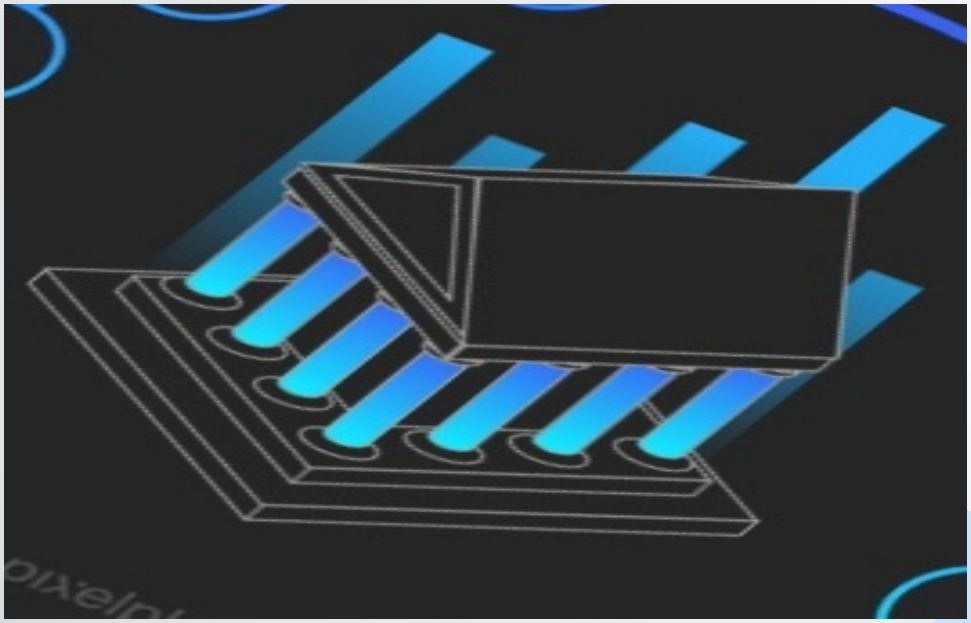
BANK MARKETING DATASET

CLASSIFICATION METHOD

TEAM MEMBERS
G.SAKETH
K.SATHWIKA
T.SANTHOSHI

Machine Learning Project Workflow



Data pre processing

Exploratory Data Analysis

Training The Model

Determining Model Accuracy

Comparing Results with Other Models

Conclusions and Insights

About the Data

- The dataset is from a bank's marketing campaign and includes details like age, job, marital status, education, contact type, and previous campaign outcomes. It contains 4,521 rows and 17 columns, with the target variable indicating if the client subscribed to a term deposit.

Objective

- The aim is to analyze client data and campaign history to predict whether a client will subscribe to a term deposit, helping the bank improve its targeting strategy.

The Path

- The process involved cleaning and preprocessing the data, converting categorical variables, handling class imbalance, and applying machine learning models such as Logistic Regression, Decision Tree, Random Forest, and SVM to find the most effective model.

Data and Data Quality

- Data Introduction
- The dataset pertains to a bank's direct marketing campaign, a aimed at promoting term deposits. It includes 4521 customer records with 17 features, and the goal is to predict whether a customer subscribed to the product (y). The data spans various customer details, contact campaign attributes, and previous campaign outcomes.

Variable OverView

AGE : Numerical customer's age.

BALANCE : Numerical Average yearly balance in the account

DAY : Numerical Last Contact day of the month

DURATION : Numerical Last contact duration in seconds

CAMPIGN : Numerical Number of contacts during this campaign

PDAYS : Numerical Days since last contact from previous campaign

PREVIOUS : Numerical Number of contacts before this campaign

1. JOB : Categorical Type of job (e.g., admin., technician, services)
2. MARITAL : Categorical Marital status (married, single, divorced)
3. EDUCATION: Categorical Education level (primary, secondary, tertiary, unknown)
4. DEFAULT : Categorical Whether the customer has credit in default
5. HOUSING : Categorical Whether the customer has a housing loan.
6. LOAN : Categorical Whether the customer has a personal loan.
7. CONTACT : Categorical Contact communication type (cellular, telephone, unknown).
8. MONTH : Categorical Last contact month of the year
9. POUTCOME : Categorical Outcome of the previous campaign (success, failure, etc.).
10. Y. : Binary Target Whether the client subscribed to the term deposit (yes/no).

Missing Values

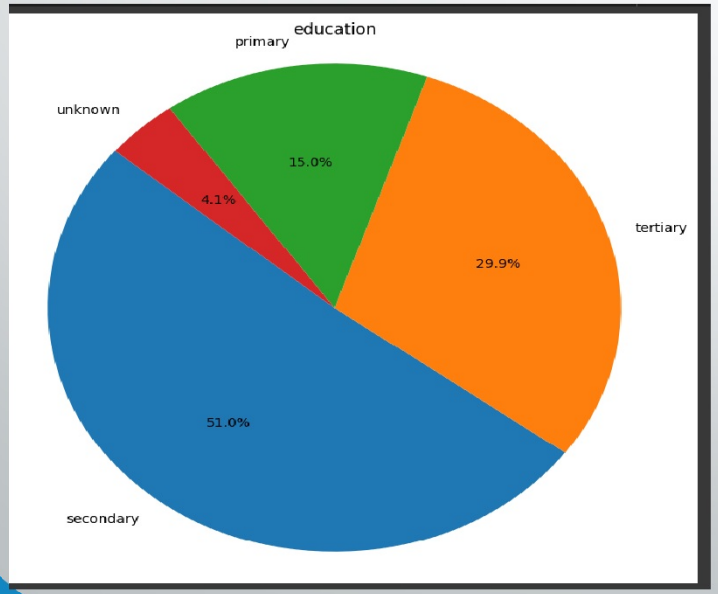
- There were no missing values in the dataset. All variables were completely filled, eliminating the need for imputation.

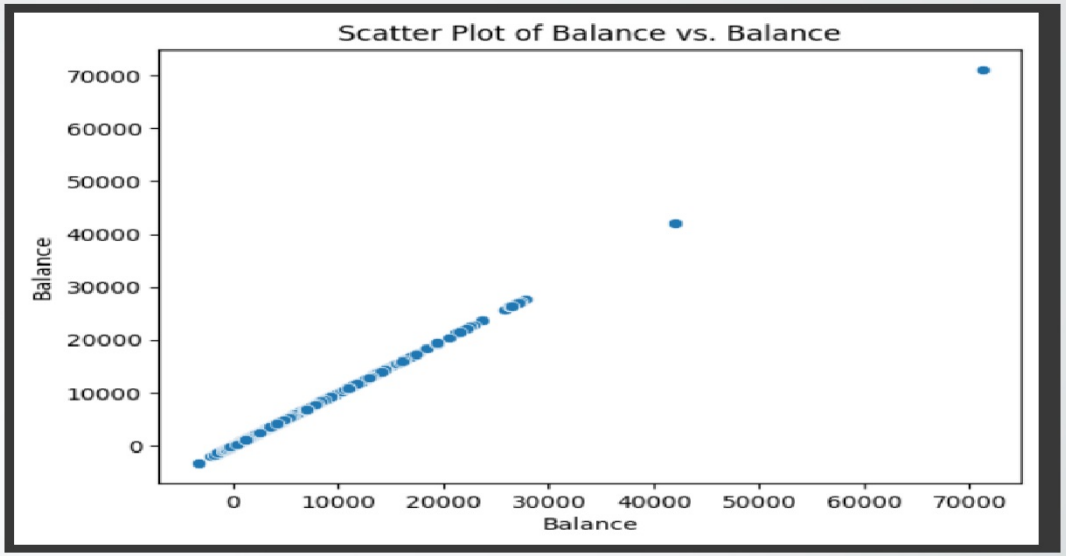
Dropped & Engineered Columns

The columns job were dropped due to missing data, but a few were removed after feature engineering for model efficiency.

Data Visualisation

- Pie Chart Shows education distribution with secondary 51.0% , tertiary 29.9% , primary 15.09% , unknown 4.1% in the pie chart

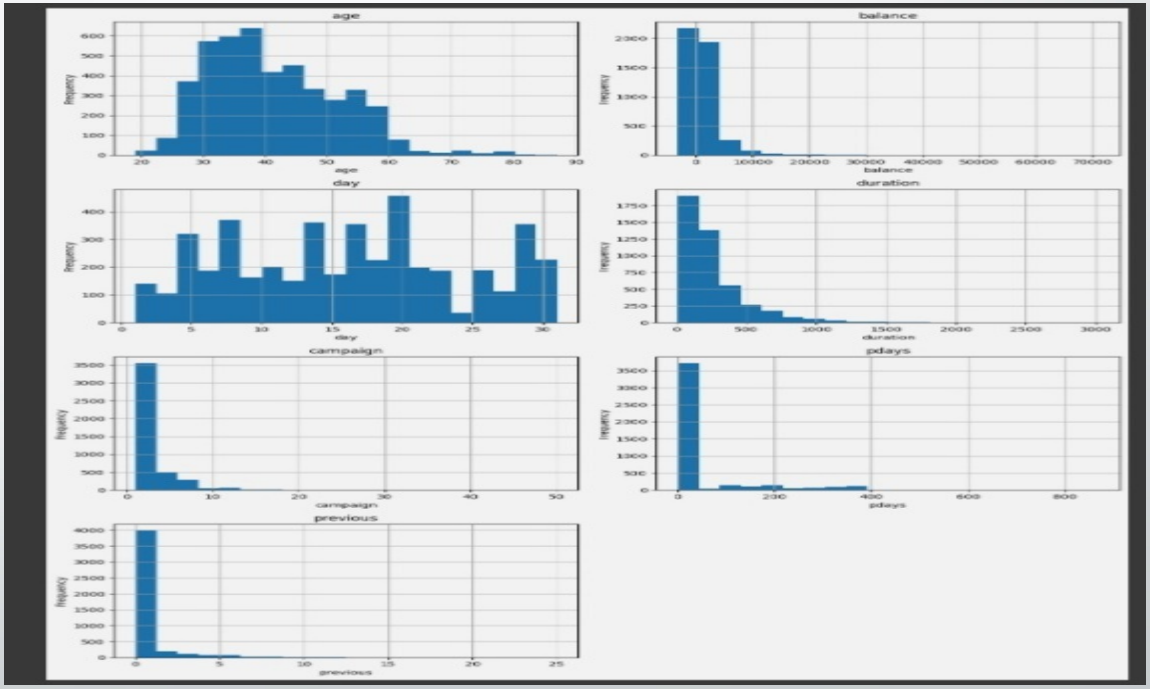




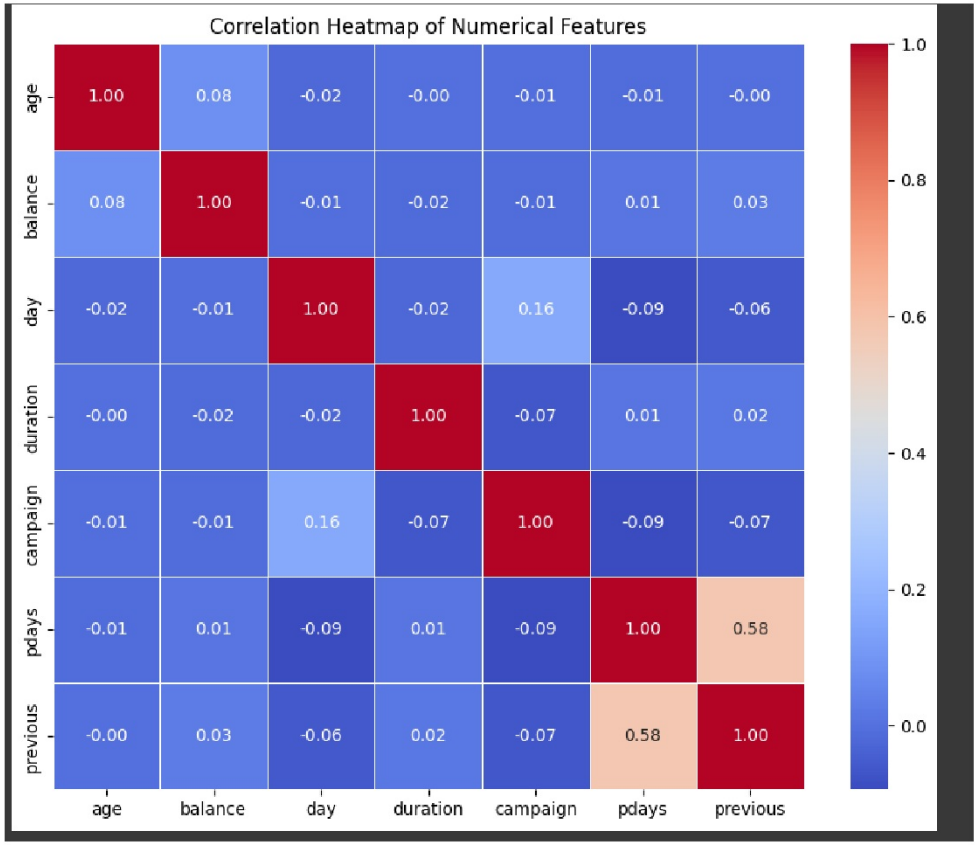
- The scatter plot displays in Balance vs Balance 0 to start in dot line 25000 , some dots are in 25000 to 30000 and two dots in 40000 to 70000 show in scatter plot Balance vs Balance in equal in scatter plot.

HISTOGRAM

- Displays frequency distribution and skewness of each feature.



HEAT MAP



PresentationMeasures and visualizes pairwiselinear correlation between features.

Algorithms

- Logistics classification

Train/Test spilt	Accuracy
80-20	62%
75-25	61.2%
70-30	62.3%

- In Conclusion, we get the best accuracy with 80:20 ratio for for this dataset.

- Logistics Classification shows a gradual increase in accuracy with more training data.

KNN Classification

Train/Test split	Accuarcy
80-20	70.1%
75-25	66%
70-30	67.8%
65-35	67.7%

- In Conclusion, we get the best accuracy with 80:20 ratio for this dataset.
- KNN model is sensitive to scaling, good performance with large training sizes.

SVM Classification

Train/Test split		Accuracy
80-20		62.5%
75-25		58.1%
70-30		62.3%
65-35		58.7%

- In Conclusion, we get the best accuracy with 80:20 ratio for this dataset.

- SVM performance steadily but with slightly lower accuracy than tree-based models.

Decision Tree Classification

Train/Test split		Accuracy
80-20		73%
75-25		71.1%
70-30		71%
65-35		70.3%

- In Conclusion, we get the best accuracy with 80:20 ratio for this dataset.

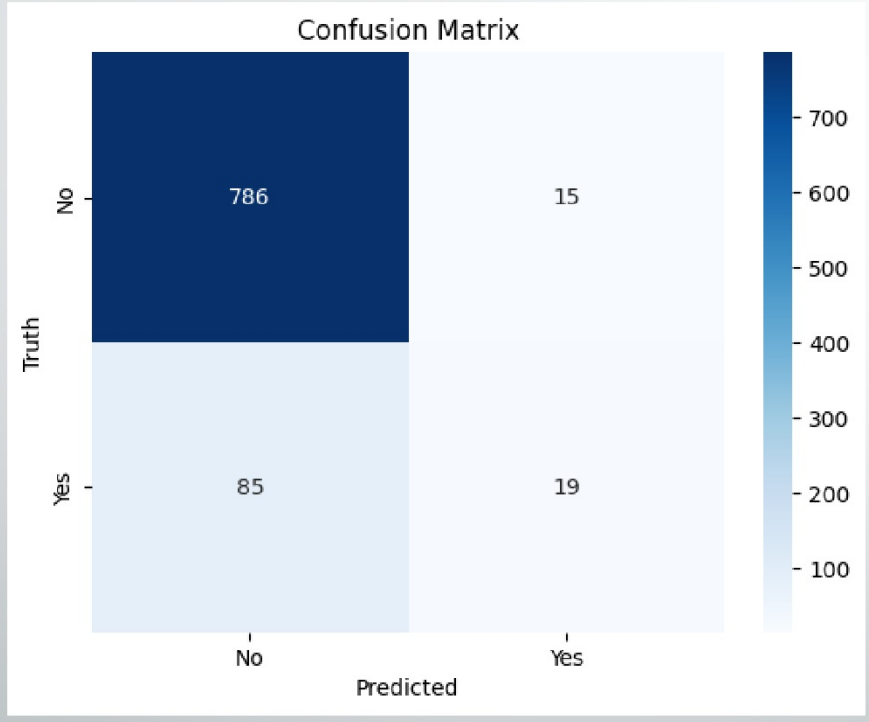
- Decision Tree shows high accuracy, especially with more training data.

Xg Boost

Train/Test split	Accuracy
80-20	72%
75-25	71%
70-30	68%
65-35	67%

- In Conclusion, we get the best accuracy with 80:20 ratio for this dataset.

Confusion Matrix



- It is used to summarize how well the model distinguishes zero vs onces predictions.

Classification Report

class	precision	recall	F1-score	support
0	0.71	0.68	0.70	506
1	0.06	0.63	0.63	394

- **It is used to evaluate how well the model performs across different prediction metrics.**

Model Comparison Table

Model68%	80-20	75-25	70-30	65-35
Logistic Regression	62.8%	61.2%	62.3%	62.3%
KNN	70.1%	66%	67.8%	67.7%
SVM	73%	71.3%	71%	70.3%
Decision Tree	62.5%	58.1%	62.3%	58.7%
XG Boost	72%	71%	68%	67%

Conclusion:-

- After application Of different ML algorithms to the dataset, the best accuracy is given by SVM.
- In Conclusion SVM is the best fit for the dataset.


Thank you!


APPENDIX

Training And Testing

```
from sklearn.model_selection import train_test_split
X = df_combined.drop('salary_label', axis=1)
y = df_combined['salary_label']
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.30,train_size=0.70)
```

Logistic Regression

```
from sklearn.linear_model import LogisticRegression
logreg = LogisticRegression(C=1e9)

logreg.fit(X_train, y_train)
predictions = logreg.predict(X_test)
print(predictions)
```

```
[ ]  from sklearn.metrics import accuracy_score
     accuracy_score(y_test,y_pred)
```

```
0.8530386740331491
```

KNN

```
[ ]  from sklearn.model_selection import train_test_split

[ ]  from sklearn.neighbors import KNeighborsClassifier

[ ]  model=KNeighborsClassifier(n_neighbors=25)

[ ]  model.fit(X_train, y_train)
```

```
KNeighborsClassifier
KNeighborsClassifier(n_neighbors=25)
```

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

SVM

```
from sklearn.svm import SVC
```

```
model = SVC(kernel='linear')
```

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

```
SVC(kernel='linear')
```

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

```
svm = pd.DataFrame({'Predicted':y_pred,'Actual':y_test})
svm
```

	Predicted	Actual
267	1	0
933	0	0
403	1	1
1006	1	1
1977	1	1
...	...	...

```
from sklearn.metrics import classification_report, confusion_matrix
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       741
           1       0.35      0.96      0.52        57
           2       0.00      0.00      0.00        37
           3       0.00      0.00      0.00        22
           4       0.00      0.00      0.00        14
           5       0.00      0.00      0.00         9
           6       0.00      0.00      0.00        10
           7       0.00      0.00      0.00         7
           8       0.00      0.00      0.00         4
           9       0.00      0.00      0.00         1
          10       0.00      0.00      0.00         1
          12       0.00      0.00      0.00         1
          17       0.00      0.00      0.00         1
          18       0.00      0.00      0.00         0
          25       0.00      0.00      0.00         0

    accuracy                           0.88       905
   macro avg       0.09      0.13      0.10       905
weighted avg       0.84      0.88      0.85       905
```

Decision Tree

```
[ ]  from sklearn.tree import DecisionTreeClassifier
     from sklearn.ensemble import RandomForestClassifier
     from sklearn.model_selection import train_test_split
     from sklearn.metrics import classification_report, confusion_matrix
     from sklearn.tree import plot_tree

     from matplotlib import pyplot as plt
```

```
     print(X_train.shape)
     print(y_train.shape)
     print(X_test.shape)
     print(y_test.shape)
```

```
     (3616, 25)
     (3616,)
     (905, 25)
     (905,)
```

```
[ ]  dt = DecisionTreeClassifier()
```

```
[ ]  dt.fit(X_train, y_train)
```

```
     DecisionTreeClassifier
     DecisionTreeClassifier()
```

```
     print(classification_report(y_test, y_pred))
     print(confusion_matrix(y_test, y_pred))
```

```
                   precision    recall  f1-score   support

                0       1.00      1.00      1.00       741
                1       0.37      0.30      0.33        57
                2       0.23      0.27      0.25        37
                3       0.19      0.23      0.21        22
                4       0.14      0.21      0.17        14
                5       0.00      0.00      0.00         9
                6       0.00      0.00      0.00        10
                7       0.00      0.00      0.00         7
                8       0.25      0.25      0.25         4
                9       0.00      0.00      0.00         1
               10       0.00      0.00      0.00         1
               11       0.00      0.00      0.00         0
               12       0.00      0.00      0.00         1
               15       0.00      0.00      0.00         0
               17       0.00      0.00      0.00         1
               19       0.00      0.00      0.00         0
               20       0.00      0.00      0.00         0

         accuracy                           0.86       905
        macro avg       0.13      0.13      0.13       905
     weighted avg       0.86      0.86      0.86       905

     [[741   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
      [  0  17  17   8   8   2   1   0   2   0   0   0   0   0   0   1   1]
      [  0  12  10   5   4   2   1   0   0   0   0   1   1   0   0   1   0]
```

Xg Boost

```
import numpy as np # linear algebra
import pandas as pd
import xgboost as xgb
from sklearn.model_selection import train_test_split
```

```
X = df.drop('previous', axis=1)
y = df['previous']
X = pd.get_dummies(X, drop_first=True)
```

```
#Split the dataset into train and Test
seed = 7
test_size = 0.3
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=test_size, random_state=seed)
```

```
# For evaluation, use the encoded test set.
print("Accuracy for model 3: %.2f" % (accuracy_score(y_test_encoded, pred3_encoded) * 100))

# If you need the original class labels for predictions, you would need to map pred4_encoded back
# using unique_classes_test or a combined mapping from the original y
```

```
Accuracy for model 3: 85.92
```

```
xgb2 = xgb.XGBClassifier(
 learning_rate =0.7,
 n_estimators=1000,
 max_depth=4,
 min_child_weight=6,
 gamma 0,
 subsample=0.8,
 colsample_bytree=0.8,
 objective= 'binary:logistic',
 nthread 4,
 scale_pos_weight=1,
 seed=27)
```

```
from sklearn.ensemble import RandomForestClassifier

rfc = RandomForestClassifier()
rfc_model = rfc.fit(X_train, y_train)
pred8 = rfc_model.predict(X_test)
print("Accuracy for Random Forest Model: %.2f" % (accuracy_score(y_test, pred8) * 100))
```

```
Accuracy for Random Forest Model: 87.25
```