**SCHOOL OF COMPUTER SCIENCE AND ENGINEERING**

**JUNE, 2021**

# BANK MARKETING

# A PROJECT REPORT

***Submitted by***


KSHITIZ CHOUDHARY - 18BCE0606


Course Code: CSE 3013

Course Title: Artificial Intelligence

Under the guidance of

**Dr. S. Anto**

**Associate Professor, SCOPE,**

**VIT, Vellore.**

# 1 Table of Contents

## 2 Index of Tables



## 3 Index of Figures

# 1. INTRODUCTION

## 1.1. 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 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.

## 1.2. 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 predictivemodels 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 makingpredictions using existing dataset in banking marketing. To find how they can be used together ina process of converting raw data to effective decision-making knowledge. Building the predictivemodels 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 composedon 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 preparation methods 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.

# 2. LITERATURE SURVEY

## 2.1. PAPERS REFERED

*Table 1 Paper Refered*

| 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 | Unnamed Portuguese bank dataset with 17 features. | Better Ensemble Models Can be used. |

| | | | | | |
|---|---|---|---|---|---|
| | | | unbalanced dataset. | | |
| 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 | UCI Irvine machine learning repository | Better Pre-processing can be done. |

| | | | | | |
|---|---|---|---|---|---|
| | | | the correct metric for this business model | | |

## 2.2. BASE PAPERS

*Table 2 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. |

## 2.3. PROBLEM DEFINITION

With the advancement of data processing technology and changes in the competitive environment, the cooperation of strategy and technology is signaling a customer-oriented era. Currently commercial banks are highly competitive, and business insights based on data analysis and mining are becoming the core competitiveness of commercial banks in the era of big data. Marketing is technique of exposing the target clients to a product via suitable systems and channels. It ultimately facilitates the way to buy the product or service and even helps in determining the need of the product and persuade customers to buy it. The overall aim is to increase sales of products and services for enterprise, business and financial institutions. It also helps to maintain the reputation of the company.

# 3. OVERVIEW OF THE WORK

## 3.1. OBJECTIVES OF THE PROJECT

1. Provide the best model to predict customer response in a European bank marketing campaign.
2. Understand which features in the database (which attribute of user) has most impact on the response
3. Aim is to provide a methodology and Algorithm that gives better results with the prediction and analysis than existing systems.

## 3.2. SOFTWARE REQUIREMENTS

1. Jupyter Notebook
2. Ubuntu Operating

## 3.3. HARDWARE REQUIREMENTS

1. 8 GB RAM
2. 4/8 core processor machine

# 4. SYSTEM DESIGN

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 datasets. 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.

## 4.1. 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.

## 4.2. 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.

## 4.3. MACHINE LEARNING ALGORITHMS

### 4.3.1. LOGISTIC REGRESSION

Logistic Regression is a machine learning model used to model a binary dependent variable (here, bank term deposit). We apply Logistic Regression andfit 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.

### 4.3.2. 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.

### 4.3.3. 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.

### 4.3.4. 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

#### 4.3.4.1. RANDOM FORREST

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.

#### 4.3.4.2. 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.

# 5. IMPLEMENTATION

## 5.1. DESCRIPTION OF MODULES/PROGRAMS

### 1. Numpy

NumPy is a Python library used for working with arrays. It also has functions for working in domain of linear algebra, fourier transform, and matrices. NumPy stands for Numerical Python. In Python we have lists that serve the purpose of arrays, but they are slow to process. NumPy aims to provide an array object that is up to 50x faster than traditional Python lists. The array object in NumPy is called ndarray, it provides a lot of supporting functions that make working with ndarray very easy. Arrays are very frequently used in data science, where speed and resources are very important.

### 2. Pandas

Pandas is a Python package that provides fast, flexible, and expressive data structures designed to make working with structured (tabular, multidimensional, potentially heterogeneous) and time series data both easy and intuitive. It aims to be the fundamental high-level building block for doing practical, real world data analysis in Python. Additionally, it has the broader goal of becoming the most powerful and flexible open-source data analysis / manipulation tool available in any language. It is already well on its way toward this goal.

### 3. Matplotlib

Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python.
Matplotlib produces publication-quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shell, web application servers, and various graphical user interface toolkits.

## 4. Seaborn

Seaborn is a library for making statistical graphics in Python. It is built on top of matplotlib and closely integrated with pandas data structures.Here is some of the functionality that seaborn offers:

- A dataset-oriented API for examining relationships between multiple variables
- Specialized support for using categorical variables to show observations or aggregate statistics
- Options for visualizing univariate or bivariate distributions and for comparing them between subsets of data
- Automatic estimation and plotting of linear regression models for different kinds dependent variables
- Convenient views onto the overall structure of complex datasets
- High-level abstractions for structuring multi-plot grids that let you easily build complex visualizations
- Concise control over matplotlib figure styling with several built-in themes
- Tools for choosing color palettes that faithfully reveal patterns in your data

## 5. Sklearn

Scikit-learn (Sklearn) is the most useful and robust library for machine learning in Python. It provides a selection of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction via a consistence interface in Python. This library, which is largely written in Python, is built upon NumPy, SciPy and Matplotlib.

# 6. OUTPUT AND PERFORMANCE ANALYSIS

## 6.1. EXECUTION SNAPSHOTS

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

```python
url = 'https://raw.githubusercontent.com/Kshitiz2603/Bank-Marketing/main/bank-additional-full.csv'
bank = pd.read_csv(url,sep=';')
y = pd.get_dummies(bank['y'], columns = ['y'], prefix = ['y'], drop_first = True)
bank.head(10)
```

| | 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.employe |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 56 | housemaid | married | basic.4y | no | no | no | telephone | may | mon | 261 | 1 | 999 | 0 | nonexistent | 1.1 | 93.994 | -36.4 | 4.857 | 5191. |
| 1 | 57 | services | married | high.school | unknown | no | no | telephone | may | mon | 149 | 1 | 999 | 0 | nonexistent | 1.1 | 93.994 | -36.4 | 4.857 | 5191. |
| 2 | 37 | services | married | high.school | no | yes | no | telephone | may | mon | 226 | 1 | 999 | 0 | nonexistent | 1.1 | 93.994 | -36.4 | 4.857 | 5191. |
| 3 | 40 | admin. | married | basic.6y | no | no | no | telephone | may | mon | 151 | 1 | 999 | 0 | nonexistent | 1.1 | 93.994 | -36.4 | 4.857 | 5191. |
| 4 | 56 | services | married | high.school | no | no | yes | telephone | may | mon | 307 | 1 | 999 | 0 | nonexistent | 1.1 | 93.994 | -36.4 | 4.857 | 5191. |
| 5 | 45 | services | married | basic.9y | unknown | no | no | telephone | may | mon | 198 | 1 | 999 | 0 | nonexistent | 1.1 | 93.994 | -36.4 | 4.857 | 5191. |
| 6 | 59 | admin. | married | professional.course | no | no | no | telephone | may | mon | 139 | 1 | 999 | 0 | nonexistent | 1.1 | 93.994 | -36.4 | 4.857 | 5191. |
| 7 | 41 | blue-collar | married | unknown | unknown | no | no | telephone | may | mon | 217 | 1 | 999 | 0 | nonexistent | 1.1 | 93.994 | -36.4 | 4.857 | 5191. |
| 8 | 24 | technician | single | professional.course | no | yes | no | telephone | may | mon | 380 | 1 | 999 | 0 | nonexistent | 1.1 | 93.994 | -36.4 | 4.857 | 5191. |
| 9 | 25 | services | single | high.school | no | yes | no | telephone | may | mon | 50 | 1 | 999 | 0 | nonexistent | 1.1 | 93.994 | -36.4 | 4.857 | 5191. |

```python
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
```

```python
bank.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
```

```
[ ] bank.columns
```

```
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')
```

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

| | 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 |

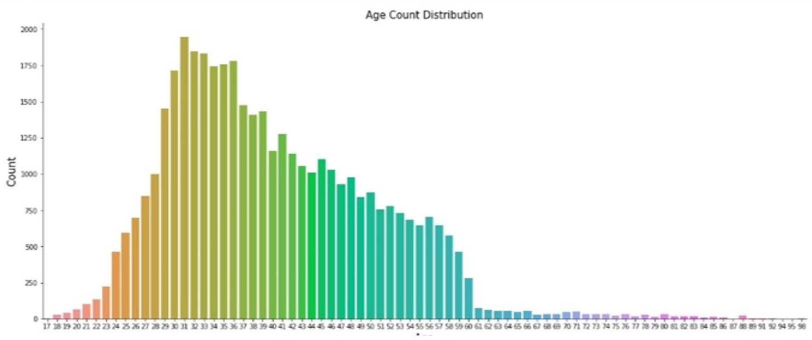
```
] print('Jobs:\n', bank_client['job'].unique())
  print('Marital:\n', bank_client['marital'].unique())
  print('Education:\n', bank_client['education'].unique())
  print('Default:\n', bank_client['default'].unique())
  print('Housing:\n', bank_client['housing'].unique())
  print('Loan:\n', bank_client['loan'].unique())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

```
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']
```

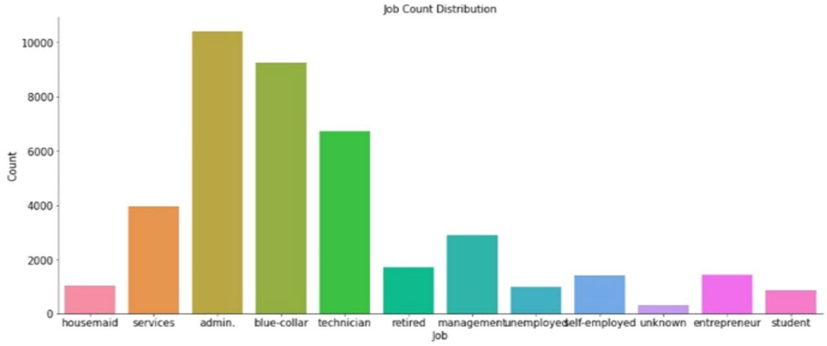
```
] print('Max age: ', bank_client['age'].max())
  print('Min age: ', bank_client['age'].min())
  print('Null Values: ', bank_client['age'].isnull().any())
```

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

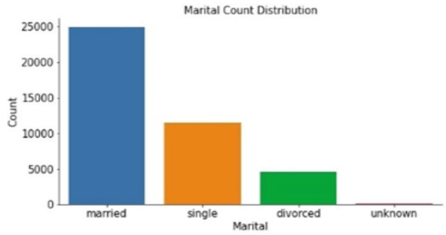
```
fig, ax = plt.subplots()
fig.set_size_inches(20, 8)
sns.countplot(x = 'age', data = bank_client)
ax.set_xlabel('Age', fontsize=15)
ax.set_ylabel('Count', fontsize=15)
ax.set_title('Age Count Distribution', fontsize=15)
sns.despine()
```



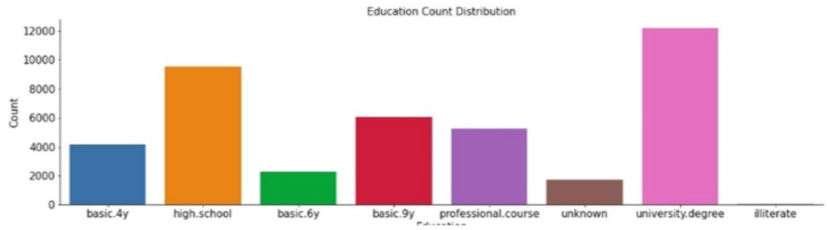
```
fig, ax = plt.subplots()
fig.set_size_inches(20, 8)
sns.countplot(x = 'job', data = bank_client)
ax.set_xlabel('Job', fontsize=15)
ax.set_ylabel('Count', fontsize=15)
ax.set_title('Job Count Distribution', fontsize=15)
ax.tick_params(labelsize=15)
sns.despine()
```



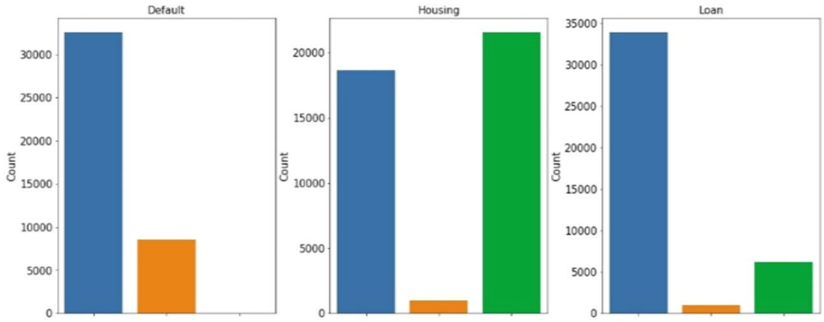

```
fig, ax = plt.subplots()
fig.set_size_inches(10, 5)
sns.countplot(x = 'marital', data = bank_client)
ax.set_xlabel('Marital', fontsize=15)
ax.set_ylabel('Count', fontsize=15)
ax.set_title('Marital Count Distribution', fontsize=15)
ax.tick_params(labelsize=15)
sns.despine()
```




```
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()
```



```
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)
```




```
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'])
```

```
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);
```

```
bank_client.head()
```

| | 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 |

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

(41188, 7)

| | 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 |

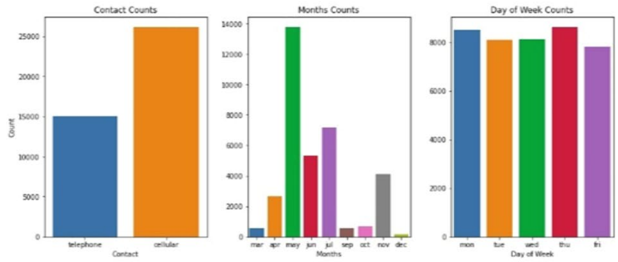
```
bank_related = bank.iloc[: , 7:11]
bank_related.head()
```

| | 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 |

```
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']
```

```
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','oct','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)
```




```
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'])
```

```
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);
```

```
bank_related.head()
```

| | 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 |

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

| | 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 |

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

| | 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 |

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

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

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

```python
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
```

```
(41188, 19)
```

```python
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(bank_final, y, test_size = 0.33)
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)
```

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

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

| | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 4874 | 2 | 0 | 1 | 3 | 0 | 1 | 1 | 1 | 6 | 4 | 2 | 1.1 | 93.994 | -36.4 | 4.858 | 5191.0 | 2 | 999 | 0 |
| 16274 | 2 | 7 | 2 | 3 | 0 | 0 | 2 | 0 | 3 | 3 | 1 | 1.4 | 93.918 | -42.7 | 4.961 | 5228.1 | 2 | 999 | 0 |
| 38335 | 2 | 0 | 2 | 3 | 0 | 0 | 0 | 0 | 8 | 0 | 1 | -3.4 | 92.431 | -26.9 | 0.739 | 5017.5 | 1 | 999 | 0 |
| 32794 | 1 | 4 | 2 | 6 | 0 | 2 | 0 | 0 | 6 | 1 | 2 | -1.8 | 92.893 | -46.2 | 1.299 | 5099.1 | 1 | 999 | 0 |
| 15003 | 2 | 1 | 1 | 2 | 1 | 2 | 0 | 1 | 3 | 2 | 3 | 1.4 | 93.918 | -42.7 | 4.958 | 5228.1 | 3 | 999 | 0 |
| 10418 | 1 | 1 | 2 | 3 | 0 | 2 | 0 | 1 | 4 | 1 | 5 | 1.4 | 94.465 | -41.8 | 4.960 | 5228.1 | 2 | 999 | 0 |
| 11043 | 3 | 9 | 1 | 5 | 1 | 2 | 0 | 1 | 4 | 4 | 4 | 1.4 | 94.465 | -41.8 | 4.962 | 5228.1 | 13 | 999 | 0 |
| 25645 | 2 | 9 | 1 | 3 | 0 | 2 | 0 | 0 | 7 | 4 | 2 | -0.1 | 93.200 | -42.0 | 4.120 | 5195.8 | 1 | 999 | 0 |
| 13095 | 1 | 0 | 2 | 2 | 0 | 0 | 0 | 1 | 3 | 3 | 3 | 1.4 | 93.918 | -42.7 | 4.962 | 5228.1 | 5 | 999 | 0 |
| 24405 | 2 | 10 | 1 | 2 | 0 | 2 | 0 | 0 | 7 | 1 | 4 | -0.1 | 93.200 | -42.0 | 4.191 | 5195.8 | 1 | 999 | 0 |

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

```python
import sklearn
import math
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))
mse = sklearn.metrics.mean_squared_error(y_test, logpred)
rmse = math.sqrt(mse)
print('Root Mean Square Error: ',rmse)
LOGCV = (cross_val_score(logmodel, X_train, y_train, cv=k_fold, n_jobs=1, scoring = 'accuracy').mean())
```

```
Confusion Matrix:
 [[11798   310]
 [  909   576]]
Accuracy:  0.9103214890016921
Precision:  0.6501128668171557
Recall:  0.3878787878787879
f1 Score 0.4858709405314214
Root Mean Square Error:  0.2994637056444536
```

```
[ ] 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))
    mse = sklearn.metrics.mean_squared_error(y_test, svcpred)
    rmse = math.sqrt(mse)
    print('Root Mean Square Error: ',rmse)
    SVCCV = (cross_val_score(svc, X_train, y_train, cv=k_fold, n_jobs=1, scoring = 'accuracy').mean())
```

```
Confusion Matrix:
 [[11164   944]
 [  903   582]]
Accuracy:  0.8641212388729493
Precision:  0.381389252948886
Recall:  0.3919191919191919194
f1 Score 0.3865825307206908
Root Mean Square Error:  0.368617364114946
```

```
[ ] 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))
    mse = sklearn.metrics.mean_squared_error(y_test, dtreepred)
    rmse = math.sqrt(mse)
    print('Root Mean Square Error: ',rmse)
    DTREECV = (cross_val_score(dtree, X_train, y_train, cv=k_fold, n_jobs=1, scoring = 'accuracy').mean())
```

```
Confusion Matrix:
 [[11296   812]
 [  706   779]]
Accuracy:  0.8883248730964467
Precision:  0.4896291640477687
Recall:  0.5245791245791246
f1 Score 0.5065019505851754
```

```
[ ] import sklearn
    import math
    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))
    mse = sklearn.metrics.mean_squared_error(y_test, rfcpred)
    rmse = math.sqrt(mse)
    print('Root Mean Square Error: ',rmse)
    RFCCV = (cross_val_score(rfc, X_train, y_train, cv=k_fold, n_jobs=1, scoring = 'accuracy').mean())
```

```
Confusion Matrix:
 [[11640   468]
 [  730   755]]
Accuracy:  0.9118664018244684
Precision:  0.6173344235486509
Recall:  0.5084175084175084
f1 Score 0.5576070901033974
Root Mean Square Error:  0.29687303376280494
```

```
[ ] 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))
    mse = sklearn.metrics.mean_squared_error(y_test, xgbprd)
    rmse = math.sqrt(mse)
    print('Root Mean Square Error: ',rmse)
    XGB = (cross_val_score(estimator = xgb, X = X_train, y = y_train, cv = 10).mean())
```

```
Confusion Matrix:
 [[11725   383]
 [  755   730]]
Accuracy:  0.9162804384609725
Precision:  0.655884995507637
Recall:  0.49158249158249157
f1 Score 0.5619707467282525
Root Mean Square Error:  0.28934332813981983
```

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

| | 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 |

### 6.2. OUTPUT – IN TERMS OF PERFORMANCE METRICS

#### 6.2.1. CROSS VALIDATION SCORE

*Table 3 Cross Validation Score of Models used*

| | 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 |

#### 6.2.2. ACCURACY AND RMSE COMPARISON

*Table 4 Accuracy and RMSE comparison for models used*

| 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% |

## 7. CONCLUSION AND FUTURE DIRECTIONS

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%.

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 client's 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.

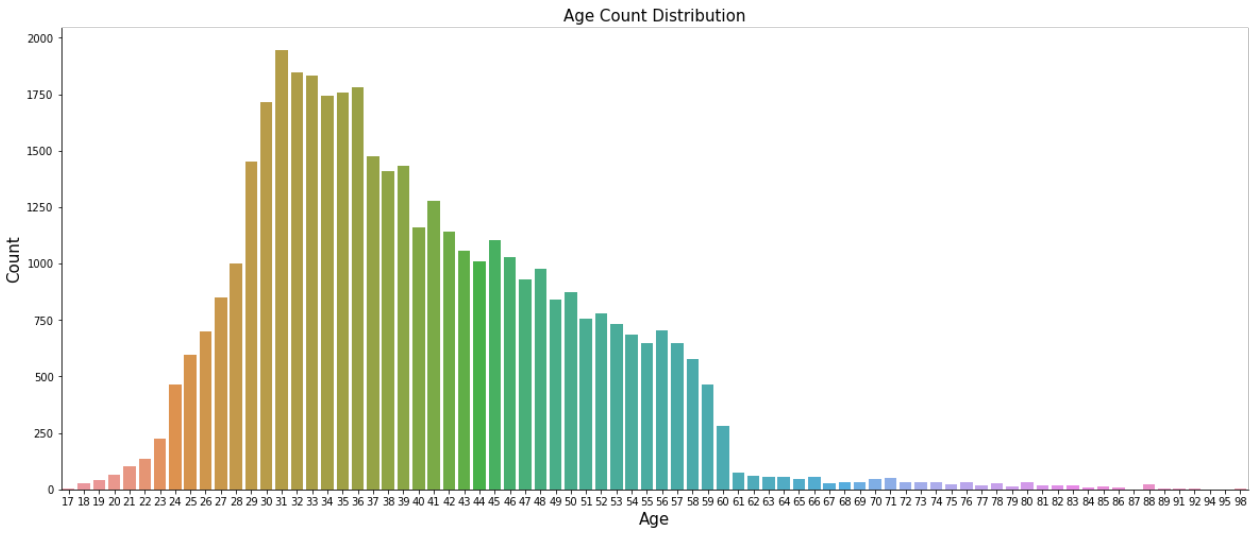



*Figure 1 Age Count Distribution*

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

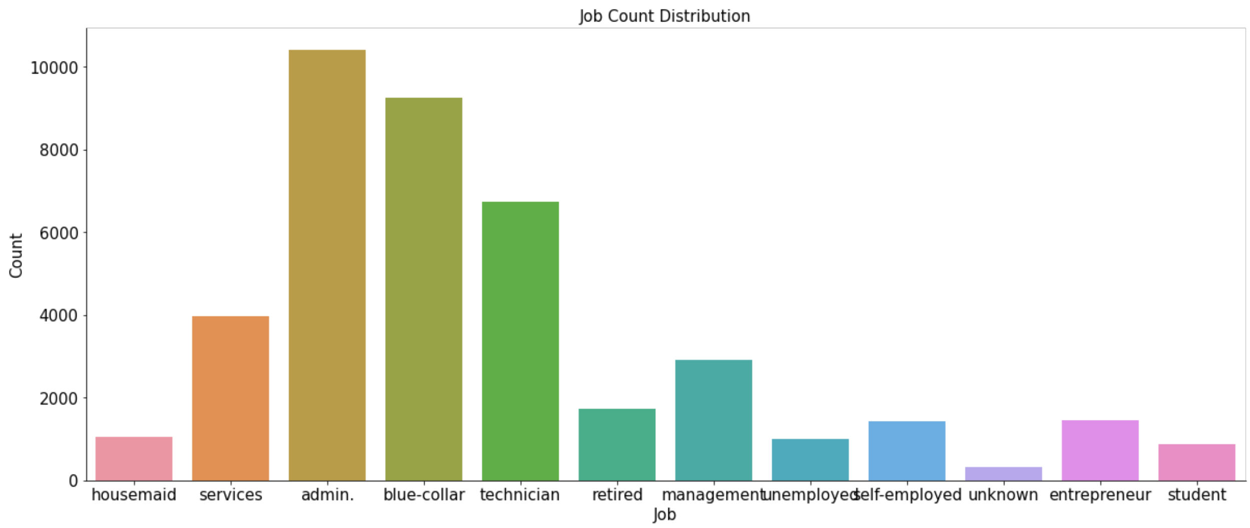



*Figure 2 Job Count Distribution*

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

.

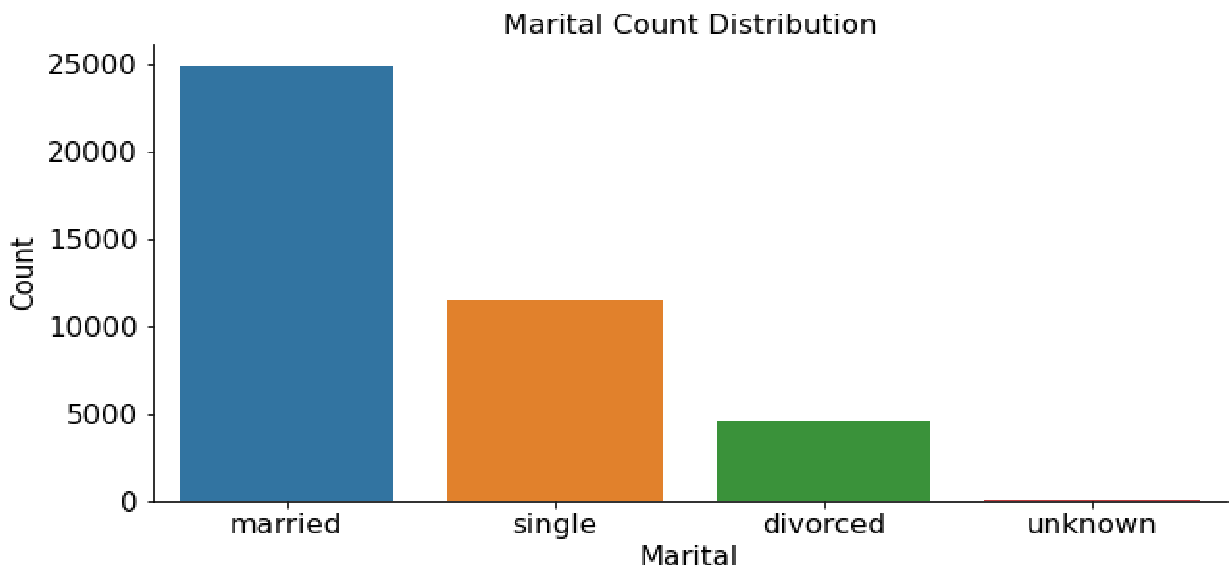



*Figure 3Marital Count Distribution*

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

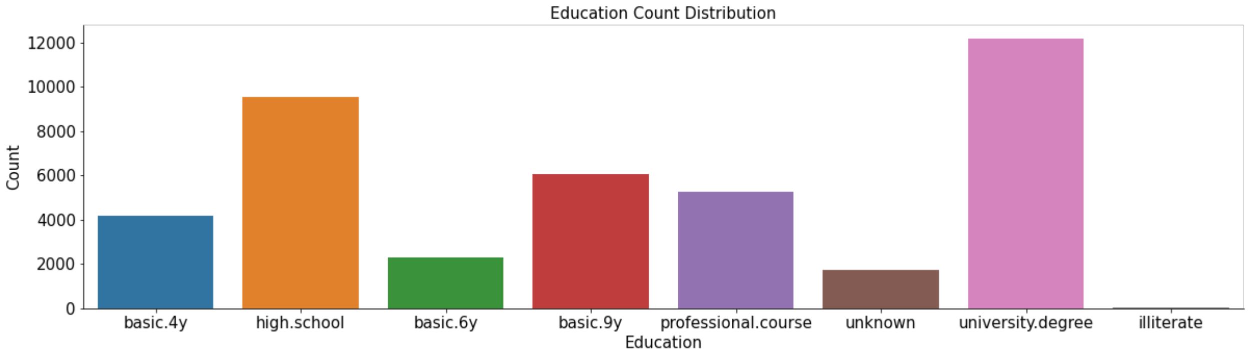



*Figure 4 Education Count Distribution*

This is the education count distribution bar plot which signifies that the majority of customers have university degrees.

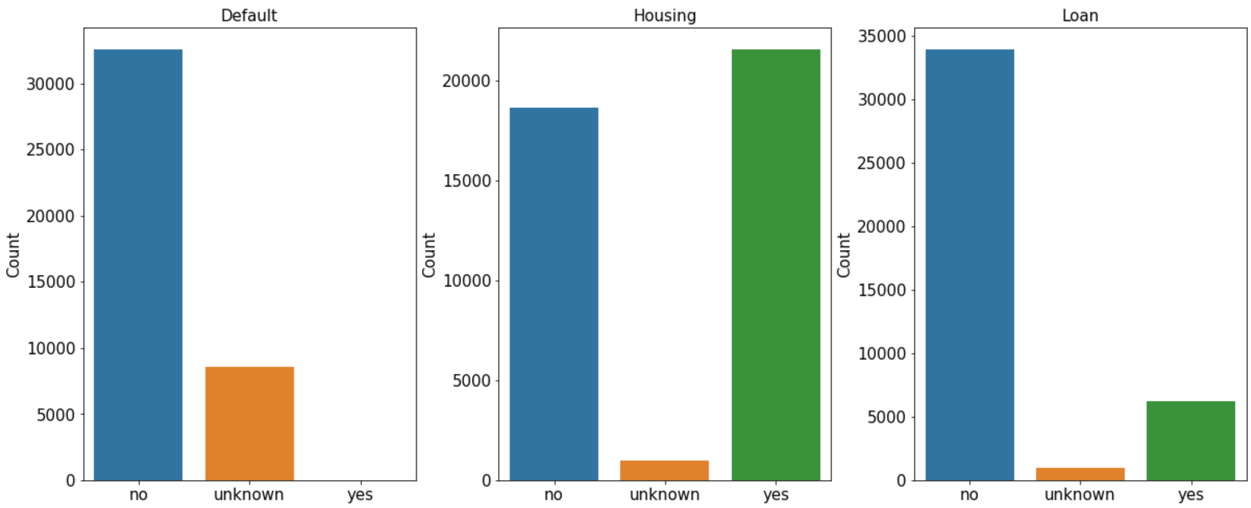



*Figure 5 Default, Housing and Loan Count Distribution*

From the default, housing loan and personal loan distribution barplot it has been deduced that themajority 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.

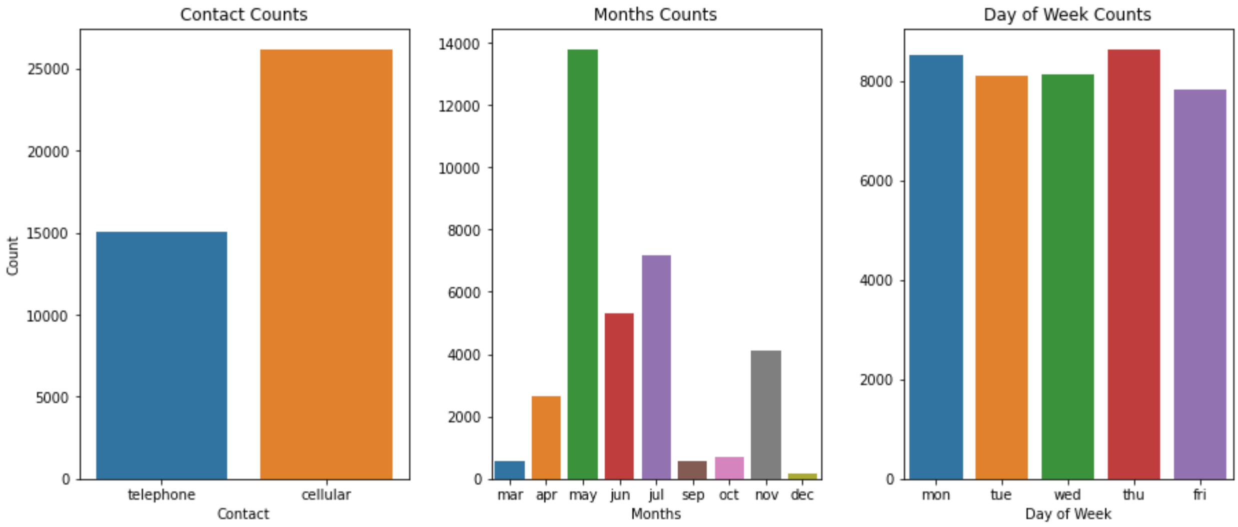



*Figure 6 Contact, Months and Days of the week Count Distribution*

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

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

## 8. 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 Communication Engineering*. Springer, Singapore, 2020. 681-691.