ANALYSIS OF BANK MARKETING DATASET

MACHINE LEARNING CLASSIFICATION METHOD

GROUP 6

K.Navya Poojitha

Saswat

Priyanka

Manohar

Steps of Machine Learning Analysis

1. DATA PRE-PROCESSING
2. EXPLORATORY DATA ANALYSIS
3. TRAINING THE MODEL
4. DETERMINING MODEL ACCURACY
5. COMPARING THE RESUTS WITH OTHER MODELS
6. CONCLUSION AND INSIGHTS

About The Data

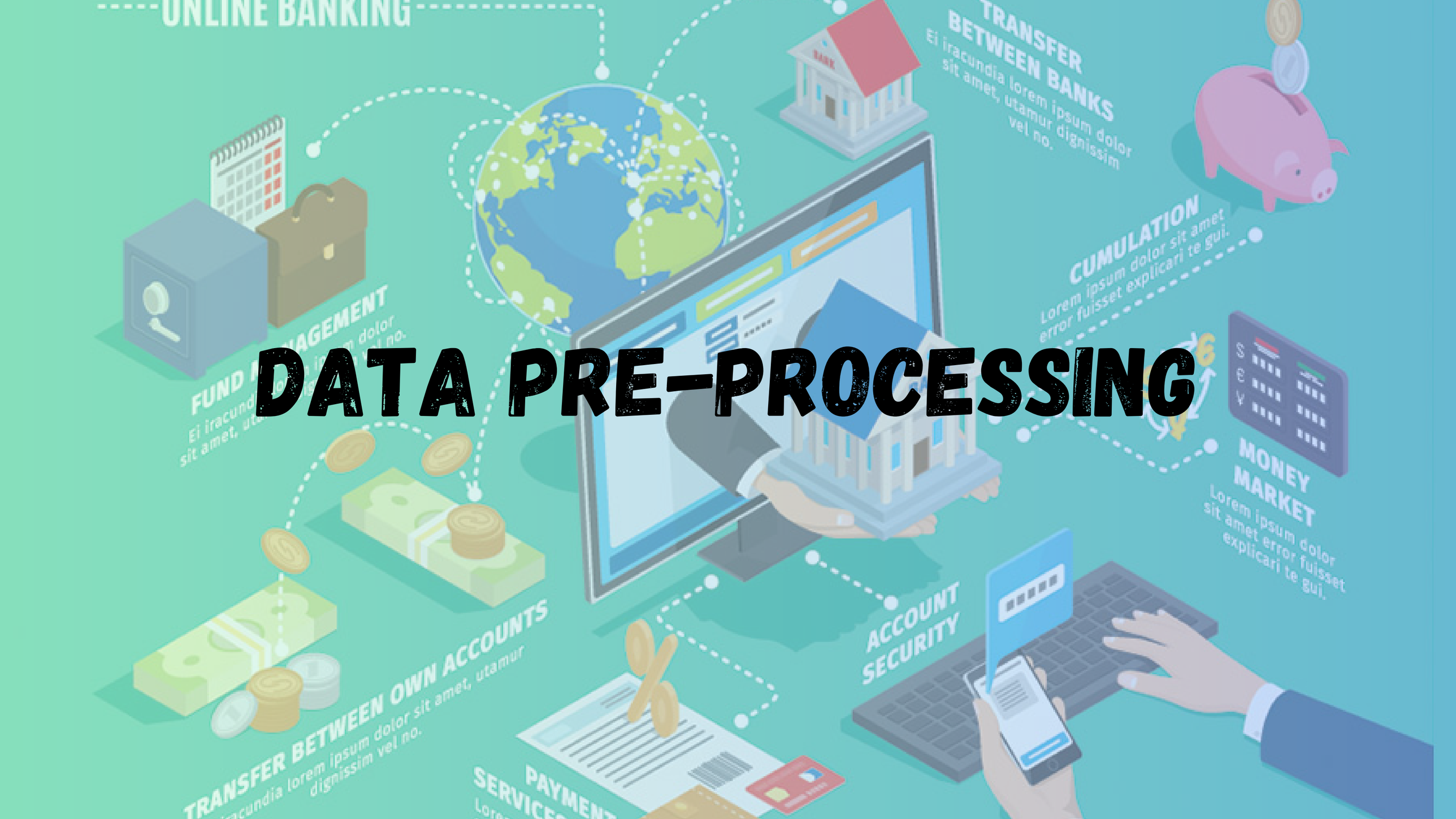
This data set contains records relevant to a direct marketing campaign of a Portuguese banking institution. The marketing campaign was executed through phone calls. Often, more than one call needs to be made to a single client before they either decline or agree to a term deposit subscription. The classification goal is to predict if the client will subscribe (yes/no) to the term deposit (variable y).

Objective

Marketing plays a crucial role in the banking industry, as **it helps to create brand awareness, attract and retain customers, increase profitability, and drive business growth, and with an aim to maximize profits and achieve customer satisfaction.**

The Path

Implementing various Machine Learning models to fit the best model for the dataset



DATA PRE-PROCESSING

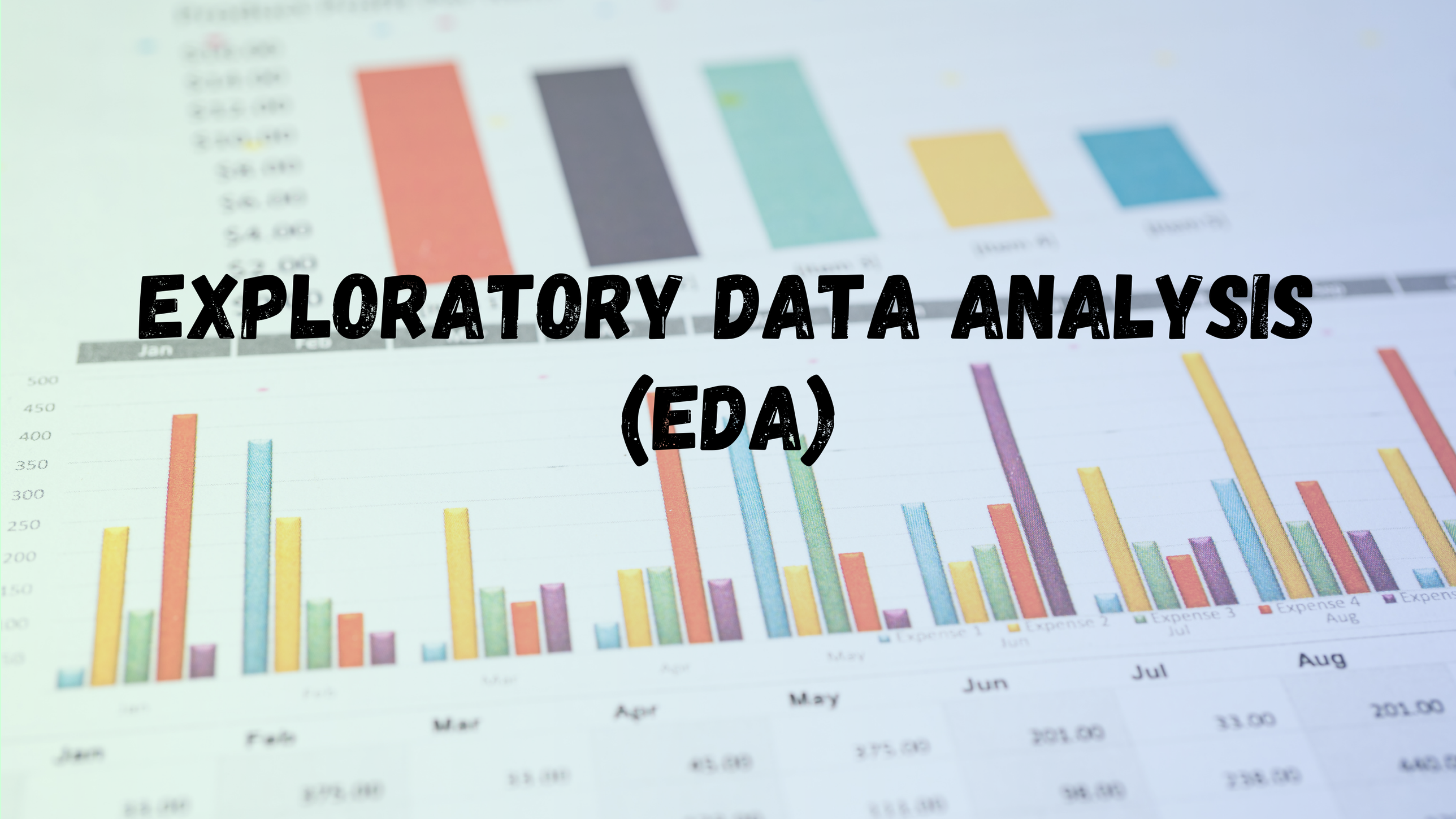
Data and Data Quality Check :

Description	Details
Total Rows	4521
Total columns	17
Missing values	None
Dropped columns	None

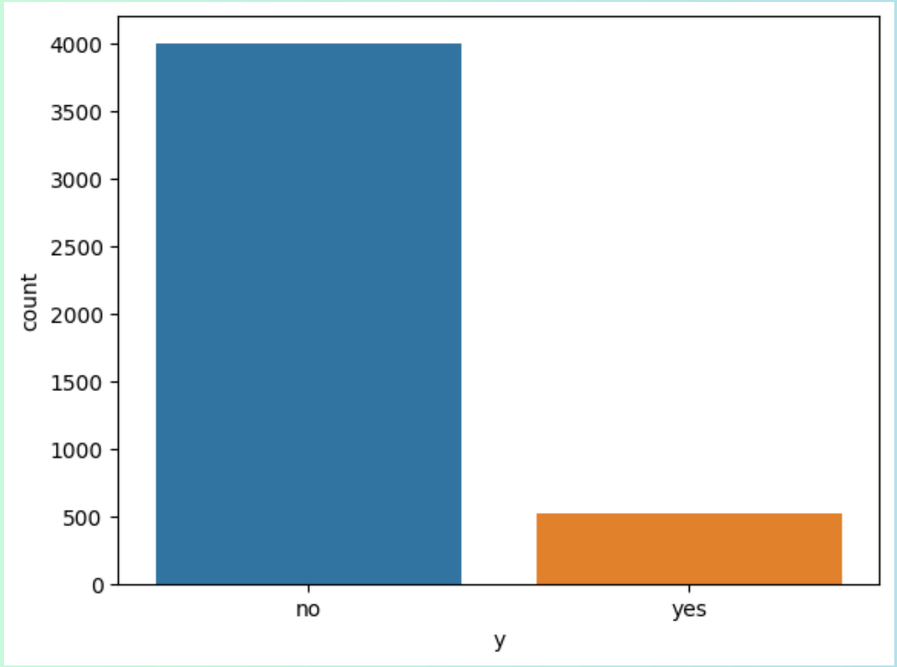
Variables :

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)
default	has credit in default? (categorical: 'no','yes','unknown')
education	(categorical, primary, tertiary, secondary):
loan	has housing loan? (categorical: 'no','yes','unknown')
contact	contact communication type (categorical: 'cellular','unknown')
campaign	number of contacts performed during this campaign and for this client (numeric, includes last contact)

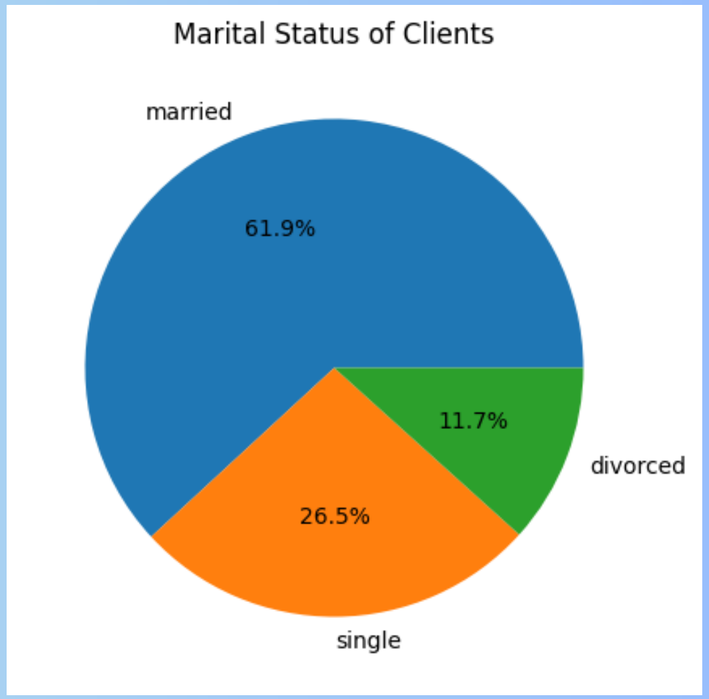
month	last contact month of year (categorical: 'jan', 'feb', 'mar', ..., 'nov', 'dec')
balance	average yearly balance, in euros (numeric)
duration	last contact duration, in seconds (numeric)
housing	has any house loan (yes/no)
pdays	number of contacts performed during this campaign and for this client (numeric, includes last contact)
previous	outcome of the previous marketing campaign (categorical:unknown","other","failure","success")
poutcome	number of days that passed by after the client was last contacted from a previous campaign (numeric, -1 means client was not previously contacted)
day	last contact day of the month(numeric)
y	has the client subscribed a term deposit? (binary: 'yes','no')


EXPLORATORY DATA ANALYSIS
(EDA)

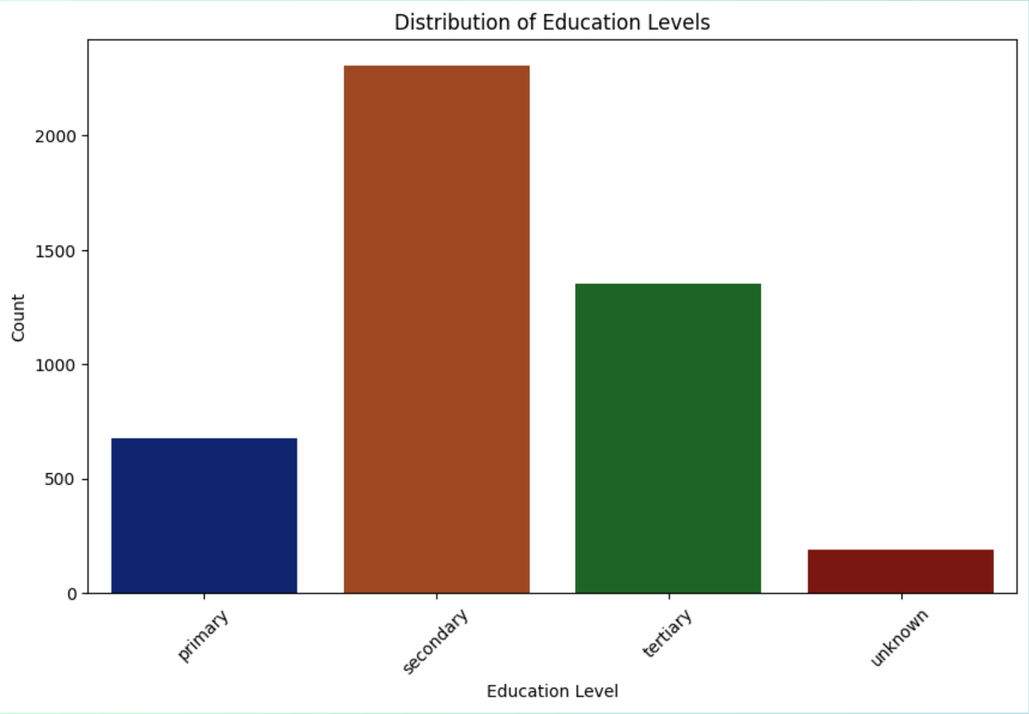
DATA VISUALIZATION



Count plot representing number of clients will subscribe (yes/no) to the term deposit (variable y)

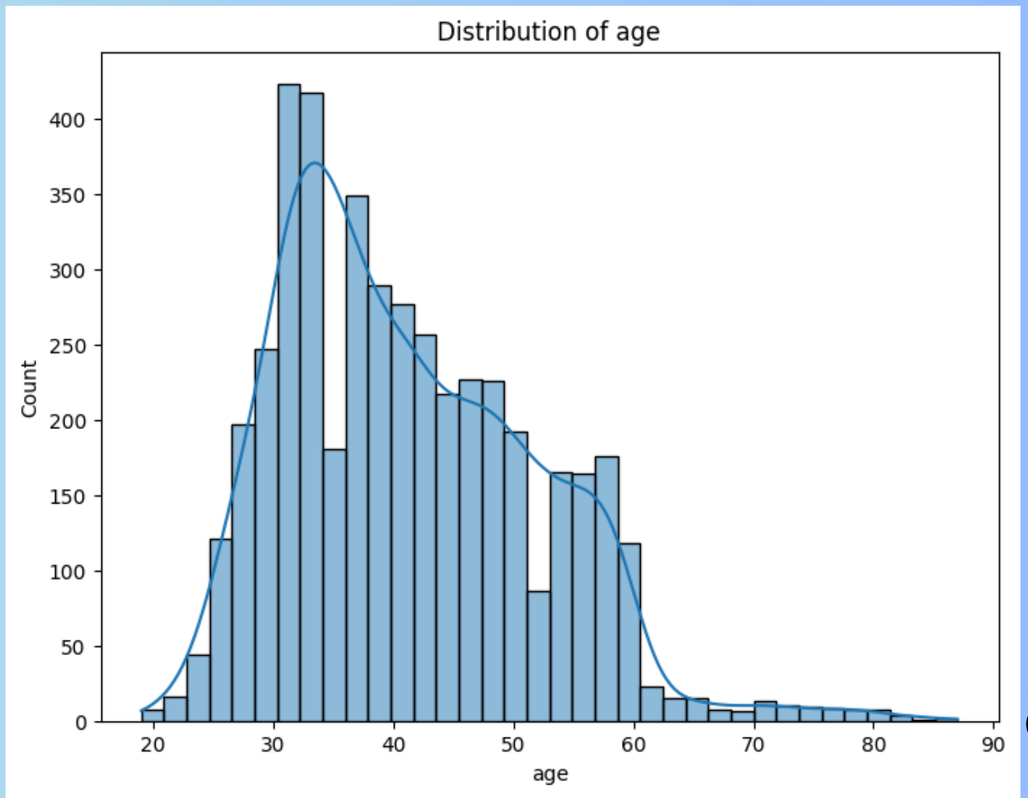


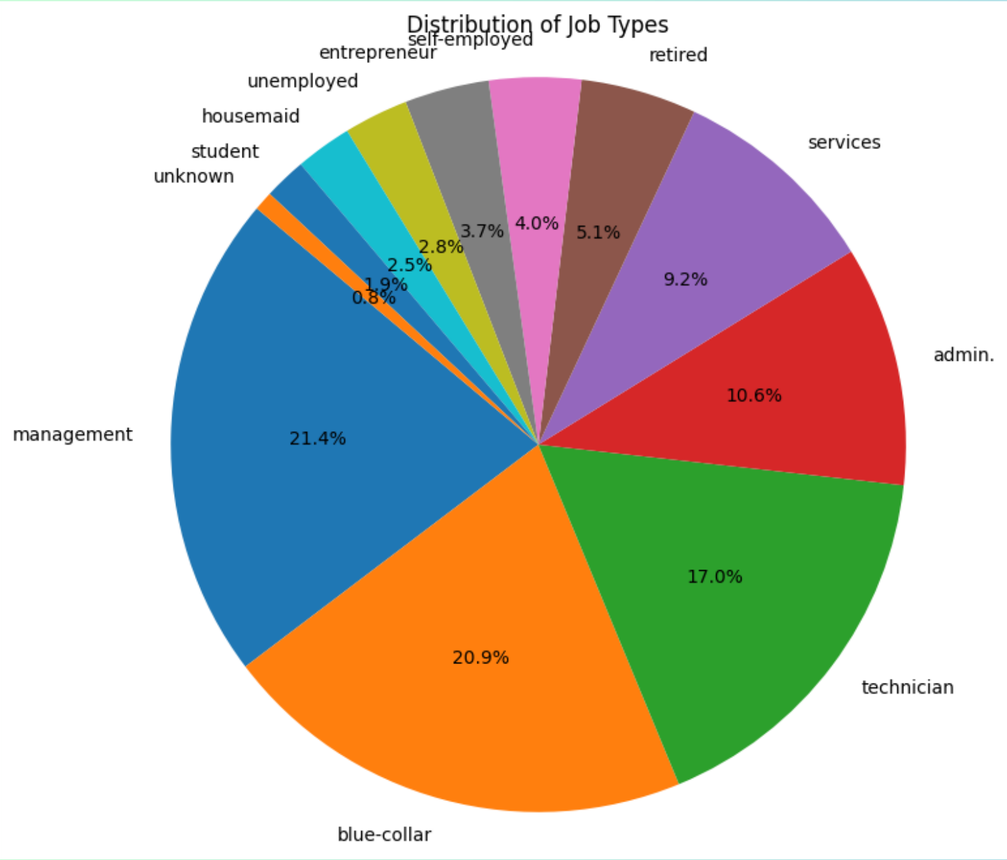
Marital status of the client based on the Bank Marketing Dataset



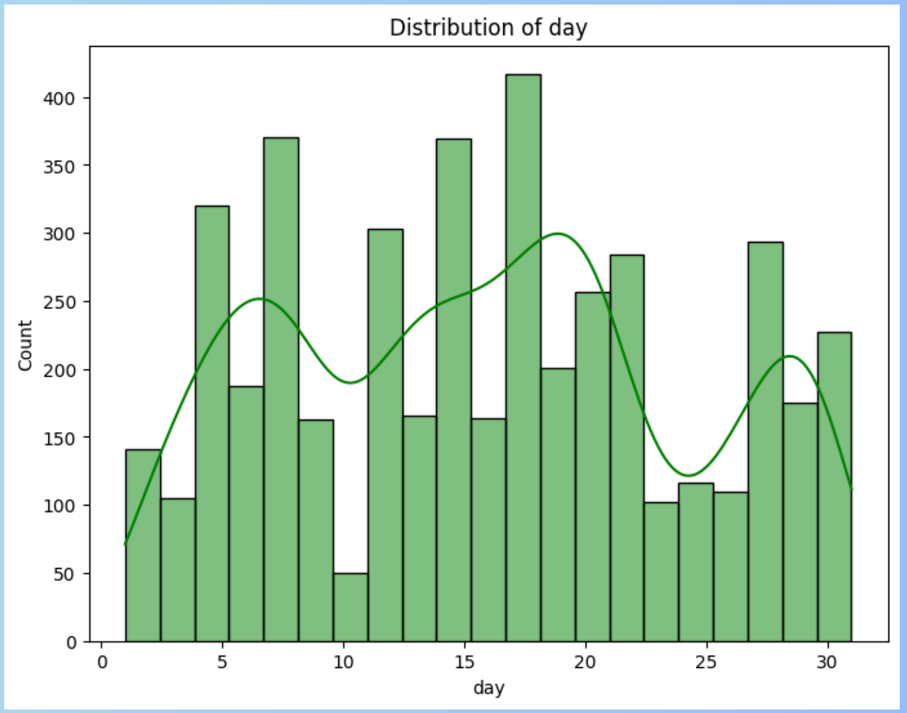
Distribution of education levels based on the Dataset.

Distribution of age of the clients based on the “bank marketing dataset”

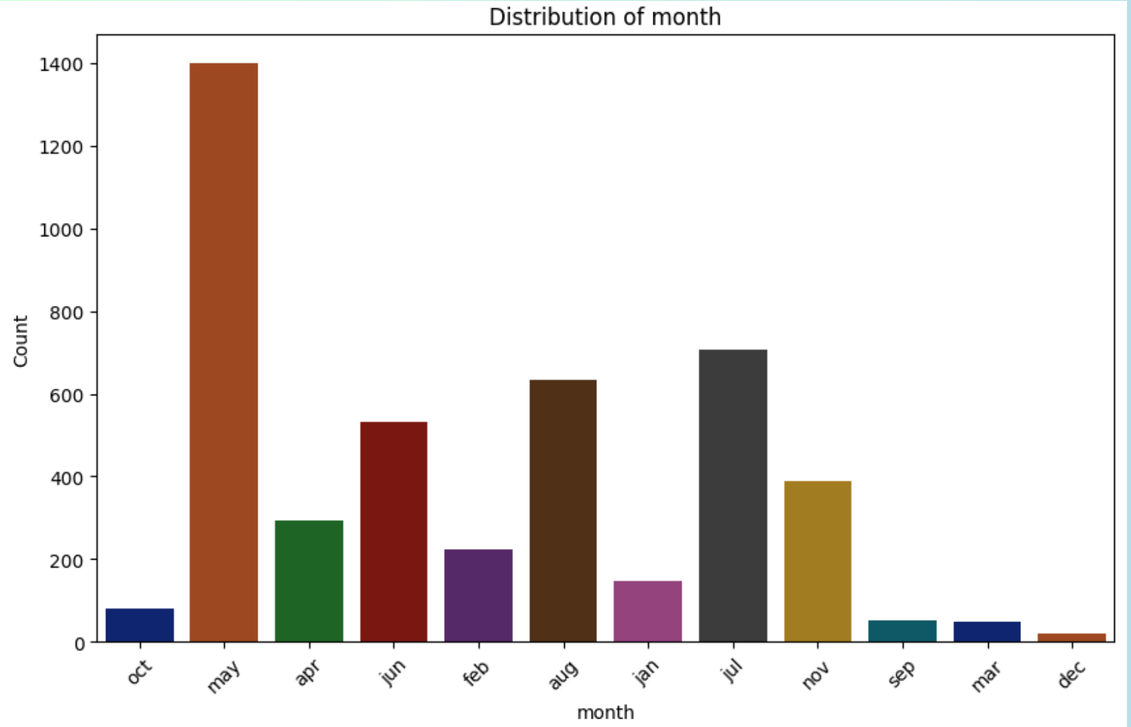




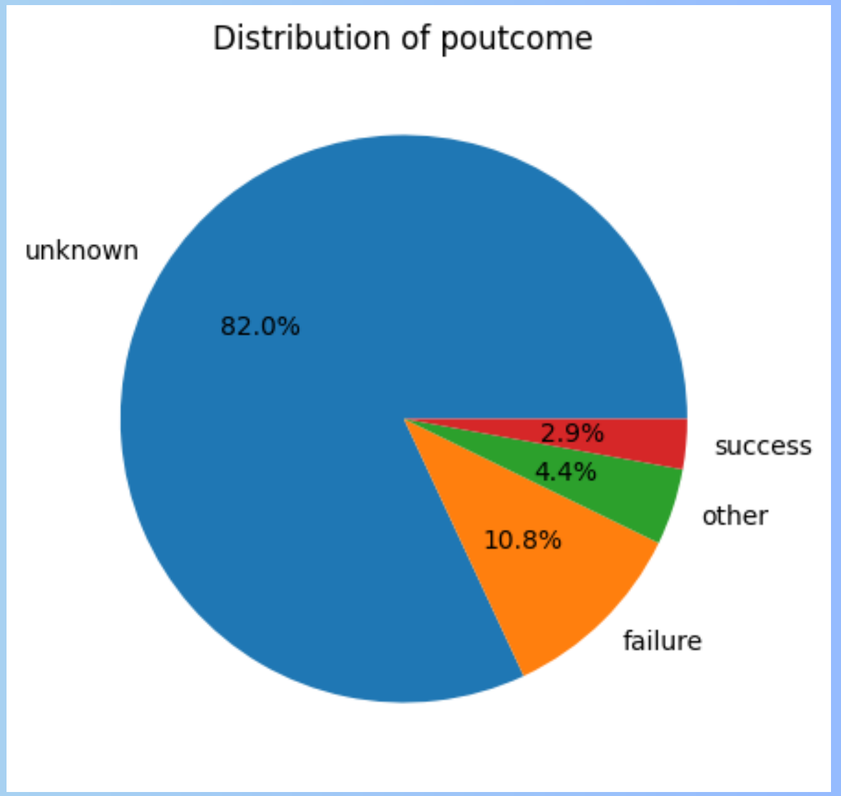
The above pie chart represents the distribution of jobs according to the Bank Marketing dataset



The hist plot represents number of clients who have subscribed the term deposit per day of the month.



The above is the plot represents the distribution of month which says, when the customer last contacted in the year to the bank.



This represents the distribution of poutcome which says the outcome of the previous marketing campaign.

Heatmap :

“Our heat map shows the relationship between several different variables.”

“Light colors represent weak relationships, while dark colors represents stronger relationships.”

Dummy Variable Encoding :

- In job column as there are many types of job included in it and all are categorical so we replaced it with the numbers 0,1,2 to make it numeric

- In the education column as primary, secondary and tertiary are categorical variables we replaced it with numbers 0,1,2 to make the columns in numeric

- And in the marital column also we replaced the single and married categorical columns to 0,1 numeric columns

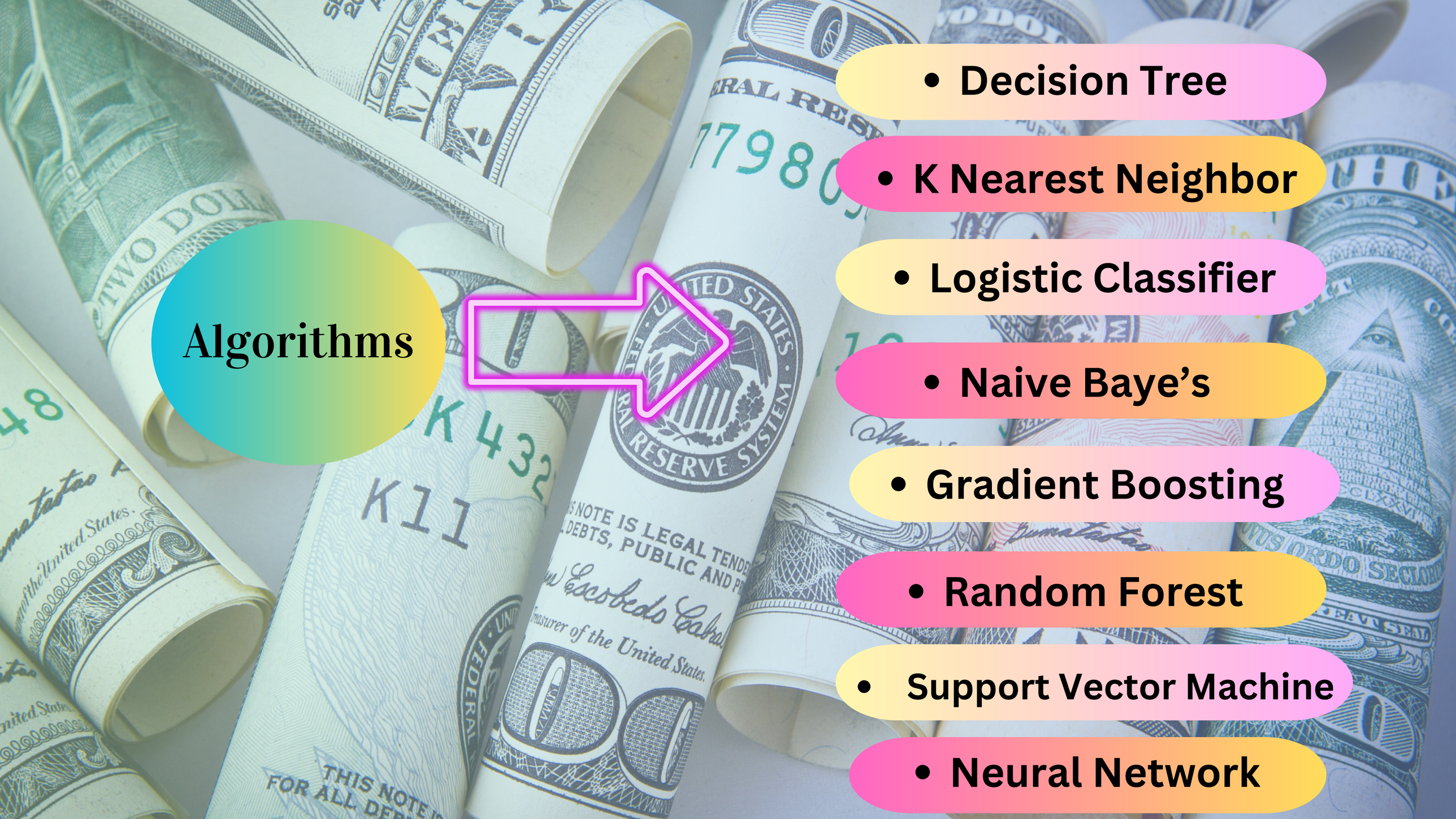
- In the Housing column we replaced yes and no categorical columns to numeric with assigning 0,1 to the columns

- In the month columns as there are many months included we changed the months to quarters and assigned the first four months as Q1, and the next four as Q2, sequentially Q3 and Q4.

Multi-collinearity :

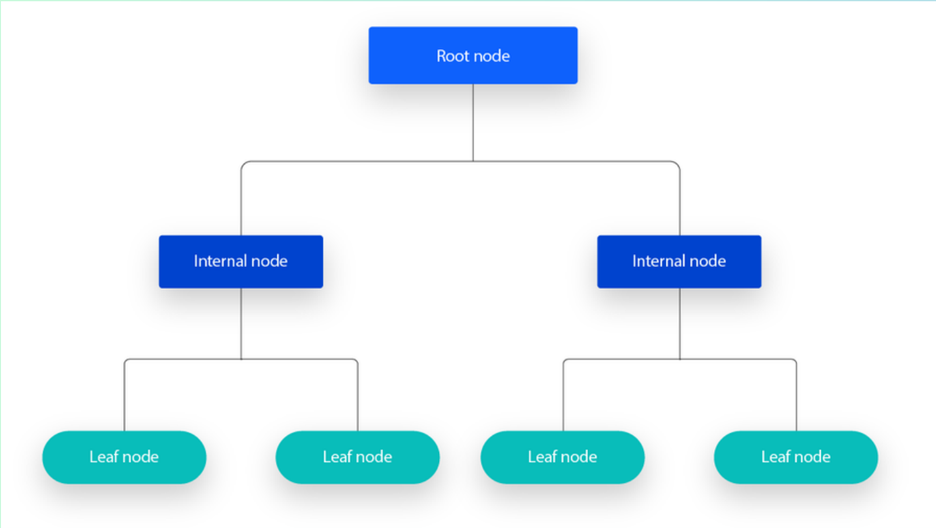
Multicollinearity occurs when predictor variables in a regression model are highly correlated, making it challenging to separate their individual effects on the target variable. It can lead to issues in interpreting coefficients, increased standard errors, and less reliable predictions, and vif is our **Variance Inflation Factor** which helps assess how much the variance of an estimated regression coefficient is inflated due to collinearity.

Variables	VIF
age	1.150454
balance	1.044702
day	1.067784
duration	1.013389
campaign	1.086890
pdays	4.524093
previous	1.891720
job	1.363747


Algorithms
• Decision Tree
• K Nearest Neighbor
• Logistic Classifier
• Naive Baye’s
• Gradient Boosting
• Random Forest
• Support Vector Machine
• Neural Network

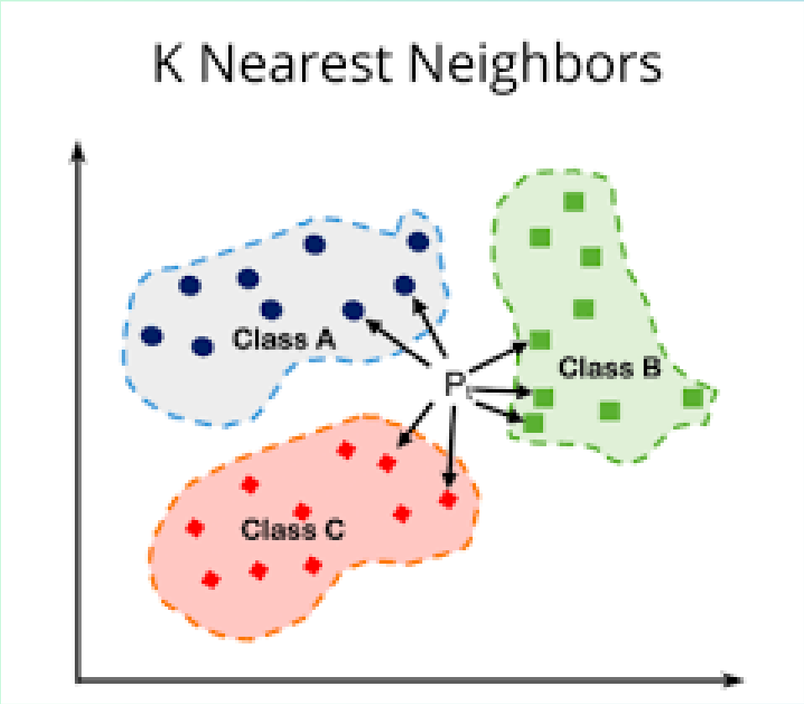
Algorithms :

1. Decision Tree classifier :



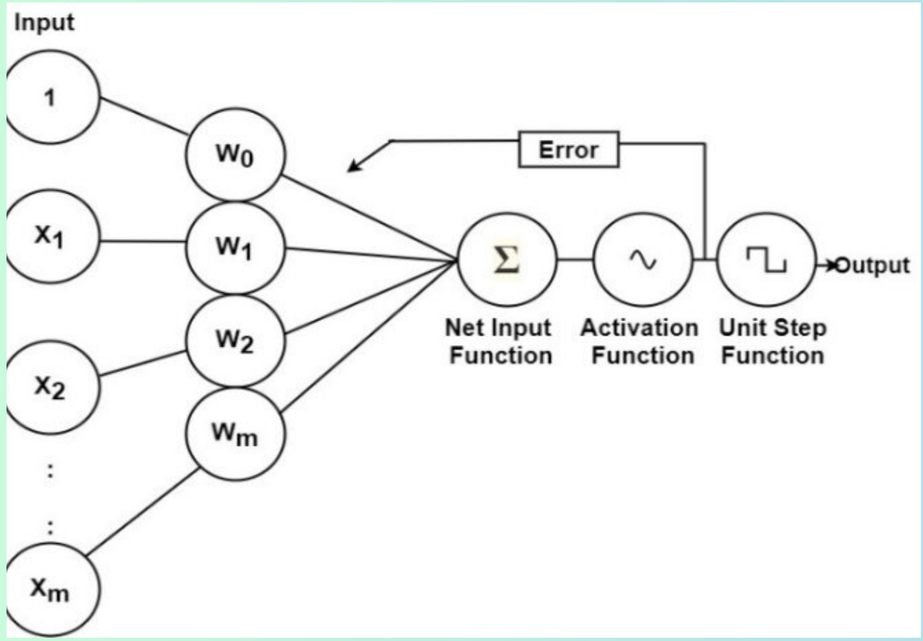
Train-Test ratio	Accuracy
80-20	86.78
70-30	**87.83**
75-35	85.88
65-35	85.33

2. K Nearest Neighbor classifier :



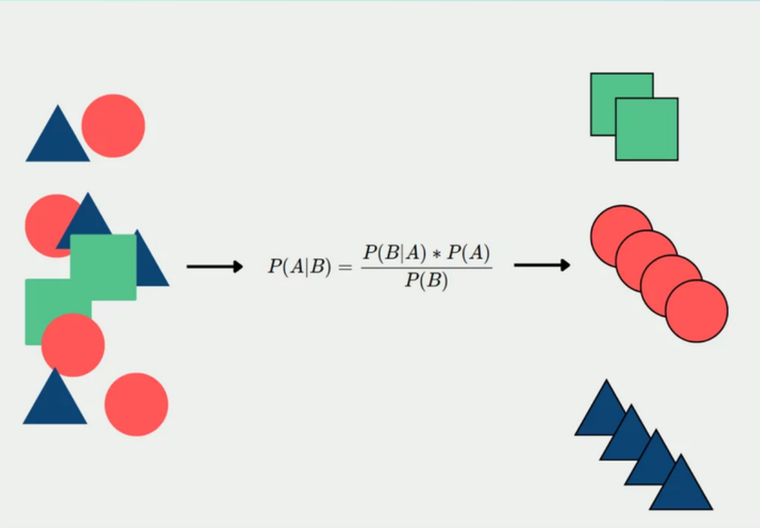
Train-Test ratio	Accuracy
80-20	**88.98**
70-30	87.88
75-35	87.56
65-35	86.74

3. Logistic classifier



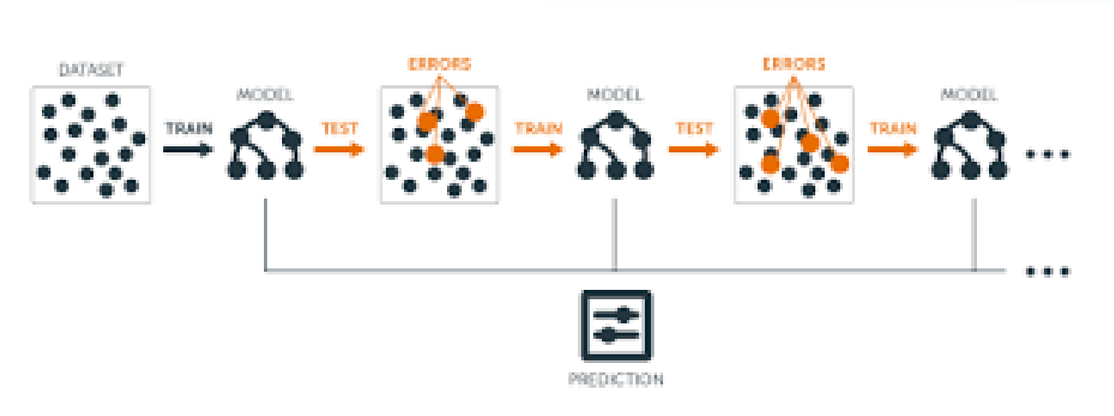
Train-Test ratio	Accuracy
80-20	**90.88**
70-30	90.15
75-35	89.90
65-35	89.30

4. Naive Baye's classifier



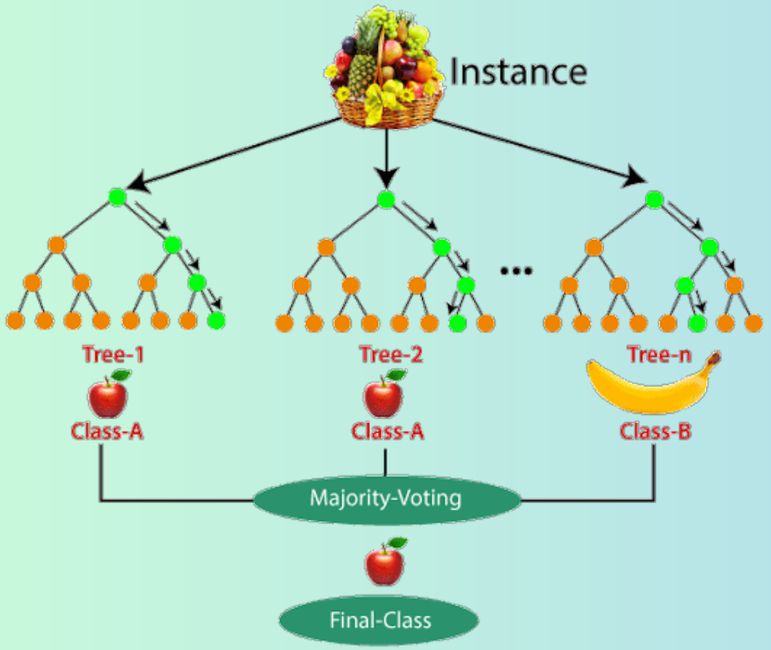
Train-Test ratio	Accuracy
80-20	**81.89**
70-30	81.24
75-35	80.67
65-35	81.32

5. Gradient Boosting classifier



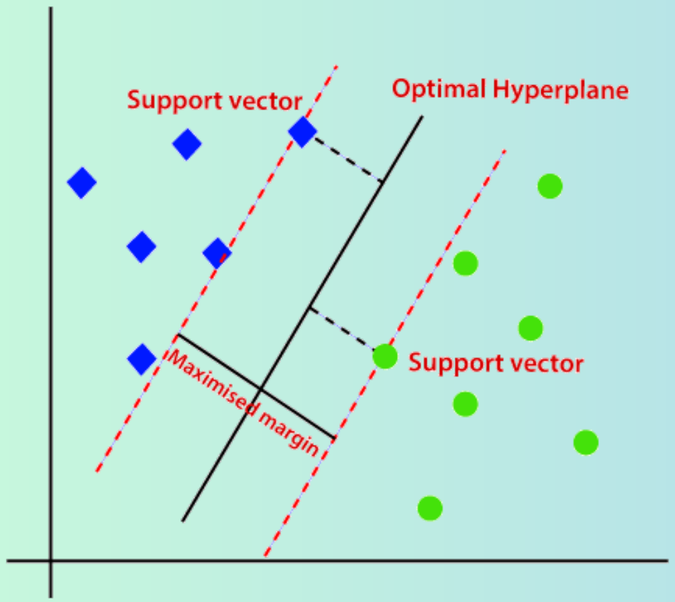
Train-Test ratio	Accuracy
80-20	**89.99**
70-30	89.76
75-35	88.95
65-35	81.32

6. Random Forest classifier



Train-Test ratio	Accuracy
80-20	**90.56**
70-30	89.78
75-35	87.64
65-35	86.97

7. Support Vector Machine classifier



Train-Test ratio	Accuracy
80-20	**89.54**
70-30	88.56
75-35	87.76
65-35	87.34

8. Neural Network

Train-Test ratio	Architectures	Optimizer	Epochs	Accuracy
80-20	64-32-1	Adam	100	**91.50**
70-30	67-30-1	Adam	120	89.17
75-35	65-21-3-4	Adamax	134	89.09
65-35	52-28-2-1	Adam	120	88.58

Comparision of algorithms :

Algorithms	Train-Test	Accuracy
Decision Tree	70-30	87.83
Naive Baye’s	80-20	88.98
Logistic classifier	80-20	90.88
Gradient Boosting	80-20	81.89
Random Forest	80-20	90.56
Support Vector Machine	80-20	89.54
Neural Network	80-20	**91.50**

Conclusion and Insights :

- For the Bank Marketing dataset, we performed eight types of machine learning algorithms: K Nearest Neighbour, Decision Tree, Naive Baye's, Logistic Regression, Gradient Boosting, Support Vector Machine, Random Forest, Neural Network.

- From all the analysis and implementations of the algorithm, we found the best results in "Neural Network" at 80-20 ratio that is 91.50.

- So, here we can conclude that Neural Network Algorithm can be considered as the best model for the given Bank Marketing dataset.

THANK YOU

Presented by :

K.Navya Poojitha

Priyanka

Manohar

Saswat

Appendix

Training and Testing

```
X = dummy.iloc[:,:-1]
y = dummy.iloc[:,-1]
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size = 0.3, random_state = 2)
X_train.shape , X_test.shape
```

Decision Tree classifier

```
from sklearn.model_selection import train_test_split#Iimporting the necessary libraries
from sklearn.tree import DecisionTreeClassifier
from sklearn import metrics
X = dummy.iloc[:,:-1]
y = dummy.iloc[:,-1]
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size = 0.2, random_state = 2)
X_train.shape , X_test.shape
dt = DecisionTreeClassifier()
dt1= dt.fit (X_train, y_train)#check for model fity_pred = dt.predict(X_test)
y_pred = dt.predict(X_test)
y_pred
print("accuracy",metrics.accuracy_score(y_test, y_pred))
```

```
accuracy 0.876243093922652
```

2. K Nearest Neighbour

```
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn import metrics
```

```
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size = 0.5, random_state = 2)
X_train.shape , X_test.shape
```

```
((2260, 26), (2261, 26))
```

```
model= KNeighborsClassifier(n_neighbors=5)
```

```
model.fit(X_train, y_train)#fitting k nearest neighbor from the classifier
KNeighborsClassifier()
```

```
▾ KNeighborsClassifier
KNeighborsClassifier()
```

```
y_pred= model.predict(X_test)
y_pred
```

```
array([0, 0, 0, ..., 0, 0, 0], dtype=uint8)
```

```
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.8748341441839894
```

Logistic Regression

```
from sklearn.model_selection import train_test_split #Importing the necessary libraries
from sklearn.linear_model import LogisticRegression
from sklearn import metrics
```

```
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size = 0.3, random_state = 2)
X_train.shape , X_test.shape
```

```
((3164, 26), (1357, 26))
```

```
clf = LogisticRegression(max_iter=5000)
```

```
clf.fit (X_train, y_train)
```

```
LogisticRegression
LogisticRegression(max_iter=5000)
```

```
y_pred = clf.predict(X_test)
y_pred
```

```
array([0, 0, 0, ..., 0, 0, 1], dtype=uint8)
```

```
print("accuracy",metrics.accuracy_score(y_test, y_pred))
```

```
accuracy 0.9034635224760501
```

Naive baye’s

```
from sklearn.model_selection import train_test_split #Importing the necessary libraries
from sklearn.naive_bayes import GaussianNB
from sklearn import metrics
```

```
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size = 0.2, random_state = 50)
X_train.shape , X_test.shape
```

```
((3616, 26), (905, 26))
```

```
nb = GaussianNB()
```

```
nb.fit (X_train, y_train)
```

```
▾ GaussianNB
GaussianNB()
```

```
y_pred = nb.predict(X_test)
y_pred
```

```
print("accuracy",metrics.accuracy_score(y_test, y_pred))
```

```
accuracy 0.8176795580110497
```

Gradient Boosting

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.metrics import accuracy_score

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.5, random_state=50)

# Initialize and train the Gradient Boosting Classifier
cls = GradientBoostingClassifier(n_estimators=100, random_state=42)
cls.fit(X_train, y_train)

# Make predictions on the test set
y_pred = cls.predict(X_test)

# Calculate accuracy
print("accuracy",metrics.accuracy_score(y_test, y_pred))
```

```
accuracy 0.8951791242812914
```

```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report
from sklearn.preprocessing import LabelEncoder
```

```python
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```python
rf_classifier = RandomForestClassifier(n_estimators=100, random_state=42)
rf_classifier.fit(X_train, y_train)
```

```
RandomForestClassifier
RandomForestClassifier(random_state=42)
```

```python
# Make predictions
y_pred = rf_classifier.predict(X_test)

# Calculate accuracy
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.2f}")
```

```
Accuracy: 0.90
```

Random Forest

Support Vector Machine

```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.5, random_state=50)

# Standardize features by removing the mean and scaling to unit variance
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
# Create an SVM model
svm_model = SVC(kernel='linear')

# Train the model
svm_model.fit(X_train_scaled, y_train)
# Make predictions
y_pred = svm_model.predict(X_test_scaled)

# Evaluate accuracy
print("accuracy",metrics.accuracy_score(y_test, y_pred))
```

```
accuracy 0.891640866873065
```

Neural Network

```python
import pandas as pd
import numpy as np
import tensorflow as tf
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

# Perform train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Scale the features
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Build the neural network model
model = tf.keras.Sequential([
    tf.keras.layers.Dense(64, activation='relu', input_shape=(X_train.shape[1],)),
    tf.keras.layers.Dense(32, activation='relu'),
    tf.keras.layers.Dense(1, activation='sigmoid')
])

# Compile the model
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

# Train the model
model.fit(X_train, y_train, epochs=10, batch_size=32, validation_data=(X_test, y_test))
```

```
Epoch 1/10
113/113 [==============================] - 4s 18ms/step - loss: 0.3801 - accuracy: 0.8648 - val_loss: 0.2780 - val_accuracy: 0.8928
Epoch 2/10
113/113 [==============================] - 1s 12ms/step - loss: 0.2640 - accuracy: 0.8960 - val_loss: 0.2549 - val_accuracy: 0.9006
Epoch 3/10
113/113 [==============================] - 1s 9ms/step - loss: 0.2472 - accuracy: 0.9013 - val_loss: 0.2477 - val_accuracy: 0.9017
Epoch 4/10
113/113 [==============================] - 1s 5ms/step - loss: 0.2388 - accuracy: 0.9040 - val_loss: 0.2464 - val_accuracy: 0.9006
Epoch 5/10
113/113 [==============================] - 1s 5ms/step - loss: 0.2327 - accuracy: 0.9046 - val_loss: 0.2398 - val_accuracy: 0.9006
Epoch 6/10
113/113 [==============================] - 0s 4ms/step - loss: 0.2256 - accuracy: 0.9074 - val_loss: 0.2414 - val_accuracy: 0.9017
Epoch 7/10
113/113 [==============================] - 1s 5ms/step - loss: 0.2209 - accuracy: 0.9087 - val_loss: 0.2408 - val_accuracy: 0.9006
Epoch 8/10
113/113 [==============================] - 1s 6ms/step - loss: 0.2169 - accuracy: 0.9090 - val_loss: 0.2365 - val_accuracy: 0.9039
Epoch 9/10
113/113 [==============================] - 1s 8ms/step - loss: 0.2125 - accuracy: 0.9121 - val_loss: 0.2344 - val_accuracy: 0.9006
Epoch 10/10
113/113 [==============================] - 1s 8ms/step - loss: 0.2079 - accuracy: 0.9134 - val_loss: 0.2336 - val_accuracy: 0.9050
<keras.src.callbacks.History at 0x78071c1d3580>
```

```python
# Evaluate the model
loss, accuracy = model.evaluate(X_test, y_test)
print(f"Accuracy: {accuracy*100:.2f}%")
```

```
29/29 [==============================] - 0s 7ms/step - loss: 0.2336 - accuracy: 0.9050
Accuracy: 90.50%
```

Classification report

K-Nearest Neighbors Classification Report:

	precision	recall	f1-score	support
0	0.91	0.97	0.94	1993
1	0.59	0.32	0.41	268
accuracy			0.89	2261
macro avg	0.75	0.64	0.68	2261
weighted avg	0.88	0.89	0.88	2261

Naive Bayes Classification Report:

	precision	recall	f1-score	support
0	0.92	0.87	0.90	1993
1	0.33	0.47	0.39	268
accuracy			0.82	2261
macro avg	0.63	0.67	0.64	2261
weighted avg	0.85	0.82	0.84	2261

random forest Classification Report:

	precision	recall	f1-score	support
0	0.98	0.99	0.99	1993
1	0.95	0.87	0.91	268
accuracy			0.98	2261
macro avg	0.97	0.93	0.95	2261
weighted avg	0.98	0.98	0.98	2261

neural network Classification Report:

	precision	recall	f1-score	support
0	0.92	0.97	0.95	1205
1	0.62	0.36	0.46	152
accuracy			0.90	1357
macro avg	0.77	0.67	0.70	1357
weighted avg	0.89	0.90	0.89	1357

Decision Tree Classification Report:

	precision	recall	f1-score	support
0	0.99	0.99	0.99	1993
1	0.93	0.95	0.94	268
accuracy			0.99	2261
macro avg	0.96	0.97	0.97	2261
weighted avg	0.99	0.99	0.99	2261

logistic regression Classification Report:

	precision	recall	f1-score	support
0	0.91	0.98	0.94	1993
1	0.68	0.29	0.40	268
accuracy			0.90	2261
macro avg	0.79	0.63	0.67	2261
weighted avg	0.88	0.90	0.88	2261

gradient boosting Classification Report:

	precision	recall	f1-score	support
0	0.92	0.97	0.94	1993
1	0.60	0.34	0.43	268
accuracy			0.90	2261
macro avg	0.76	0.65	0.69	2261
weighted avg	0.88	0.90	0.88	2261

svm Classification Report:

	precision	recall	f1-score	support
0	0.88	1.00	0.94	1993
1	0.00	0.00	0.00	268
accuracy			0.88	2261
macro avg	0.44	0.50	0.47	2261
weighted avg	0.78	0.88	0.83	2261

Months check if there's any seasonal pattern or trend regarding the months when subscriptions are higher. This could influence marketing strategies—like targeting specific months where subscriptions tend to peak.

Loan and Balance it helps to analyze if individuals with higher balances or without existing loans are more likely to subscribe. Financial stability might influence the decision to invest in term deposits.

Poutcome it helps to explore whether individuals who had a positive outcome in previous campaigns are more likely to subscribe. Positive experiences might increase the likelihood of subscription in the future.