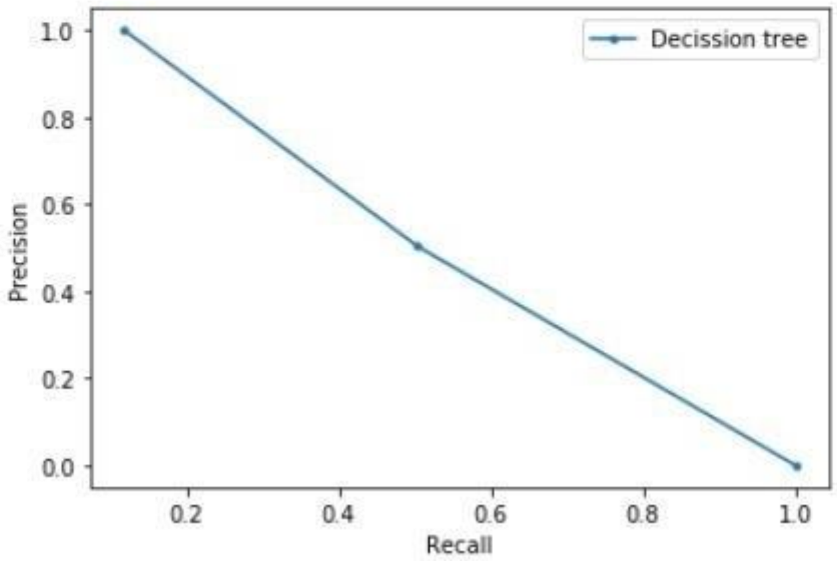


Fig 3.3 Decision Tree ROC

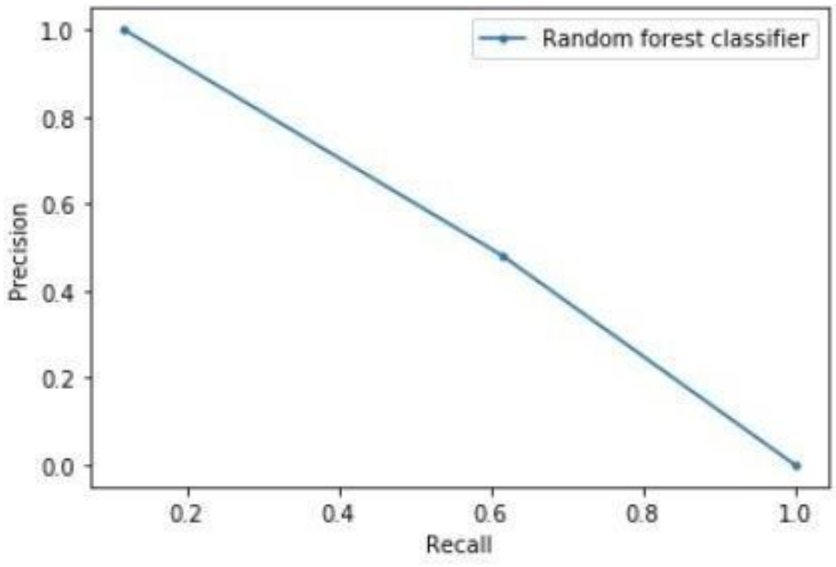


Fig 3.4 Random Forest ROC

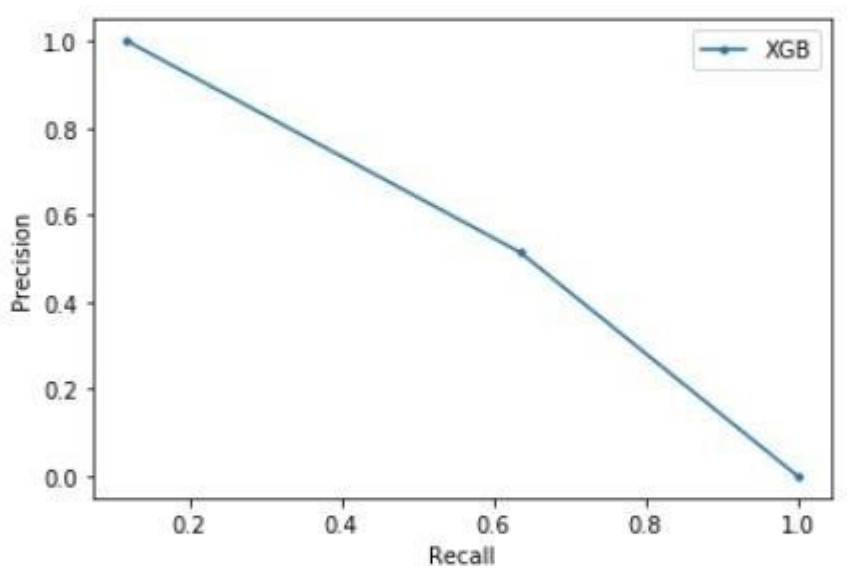


Fig 3.5 XGBoost Classifier ROC

CONCLUSION AND FUTURE WORK

In this project, we were able to find the best machine learning model for the marketing campaign of the Portugese Bank Institution, which can predict whether a customer will subscribe to their term deposit or not. We used accuracy of the model as the deciding factor in choosing the best method.

We use Random Forest Ensemble Method to classify. First import the Random Forest Classifier from sklearn. Ensemble and create an object named RFC with criterion as entropy with 200 estimators. Fit the model by using the training data and then evaluate the model using the testing data.

Accuracy: 0.904375

Precision: 0.615056818182

Recall: 0.467098166127

f1 Score: 0.530962599632

We use XGB Classifier Ensemble Method to classify. First import the XGB Classifier from XGBoost and create an object named XGB. Fit the model by using the training data and then evaluate the model using the testing data.

Accuracy: 0.907625

Precision: 0.626684636119

Recall: 0.501618122977

f1 Score: 0.557219892151

Result Analysis for Ensemble Methods

Random Forest - 90.437%

XGB Classifier - 90.7625%

As can be clearly seen from the accuracy results, the best regular model of machine learning in this case is XGB Classifier.

The other Metrics Scores for Logisitc Regression is as follows:

Precision - 0.626

Recall - 0.501

f1 score - 0.557

In banking system traditionally value was created through growth and efficiency but today creating customer value by manifesting opportunities out of the disruptive environment based on technology and external partnerships will be the success denominator and operational efficiency. The banking sector is expected to witness some unprecedented changes in the times to come.
Technology geared toward improving retail banks' operational efficiency is positively impacting the market. According to Insider Intelligence, 39% of retail banking executives say that reducing costs is where technology has the greatest impact, compared to only 24% who say it's improving customer experience.
In future, with the advancement in this field, we can focus on changing the models or adding more efficient or new models for prediction. Moreover, we can focus on altering or changing the project with changes in the bank marketing campaign. the Banking Industry is disrupted with emerging technologies and business models that have blurred the lines between business and technology.

REFERENCES

1. Grzonka, Daniel, Grażyna Suchacka, and Barbara Borowik. "Application of selected supervised classification methods to bank marketing campaign." *Information Systems in Management* 5.1 (2016): 36-48.
2. Macmillan Education Ltd. *Bank marketing management*. Macmillan International Higher Education, 2015.
3. Moro, Sergio, Raul Laureano, and Paulo Cortez. "Using data mining for bank direct marketing: An application of the crisp-dm methodology." (2011).
4. Doerr, Sebastian, Leonardo Gambacorta, and José María Serena Garralda. "Big data and machine learning in central banking." *BIS Working Papers* 930 (2021).
5. Dalmia, Hemlata, Ch VSS Nikil, and Sandeep Kumar. "Churning of Bank Customers Using Supervised Learning." *Innovations in Electronics and CommunicationEngineering*. Springer, Singapore, 2020. 681-691.

APPENDIX - A - DATASET

Abstract: The data is related with direct marketing campaigns (phone calls) of a Portuguese banking institution.

The classification goal is to predict if the client will subscribe a term deposit (variable y).

Data Set Information: The data is 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 access if the product (bank term deposit) would be ('yes') or not ('no') subscribed.

Attribute Information:

Bank client data:

Age (numeric)

Job : type of job (categorical: 'admin.', 'blue-collar', 'entrepreneur', 'housemaid', 'management', 'retired', 'self-employed', 'services', 'student', 'technician', 'unemployed', 'unknown')

Marital : marital status (categorical: 'divorced', 'married', 'single', 'unknown' ; note: 'divorced' means divorced or widowed)

Education (categorical: 'basic.4y', 'basic.6y', 'basic.9y', 'high.school', 'illiterate', 'professional.course', 'university.degree', 'unknown')

Default: has credit in default? (categorical: 'no', 'yes', 'unknown')

Housing: has housing loan? (categorical: 'no', 'yes', 'unknown')

Loan: has personal loan? (categorical: 'no', 'yes', 'unknown')

Related with the last contact of the current campaign:

Contact: contact communication type (categorical: 'cellular','telephone')

Month: last contact month of year (categorical: 'jan', 'feb', 'mar', ..., 'nov', 'dec')

Day_of_week: last contact day of the week (categorical: 'mon','tue','wed','thu','fri')

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:

Campaign: number of contacts performed during this campaign and for this client (numeric,includes last contact)

Pdays: number of days that passed by after the client was last contacted from a previouscampaign (numeric; 999 means client was not previously contacted)

Previous: number of contacts performed before this campaign and for this client (numeric) Poutcome: outcome of the previous marketing campaign (categorical: 'failure', 'nonexistent','success')

Social and economic context attributes

Emp.var.rate: employment variation rate - quarterly indicator (numeric) Cons.price.idx: consumer price index - monthly indicator (numeric) Cons.conf.idx: consumer confidence index - monthly indicator (numeric)Euribor3m: euribor 3 month rate - daily indicator (numeric) Nr.employed: number of employees - quarterly indicator (numeric)

Output variable (desired target):

y - has the client subscribed a term deposit? (binary: 'yes', 'no')

Data-set Source: http://archive.ics.uci.edu/ml/datasets/Bank+Marketing (http://archive.ics.uci.edu/ml/datasets/Bank+Marketing)

Bank Marketing

Importing libraries

In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
import warnings
warnings.filterwarnings('ignore')
```

Imporing Dataset and displaying the firt 10 rows

In [2]:

```
bank  = pd.read_csv('C:/Users/user/Downloads/bank-additional-full.csv', sep  = ';')
y = pd.get_dummies(bank['y'], columns = ['y'], prefix = ['y'], drop_first = True)
bank.head(10)
```

Out[2]:

	age	job	marital	education	default	housing	loan	contact	month	day_of_week	...	campaign	pdays	previou
0	56	housemaid	married	basic.4y	no	no	no	telephone	may	mon	...	1	999	
1	57	services	married	high.school	unknown	no	no	telephone	may	mon	...	1	999	
2	37	services	married	high.school	no	yes	no	telephone	may	mon	...	1	999	
3	40	admin.	married	basic.6y	no	no	no	telephone	may	mon	...	1	999	
4	56	services	married	high.school	no	no	yes	telephone	may	mon	...	1	999	
5	45	services	married	basic.9y	unknown	no	no	telephone	may	mon	...	1	999	
6	59	admin.	married	professional.course	no	no	no	telephone	may	mon	...	1	999	
7	41	blue-collar	married	unknown	unknown	no	no	telephone	may	mon	...	1	999	
8	24	technician	single	professional.course	no	yes	no	telephone	may	mon	...	1	999	
9	25	services	single	high.school	no	yes	no	telephone	may	mon	...	1	999	

10 rows × 21 columns

In [3]:

```
bank.info()
```

```
<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
```

In [4]:

```
bank.columns
```

Out[4]:

```
Index(['age', 'job', 'marital', 'education', 'default', 'housing', 'loan',
       'contact', 'month', 'day_of_week', 'duration', 'campaign', 'pdays',
       'previous', 'poutcome', 'emp.var.rate', 'cons.price.idx',
       'cons.conf.idx', 'euribor3m', 'nr.employed', 'y'],
      dtype='object')
```

Detailed analysis of the Dataset

1. Bank client data Analysis and processing

Extract the first 7 colums of data (Bank Client Details) from the dataset and display the first 10 rows.

In [5]:

```
bank_client = bank.iloc[: , 0:7]
bank_client.head(10)
```

Out[5]:

	age	job	marital	education	default	housing	loan
0	56	housemaid	married	basic.4y	no	no	no
1	57	services	married	high.school	unknown	no	no
2	37	services	married	high.school	no	yes	no
3	40	admin.	married	basic.6y	no	no	no
4	56	services	married	high.school	no	no	yes
5	45	services	married	basic.9y	unknown	no	no
6	59	admin.	married	professional.course	no	no	no
7	41	blue-collar	married	unknown	unknown	no	no
8	24	technician	single	professional.course	no	yes	no
9	25	services	single	high.school	no	yes	no

1.1. Knowing the categorical variables

Printing all the various categories/labels of the attributes

In [6]:

```
print('Jobs:\n', bank_client['job'].unique())
print('Marital:\n', bank_client['marital'].unique())
print('Education:\n', bank_client['education'].unique())
print('Default:\n', bank_client['default'].unique())
print('Housing:\n', bank_client['housing'].unique())
print('Loan:\n', bank_client['loan'].unique())
```

```
Jobs:
 ['housemaid' 'services' 'admin.' 'blue-collar' 'technician' 'retired'
 'management' 'unemployed' 'self-employed' 'unknown' 'entrepreneur'
 'student']
Marital:
 ['married' 'single' 'divorced' 'unknown']
Education:
 ['basic.4y' 'high.school' 'basic.6y' 'basic.9y' 'professional.course'
 'unknown' 'university.degree' 'illiterate']
Default:
 ['no' 'unknown' 'yes']
Housing:
 ['no' 'yes' 'unknown']
Loan:
 ['no' 'yes' 'unknown']
```

1.2. Analysing and plotting Age Count

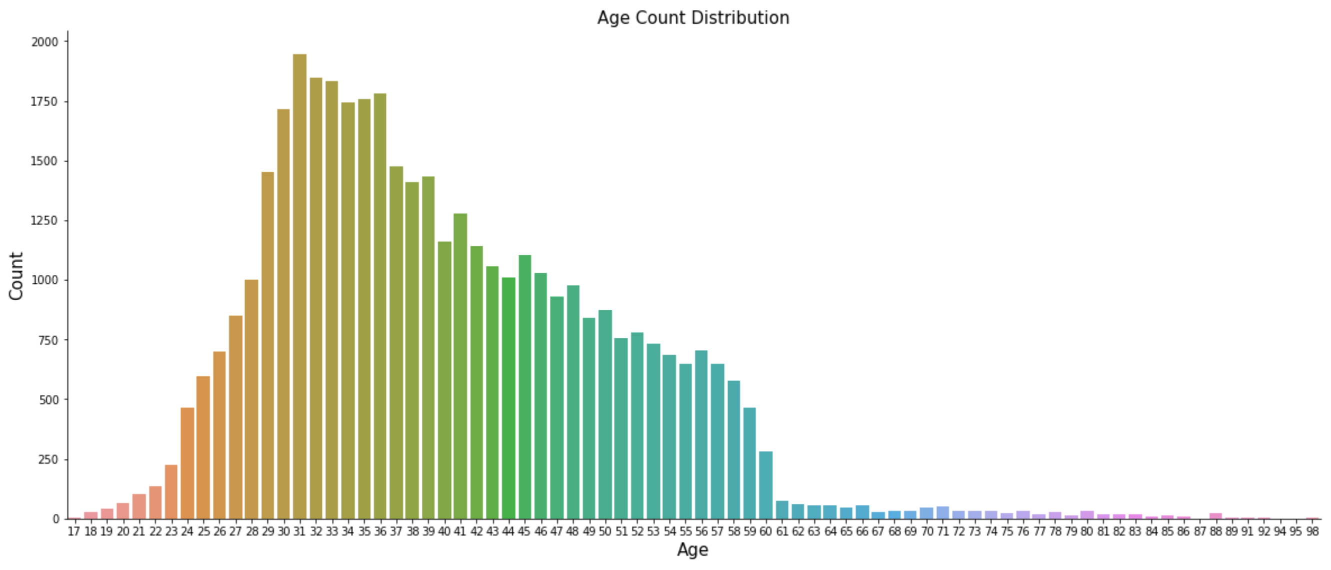
In [7]:

```
print('Min age: ', bank_client['age'].max())
print('Max age: ', bank_client['age'].min())
print('Null Values: ', bank_client['age'].isnull().any())
```

```
Min age:  98
Max age:  17
Null Values:  False
```

In [8]:

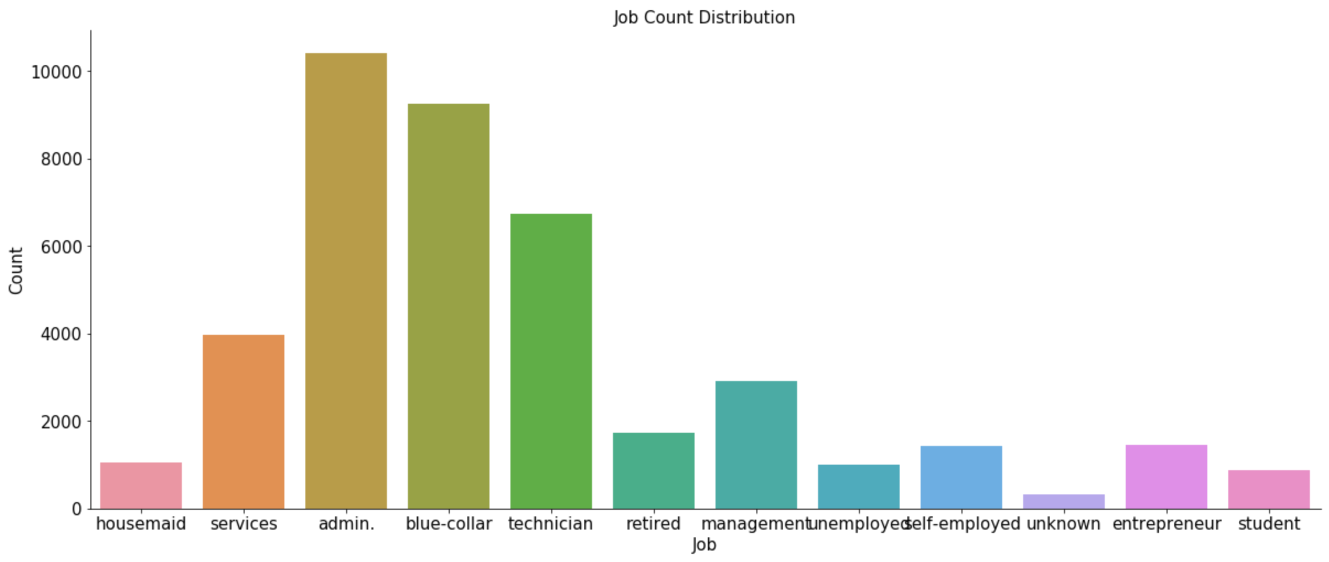
```
fig, ax = plt.subplots()
fig.set_size_inches(20, 8)
sns.countplot(x = 'age', data = bank_client)
ax.set_xlabel('Age', fontsize=15)
ax.set_ylabel('Count', fontsize=15)
ax.set_title('Age Count Distribution', fontsize=15)
sns.despine()
```



1.3 Plotting Jobs

In [9]:

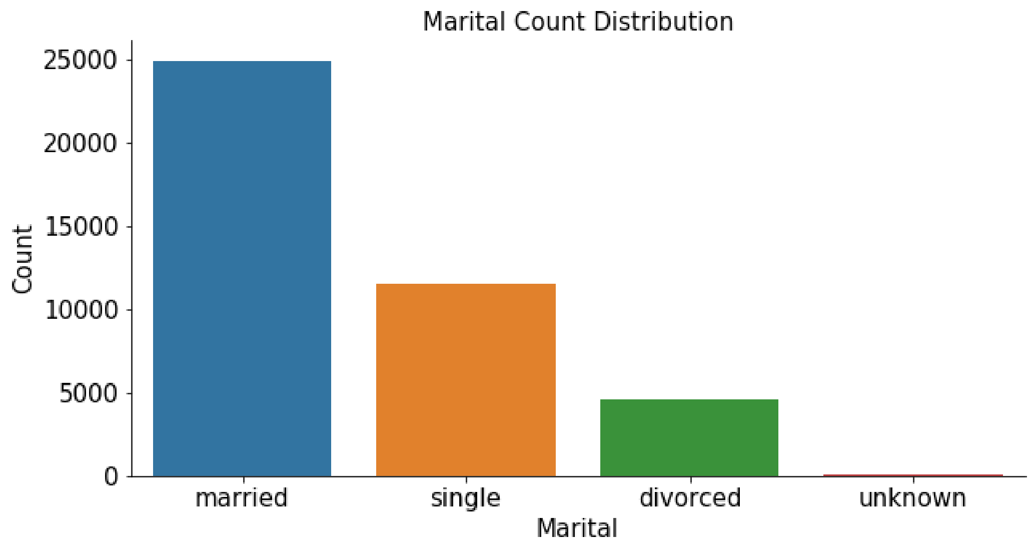
```
fig, ax = plt.subplots()
fig.set_size_inches(20, 8)
sns.countplot(x = 'job', data = bank_client)
ax.set_xlabel('Job', fontsize=15)
ax.set_ylabel('Count', fontsize=15)
ax.set_title('Job Count Distribution', fontsize=15)
ax.tick_params(labelsize=15)
sns.despine()
```



1.4 Plotting Marital

In [10]:

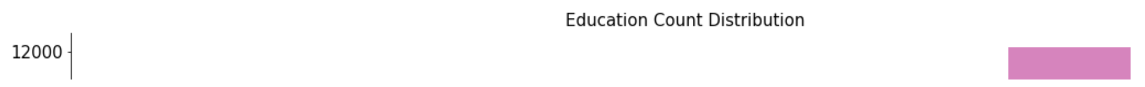
```
fig, ax = plt.subplots()
fig.set_size_inches(10, 5)
sns.countplot(x = 'marital', data = bank_client)
ax.set_xlabel('Marital', fontsize=15)
ax.set_ylabel('Count', fontsize=15)
ax.set_title('Marital Count Distribution', fontsize=15)
ax.tick_params(labelsize=15)
sns.despine()
```

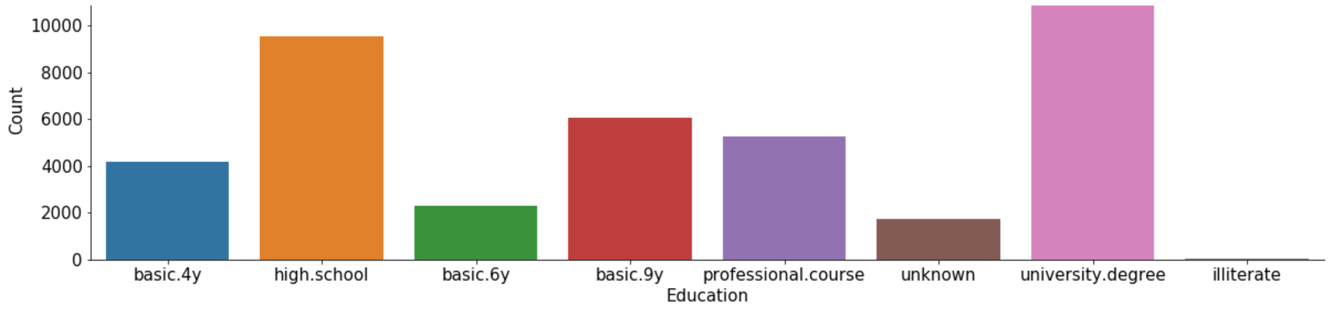


1.5 Plotting Education

In [11]:

```
fig, ax = plt.subplots()
fig.set_size_inches(20, 5)
sns.countplot(x = 'education', data = bank_client)
ax.set_xlabel('Education', fontsize=15)
ax.set_ylabel('Count', fontsize=15)
ax.set_title('Education Count Distribution', fontsize=15)
ax.tick_params(labelsize=15)
sns.despine()
```

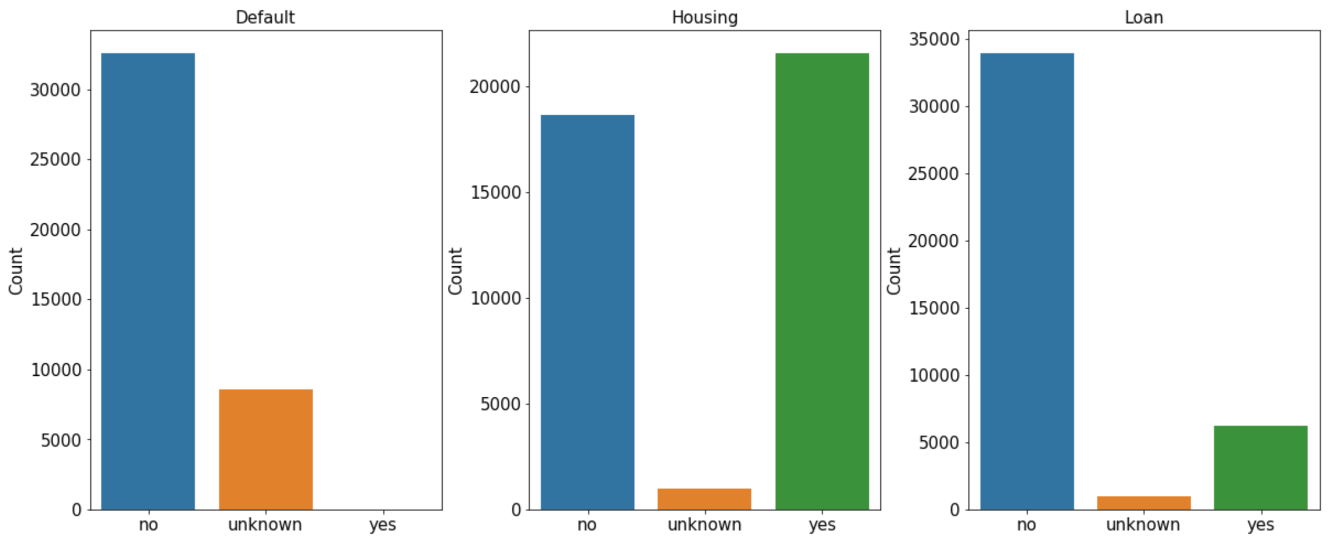




1.6 Plotting Default, housing, loan

In [12]:

```
# Default
fig, (ax1, ax2, ax3) = plt.subplots(nrows = 1, ncols = 3, figsize = (20,8))
sns.countplot(x = 'default', data = bank_client, ax = ax1, order = ['no', 'unknown', 'yes'])
ax1.set_title('Default', fontsize=15)
ax1.set_xlabel('')
ax1.set_ylabel('Count', fontsize=15)
ax1.tick_params(labelsize=15)
# Housing
sns.countplot(x = 'housing', data = bank_client, ax = ax2, order = ['no', 'unknown', 'yes'])
ax2.set_title('Housing', fontsize=15)
ax2.set_xlabel('')
ax2.set_ylabel('Count', fontsize=15)
ax2.tick_params(labelsize=15)
# Loan
sns.countplot(x = 'loan', data = bank_client, ax = ax3, order = ['no', 'unknown', 'yes'])
ax3.set_title('Loan', fontsize=15)
ax3.set_xlabel('')
ax3.set_ylabel('Count', fontsize=15)
ax3.tick_params(labelsize=15)
plt.subplots_adjust(wspace=0.25)
```



Converting Bank Clients attribute values to into various numeric groups.

In [13]:

```
from sklearn.preprocessing import LabelEncoder
labelencoder_X = LabelEncoder()
bank_client['job'] = labelencoder_X.fit_transform(bank_client['job'])
bank_client['marital'] = labelencoder_X.fit_transform(bank_client['marital'])
bank_client['education']= labelencoder_X.fit_transform(bank_client['education'])
bank_client['default'] = labelencoder_X.fit_transform(bank_client['default'])
bank_client['housing'] = labelencoder_X.fit_transform(bank_client['housing'])
bank_client['loan'] = labelencoder_X.fit_transform(bank_client['loan'])
```

Since there are many age values, we group them into fewer values to improve the training efficiency.

In [14]:

```
def age(dataframe):
    dataframe.loc[dataframe['age'] <= 32, 'age'] = 1
    dataframe.loc[(dataframe['age'] > 32) & (dataframe['age'] <= 47), 'age'] = 2
    dataframe.loc[(dataframe['age'] > 47) & (dataframe['age'] <= 70), 'age'] = 3
    dataframe.loc[(dataframe['age'] > 70) & (dataframe['age'] <= 98), 'age'] = 4
    return dataframe
age(bank_client);
```

In [15]:

```
bank_client.head()
```

Out[15]:

	age	job	marital	education	default	housing	loan
0	3	3	1	0	0	0	0
1	3	7	1	3	1	0	0
2	2	7	1	3	0	2	0
3	2	0	1	1	0	0	0
4	3	7	1	3	0	0	2

In [16]:

```
print(bank_client.shape)
bank_client.head()
```

```
(41188, 7)
```

Out[16]:

	age	job	marital	education	default	housing	loan
0	3	3	1	0	0	0	0
1	3	7	1	3	1	0	0
2	2	7	1	3	0	2	0
3	2	0	1	1	0	0	0
4	3	7	1	3	0	0	2

Displaying data related to the last contact of the current campaign

In [17]:

```
bank_related = bank.iloc[: , 7:11]
bank_related.head()
```

Out[17]:

	contact	month	day_of_week	duration
0	telephone	may	mon	261
1	telephone	may	mon	149
2	telephone	may	mon	226
3	telephone	may	mon	151
4	telephone	may	mon	307

In [18]:

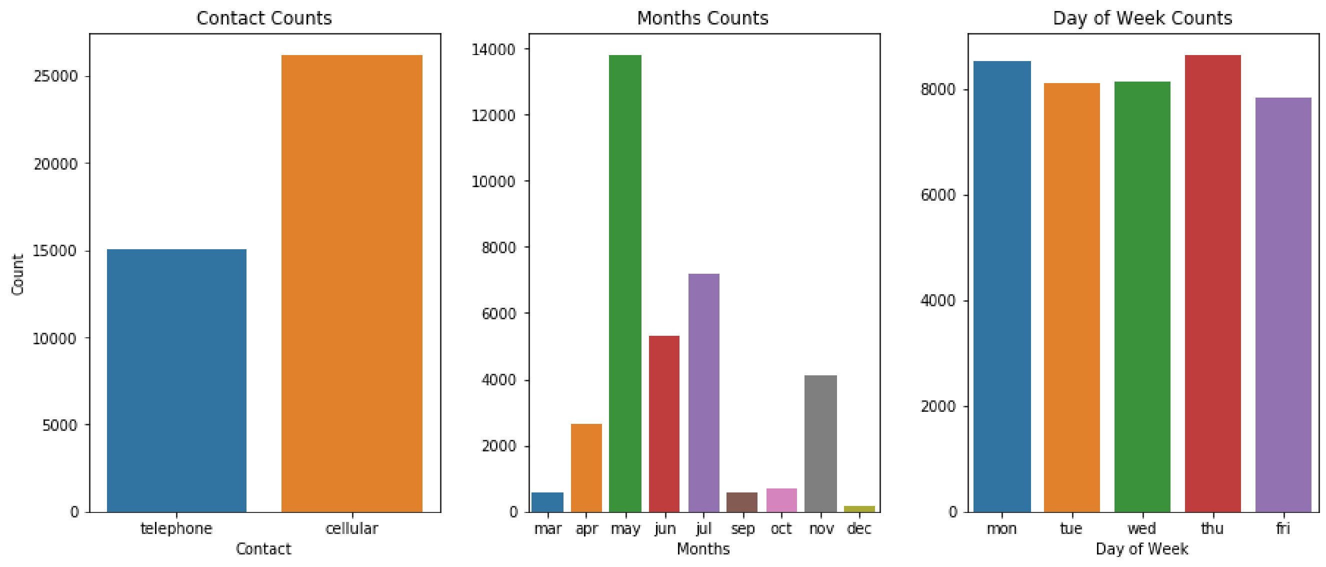
```
print("Kind of Contact: \n", bank_related['contact'].unique())
print("Which monthis this campaign work: \n", bank_related['month'].unique())
print("Which days of week this campaign work: \n", bank_related['day_of_week'].unique())
```

```
Kind of Contact: 
 ['telephone' 'cellular']
Which monthis this campaign work: 
 ['may' 'jun' 'jul' 'aug' 'oct' 'nov' 'dec' 'mar' 'apr' 'sep']
Which days of week this campaign work: 
 ['mon' 'tue' 'wed' 'thu' 'fri']
```

Plotting Contact, Month, Day of Week

In [19]:

```
fig, (ax1, ax2, ax3) = plt.subplots(nrows = 1, ncols = 3, figsize = (15,6))
sns.countplot(bank_related['contact'], ax = ax1)
ax1.set_xlabel('Contact', fontsize = 10)
ax1.set_ylabel('Count', fontsize = 10)
ax1.set_title('Contact Counts')
ax1.tick_params(labelsize=10)
sns.countplot(bank_related['month'], ax = ax2, order = ['mar', 'apr', 'may', 'jun', 'jul','sep','oc
t','nov','dec'])
ax2.set_xlabel('Months', fontsize = 10)
ax2.set_ylabel('')
ax2.set_title('Months Counts')
ax2.tick_params(labelsize=10)
sns.countplot(bank_related['day_of_week'], ax = ax3)
ax3.set_xlabel('Day of Week', fontsize = 10)
ax3.set_ylabel('')
ax3.set_title('Day of Week Counts')
ax3.tick_params(labelsize=10)
plt.subplots_adjust(wspace=0.25)
```



Converting the labels of Contact, Month, Day of Week to numeric values.

In [20]:

```
from sklearn.preprocessing import LabelEncoder
labelencoder_X = LabelEncoder()
bank_related['contact']  = labelencoder_X.fit_transform(bank_related['contact'])
bank_related['month'] = labelencoder_X.fit_transform(bank_related['month'])
bank_related['day_of_week']  =  labelencoder_X.fit_transform(bank_related['day_of_week'])
```

In [21]:

```
bank_related.head()
```

Out[21]:

	contact	month	day_of_week	duration
0	1	6	1	261
1	1	6	1	149
2	1	6	1	226
3	1	6	1	151
4	1	6	1	307

Converting duration attribute values to into various numeric groups.

In [22]:

```
def duration(data):
    data.loc[data['duration'] <= 102, 'duration'] = 1
    data.loc[(data['duration'] > 102) & (data['duration'] <= 180) , 'duration'] = 2
    data.loc[(data['duration'] > 180) & (data['duration'] <= 319) , 'duration'] = 3
    data.loc[(data['duration'] > 319) & (data['duration'] <= 644.5), 'duration'] = 4
    data.loc[data['duration'] > 644.5, 'duration'] = 5
    return data
duration(bank_related);
```

In [23]:

```
bank_related.head()
```

Out[23]:

	contact	month	day_of_week	duration
0	1	6	1	3
1	1	6	1	2
2	1	6	1	3
3	1	6	1	2
4	1	6	1	3

Social and economic context attributes

In [24]:

```
bank_se = bank.loc[: , ['emp.var.rate', 'cons.price.idx', 'cons.conf.idx', 'euribor3m', 'nr.employe
d']]
bank_se.head()
```

Out[24]:

	emp.var.rate	cons.price.idx	cons.conf.idx	euribor3m	nr.employed
0	1.1	93.994	-36.4	4.857	5191.0
1	1.1	93.994	-36.4	4.857	5191.0
2	1.1	93.994	-36.4	4.857	5191.0
3	1.1	93.994	-36.4	4.857	5191.0
4	1.1	93.994	-36.4	4.857	5191.0

Other attributes

In [25]:

```
bank_o = bank.loc[: , ['campaign', 'pdays','previous', 'poutcome']]
bank_o.head()
```

Out[25]:

	campaign	pdays	previous	poutcome
0	1	999	0	nonexistent
1	1	999	0	nonexistent
2	1	999	0	nonexistent
3	1	999	0	nonexistent
4	1	999	0	nonexistent

In [26]:

```
bank_o['poutcome'].unique()
```

Out[26]:

```
array(['nonexistent', 'failure', 'success'], dtype=object)
```

In [27]:

```
bank_o['poutcome'].replace(['nonexistent', 'failure', 'success'], [1,2,3], inplace = True)
```

DataSet Preprocessing

Combining all of the above changed values into a final dataset by concatenating everything and adding the attributes names. The combined dataframe would now have 20 columns with 41188 entries.

In [28]:

```
bank_final= pd.concat([bank_client, bank_related, bank_se, bank_o], axis = 1)
bank_final = bank_final[['age', 'job', 'marital', 'education', 'default', 'housing', 'loan',
'contact', 'month', 'day_of_week', 'duration', 'emp.var.rate', 'cons.price.idx',
'cons.conf.idx', 'euribor3m', 'nr.employed', 'campaign', 'pdays', 'previous']]
bank_final.shape
```

Out[28]:

```
(41188, 19)
```

Split the dataset into training set and testing set

In [29]:

```
from  sklearn.model_selection  import  train_test_split
X_train, X_test, y_train, y_test = train_test_split(bank_final, y, test_size = 0.1942313295)
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
k_fold = KFold(n_splits=10, shuffle=True, random_state=0)
```

We also import the various metrics that are crucial for the evaluation of the created models.

In [30]:

```
from sklearn.metrics import confusion_matrix, accuracy_score, precision_score,
recall_score,f1_score
```

Display the details of the Dataset

In [31]:

```
X_test.head(10)
```

Out[31]:

	age	job	marital	education	default	housing	loan	contact	month	day_of_week	duration	emp.var.rate	cons.price.idx	cons
34019	1	1	1	2	0	2	0	0	6	4	1	-1.8	92.893	
39027	3	3	1	0	0	0	0	1	2	2	3	-3.0	92.713	
5360	2	1	1	1	1	0	0	1	6	0	2	1.1	93.994	
14080	1	0	1	2	0	0	0	0	3	1	1	1.4	93.918	
6345	3	0	1	3	0	2	0	1	6	3	1	1.1	93.994	
13763	2	1	1	2	0	2	0	0	3	2	1	1.4	93.918	
11812	2	6	0	6	1	2	0	1	4	0	1	1.4	94.465	
30696	2	1	1	2	0	2	0	0	6	3	3	-1.8	92.893	
15783	2	6	1	3	0	0	0	0	3	1	4	1.4	93.918	
13289	3	0	2	3	0	2	0	0	3	4	2	1.4	93.918	

We scale the data using the StandardScaler

In [32]:

```
from sklearn.preprocessing import StandardScaler
sc_X = StandardScaler()
X_train = sc_X.fit_transform(X_train)
X_test = sc_X.transform(X_test)
```

Applying Machine Learning Models

We are going to apply 3 Machine Learning techniques on this dataset to compare which one would lead to a better result.

1. Logistic Regression
2. Support Vector Machines
3. Decision Tree Classifier

1. Logistic Regression

Logisitc Regression is a machine learning model used to model a binary dependant variable (here, bank term deposit). The model has a continuous output and the accuracy scores are given below. We apply Logistic Regression and fit the model using the Training Data. First import the LogisticRegression from sklearn and create an object logmodel. Fit the model by using the training data and then evaluate the model using the testing data.

In [33]:

```
from sklearn.linear_model import LogisticRegression
logmodel = LogisticRegression()
logmodel.fit(X_train,y_train)
logpred = logmodel.predict(X_test)
print('Confusion Matrix: \n', confusion_matrix(y_test, logpred))
print('Accuracy: ', accuracy_score(y_test, logpred))
print('Precision: ', precision_score(y_test, logpred))
print('Recall: ', recall_score(y_test, logpred))
print('f1 Score', f1_score(y_test, logpred))
LOGCV = (cross_val_score(logmodel, X_train, y_train, cv=k_fold, n_jobs=1, scoring = 'accuracy'))
```

```
Confusion Matrix:
 [[6916  146]
 [ 566  372]]
Accuracy:  0.911
Precision:  0.7181467181467182
Recall:  0.39658848614072495
f1 Score 0.5109890109890111
```

2. Support Vector Machine

Support Vector Machine (or SVM) is a classification model used to divide various points into different zones and then predict the output within a zone.We use Support Vector Machine to fit the model using the Training Data. First import the SVC from sklearn.svm and create an object named svc with sigmoid kernel. Fit the model by using the training data and then evaluate the model using the testing data.

In [34]:

```
from sklearn.svm import SVC
svc = SVC(kernel = 'sigmoid')
svc.fit(X_train, y_train)
svcpred = svc.predict(X_test)
print('Confusion Matrix: \n', confusion_matrix(y_test, svcpred))
print('Accuracy: ', accuracy_score(y_test, svcpred))
print('Precision: ', precision_score(y_test, svcpred))
print('Recall: ', recall_score(y_test, svcpred))
print('f1 Score', f1_score(y_test, svcpred))
SVCCV = (cross_val_score(svc, X_train, y_train, cv=k_fold, n_jobs=1, scoring = 'accuracy'))
```

```
Confusion Matrix:
 [[6568  494]
 [ 603  335]]
Accuracy:  0.862875
Precision:  0.4041013268998794
Recall:  0.35714285714285715
f1 Score 0.379173740803622
```

3. Decision Tree Classifier

Decisin Tree Classification is used to go from various variables and their ranged or discrete values to predict a discrete output. We now use Decision Tree to classify. First import the DecisionTreeClassifier from sklearn.tree and create an object named dtree with criterion as entropy. Fit the model by using the training data and then evaluate the model using the testing data.

In [35]:

```
from sklearn.tree import DecisionTreeClassifier
dtree  =  DecisionTreeClassifier(criterion='entropy')
dtree.fit(X_train, y_train)
dtreepred = dtree.predict(X_test)
print('Confusion Matrix: \n', confusion_matrix(y_test, dtreepred))
print('Accuracy: ', accuracy_score(y_test, dtreepred))
print('Precision: ', precision_score(y_test, dtreepred))
print('Recall: ', recall_score(y_test, dtreepred))
print('f1 Score', f1_score(y_test, dtreepred))
DTREECV = (cross_val_score(dtree, X_train, y_train, cv=k_fold, n_jobs=1, scoring = 'accuracy'))
```

```
Confusion Matrix:
 [[6591  471]
 [ 462  476]]
Accuracy:  0.883375
Precision:  0.5026399155227033
Recall:  0.5074626865671642
f1 Score 0.5050397877984084
```

Result Analysis for the regular Machine Learning Methods

Logistic Regression - 91.1%
Support Vector Machine - 86.2%
Decision Tree - 88.3%

As can be clearly seen from the accuracy reults, the best rgular model of machine learning in this case is Logistic Regression. The other Metrics Scores for Logisitc Regression as as follows:

Precision - 0.718
Recall - 0.396
f1 score - 0.51

Applying Ensemble Learning Models

In order to boost the accuracy of the models, we are going to apply these 2 following ensemble techniques.

1. Random Forest
2. XGBClassifier

1. Random Forest

Random forests are ensemble of decision tree classifications and hence generate multiple decision trees to have the most accurate results. We use Random Forest Ensemble Method to classify. First import the RandomForestClassifier from sklearn.ensemble and create an object named rfc with criterion as entropy with 200 estimators. Fit the model by using the training data and then evaluate the model using the testing data.

In [36]:

```
from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier(n_estimators = 200, criterion='entropy')#criterion = entopy,gi
rfc.fit(X_train, y_train)
rfcpred = rfc.predict(X_test)
print('Confusion Matrix: \n', confusion_matrix(y_test, rfcpred))
print('Accuracy: ', accuracy_score(y_test, rfcpred))
print('Precision: ', precision_score(y_test, rfcpred))
print('Recall: ', recall_score(y_test, rfcpred))
print('f1 Score', f1_score(y_test, rfcpred))
RFCCV = (cross_val_score(rfc, X_train, y_train, cv=k_fold, n_jobs=1, scoring = 'accuracy'))
```

```
Confusion Matrix:
 [[6798  264]
 [ 494  444]]
Accuracy:  0.90525
Precision:  0.6271186440677966
Recall:  0.47334754797441364
f1 Score 0.5394896719319562
```

2. XGBClassifier

XGBClassification or Gradient Boosting works as an ensemble of weak prediction models and then chooses the best to generate results. Next we use XGBClassifier Ensemble Method to classify. First import the XGBClassifier from xgboost and create an object named xgb. Fit the model by using the training data and then evaluate the model using the testing data.

In [37]:

```
from xgboost import XGBClassifier
xgb = XGBClassifier()
xgb.fit(X_train, y_train)
xgbprd = xgb.predict(X_test)
print('Confusion Matrix: \n', confusion_matrix(y_test, xgbprd))
print('Accuracy: ', accuracy_score(y_test, xgbprd))
print('Precision: ', precision_score(y_test, xgbprd))
print('Recall: ', recall_score(y_test, xgbprd))
print('f1 Score', f1_score(y_test, xgbprd))
XGB = (cross_val_score(estimator = xgb, X = X_train, y = y_train, cv = 10).mean())
```

```
[17:50:02] WARNING: ..\src\learner.cc:1061: Starting in XGBoost 1.3.0, the default evaluation metr
ic used with the objective 'binary:logistic' was changed from 'error' to 'logloss'. Explicitly set
eval_metric if you'd like to restore the old behavior.
Confusion Matrix:
 [[6785  277]
 [ 458  480]]
Accuracy:  0.908125
Precision:  0.6340819022457067
Recall:  0.511727078891258
f1 Score 0.5663716814159292
[17:50:04] WARNING: ..\src\learner.cc:1061: Starting in XGBoost 1.3.0, the default evaluation metr
ic used with the objective 'binary:logistic' was changed from 'error' to 'logloss'. Explicitly set
eval_metric if you'd like to restore the old behavior.
[17:50:06] WARNING: ..\src\learner.cc:1061: Starting in XGBoost 1.3.0, the default evaluation metr
ic used with the objective 'binary:logistic' was changed from 'error' to 'logloss'. Explicitly set
eval_metric if you'd like to restore the old behavior.
[17:50:08] WARNING: ..\src\learner.cc:1061: Starting in XGBoost 1.3.0, the default evaluation metr
ic used with the objective 'binary:logistic' was changed from 'error' to 'logloss'. Explicitly set
eval_metric if you'd like to restore the old behavior.
```

```
[17:50:09] WARNING: ..\src\learner.cc:1061: Starting in XGBoost 1.3.0, the default evaluation metr
ic used with the objective 'binary:logistic' was changed from 'error' to 'logloss'. Explicitly set
eval_metric if you'd like to restore the old behavior.
[17:50:11] WARNING: ..\src\learner.cc:1061: Starting in XGBoost 1.3.0, the default evaluation metr
ic used with the objective 'binary:logistic' was changed from 'error' to 'logloss'. Explicitly set
eval_metric if you'd like to restore the old behavior.
[17:50:13] WARNING: ..\src\learner.cc:1061: Starting in XGBoost 1.3.0, the default evaluation metr
ic used with the objective 'binary:logistic' was changed from 'error' to 'logloss'. Explicitly set
eval_metric if you'd like to restore the old behavior.
[17:50:15] WARNING: ..\src\learner.cc:1061: Starting in XGBoost 1.3.0, the default evaluation metr
ic used with the objective 'binary:logistic' was changed from 'error' to 'logloss'. Explicitly set
eval_metric if you'd like to restore the old behavior.
[17:50:16] WARNING: ..\src\learner.cc:1061: Starting in XGBoost 1.3.0, the default evaluation metr
ic used with the objective 'binary:logistic' was changed from 'error' to 'logloss'. Explicitly set
eval_metric if you'd like to restore the old behavior.
[17:50:18] WARNING: ..\src\learner.cc:1061: Starting in XGBoost 1.3.0, the default evaluation metr
ic used with the objective 'binary:logistic' was changed from 'error' to 'logloss'. Explicitly set
eval_metric if you'd like to restore the old behavior.
[17:50:20] WARNING: ..\src\learner.cc:1061: Starting in XGBoost 1.3.0, the default evaluation metr
ic used with the objective 'binary:logistic' was changed from 'error' to 'logloss'. Explicitly set
eval_metric if you'd like to restore the old behavior.
```

Result Analysis for Ensemble Methods

Random Forest - 90.5%
XGBClassifier - 90.8%

As can be clearly seen from the accuracy reults, the best rgular model of machine learning in this case is XGBClassifier. The other Metrics Scores for Logisitc Regression as as follows:

Precision - 0.635
Recall - 0.511
f1 score - 0.566

In [38]:

```
RFCCV=np.max(RFCCV)
DTREECV=np.max(DTREECV)
SVCCV=np.max(SVCCV)
LOGCV=np.max(LOGCV)
models = pd.DataFrame({'Models': ['Random Forest Classifier', 'Decision Tree Classifier', 'Support
Vector',
          'Logistic Model', 'XGBoost'],
          'Score': [RFCCV, DTREECV, SVCCV, LOGCV, XGB]})
models.head()
```

Out[38]:

	Models	Score
0	Random Forest Classifier	0.916541
1	Decision Tree Classifier	0.895149
2	Support Vector	0.874058
3	Logistic Model	0.916842
4	XGBoost	0.911956

In [39]:

```
models.sort_values(by='Score',  ascending=False)
```

Out[39]:

	Models	Score
3	Logistic Model	0.916842
0	Random Forest Classifier	0.916541
4	XGBoost	0.911956

	Models	Score
1	Decision Tree Classifier	0.895149
2	Support Vector	0.874058