BAN 5753 Mini Project – 2

Team 8 Care Catalysts

Ashutosh Maharana, A20394179

Omkar Narayan Patade, A20392690

Rishitha Koneru, A20397813

Satishwar Kedas, A20383206

Table of Contents

[Introduction 2](#_Toc152186618)

[Dataset 2](#_Toc152186619)

[Exploratory Data Analysis 2](#_Toc152186620)

[Target Variable Distribution 2](#_Toc152186621)

[Numerical Features 3](#_Toc152186622)

[Summary Statistics 3](#_Toc152186623)

[Distributions 4](#_Toc152186624)

[Correlations 5](#_Toc152186625)

[Predictive Modelling 6](#_Toc152186626)

[Modeling 6](#_Toc152186627)

[Model Performance Comparison 6](#_Toc152186628)

[Feature Importance 6](#_Toc152186629)

[Key Recommendations 7](#_Toc152186630)

[K-mean Clustering 7](#_Toc152186631)

[Appendix 8](#_Toc152186632)

[Exploratory Data Analysis Code 8](#_Toc152186633)

[Predictive Modeling Code 9](#_Toc152186634)

[K-Means Clustering Code 11](#_Toc152186635)

# Introduction

XYZ Bank conducted a direct marketing campaign through telephone calls to sell their term deposit product. Based on the data collected through the marketing campaign, XYZ Bank is interested in identifying and predicting individuals who are most likely to purchase their term deposit product.

As consultants, we have been asked to analyze the provided dataset and generate key recommendations that will enable XYZ bank to effectively target prospective customers based on their likelihood to purchase the term deposit.

# Dataset

The XYZ Bank Deposit Data Classification dataset, comprising 41,188 entries across 20 columns, captures details of the bank's direct marketing campaigns conducted via telephone between May 2008 and November 2010. It includes a blend of categorical and numerical attributes ranging from client demographics (age, job, marital status, education) and financial information (credit default, housing, and personal loans) to campaign-specific data (contact type, month, day of the week, duration, campaign contacts, and previous campaign outcomes). Key economic indicators such as employment variation rate, consumer price and confidence indices, Euribor 3-month rate, and the number of employees are also included.

# Exploratory Data Analysis

## Target Variable Distribution

The outcome variable is a binary variable indicating whether the individual signed up for a term deposit or not indicating by Yes or No. The distribution of the outcome variable indicates that there is significant class imbalance. Of all the 41,188 records, we observe that only 11.26% of the individuals signed up.

A black background with white text

Description automatically generated

## Numerical Features

### Summary Statistics

A screenshot of a computer screen

Description automatically generated

The dataset has 41,188 records and several numerical variables related to the banking campaign.

* The age variable, representing the age of the individuals in the dataset, has an average of 40.02 years, with a standard deviation of 10.42. The age range spans from a minimum of 17 years to a maximum of 98 years.
* The duration variable, representing the last contact duration in seconds, has a mean of 258.29 seconds, with a standard deviation of 259.28 seconds. The duration spans from a minimum of 0 seconds to a maximum of 4918 seconds.
* The campaign variable, indicating the number of contacts performed during the campaign and for a specific client, has an average of 2.57 contacts with a standard deviation of 2.77, ranging from a minimum of 1 contact to a maximum of 56 contacts.
* Pdays, representing the number of days that passed since the client was last contacted from a previous campaign, has an average of 962.48 days with The standard deviation of 186.91 days suggests varying durations of elapsed time, ranging from 0 days to 999 days.
* The variable "Previous," denoting the number of contacts performed before the current campaign and for a specific client, has an average of 0.17 contacts. The standard deviation of 0.49 indicates a relatively low average, with a range from 0 to 7 contacts.
* The employment variation rate, a quarterly indicator, has an average of 0.08. With a standard deviation of 1.57, this variable showcases the quarterly fluctuations in employment rates, ranging from a minimum of -3.4 to a maximum of 1.4.
* The consumer price index, a monthly indicator, has an average of 93.58. The standard deviation of 0.58 indicates slight monthly variations, with a range from 92.20 to 94.77.
* The consumer confidence index, another monthly indicator, has an average of -40.50. With a standard deviation of 4.63, it reveals monthly variations in consumer confidence levels, ranging from -50.8 to -26.9.
* The Euribor 3-month rate, a daily indicator, exhibits an average of 3.62. With a standard deviation of 1.73, this variable illustrates daily fluctuations in the Euribor rate, ranging from 0.634 to 5.045.
* The number of employees, a quarterly indicator, has an average of 5167.04. The standard deviation of 72.25 indicates moderate quarterly variations, with a range from 4963.6 to 5228.1.

### Distributions

Based on the distributions shown in the figure below, we have the following insights about a few variables.

*Right-Skewed Variables* (Duration, Campaign, Previous): These variables exhibit highly right-skewed distributions, concentrating values towards the lower end with a long right tail.

*Left-Skewed and Sparse Variables* (Euribor3m and Nr\_employed): Euribor3m and Nr\_employed show left-skewed distributions with sparse populations, concentrating on the higher end with a long left tail. The mean may be higher than the median, and the sparsity of values poses challenges in model stability and generalizability.

*Age* (Close to Normal Distribution): Age stands out with a distribution close to normal, offering a symmetric spread of values around the mean, meeting regression assumptions. This reduces sensitivity to outliers and enhances overall model performance.

A diagram of a distribution of features

Description automatically generated

### Correlations

A screen shot of a black screen

Description automatically generated

The following variables have correlation more than 0.7:

* **Emp\_var\_rate and Nr\_employed: - Correlation: 0.906970**

The employment variation rate (`emp\_var\_rate`) and the number of employees (`nr\_employed`) have a very strong positive correlation of 0.906970. The high positive correlation suggests a close relationship between these two economic indicators, implying that they move in a similar direction and are likely influenced by similar factors in the dataset.

* **Euribor3m and Emp\_var\_rate: - Correlation: 0.972245**

The Euribor 3-month rate (`euribor3m`) and the employment variation rate (`emp\_var\_rate`) have an exceptionally strong positive correlation of 0.972245. This implies a nearly perfect linear relationship between these two variables, signifying that they tend to move closely together.

* **Euribor3m and Nr\_employed: - Correlation: 0.945154**

The Euribor 3-month rate (`euribor3m`) and the number of employees (`nr\_employed`) exhibit a very strong positive correlation of 0.945154. It suggests that as the Euribor 3-month rate increases, the number of employed individuals also tends to increase significantly.

# Predictive Modelling

## Modeling

We utilized five predictive modeling approaches – Logistic Regression, Decision Tree Classifier, Random Forest Classifier, Support Vector Machines (Binary Classification), and Gradient Boosting Machine (XGBoost). We utilized a random train-test split of 60-40 to achieve a good representation of the dataset for both training and testing.

The categorical features were converted into indexed features using StringIndexer() and the numerical features were scaled to unit norm (only for Logistic Regression). For other classification approaches, we utilize the numerical features without any scaling. Since, we are only interested in the improved classification performance, we are not removing any features on the grounds of multi-collinearity. For instance, we observed earlier that Euribor 3-month rate is highly positively correlated with number of employed individuals in a quarter.

All the models were evaluated on the ROC-AUC score which is measures the area under the Receiving Operator Characteristic curve. Higher the AUC score, higher the performance of the model and the better its ability to predict the target variable.

## Model Performance Comparison

Based on the AUC score, we see that the *Gradient Boosting Machine based binary classifier* achieves the highest score and we identify the Gradient Boosting Machine based classifier as the best performing model for our prediction task.

|  |  |
| --- | --- |
| **Model** | **ROC-AUC Score** |
| Logistic Regression | 0.9279 |
| Decision Tree Classifier | 0.47 |
| Random Forest Classifier | 0.9234 |
| Support Vector Classifier | 0.9244 |
| Gradient Boosting Machine | 0.9382 |

## Feature Importance

Based on the feature importance values, we identified the top 10 important features in the best performing model. We note that duration is the most important feature contributing to the model predictions. Duration indicates the amount of time spent on the telephone call in the last contact in seconds and higher the duration, higher the likelihood of the individual converting into a sale. Similarly, we find the nr\_employed shows a high feature importance, indicating that number of employees as an economic indicator has a strong effect on the likelihood of people buying term deposit.

A graph with blue squares

Description automatically generated

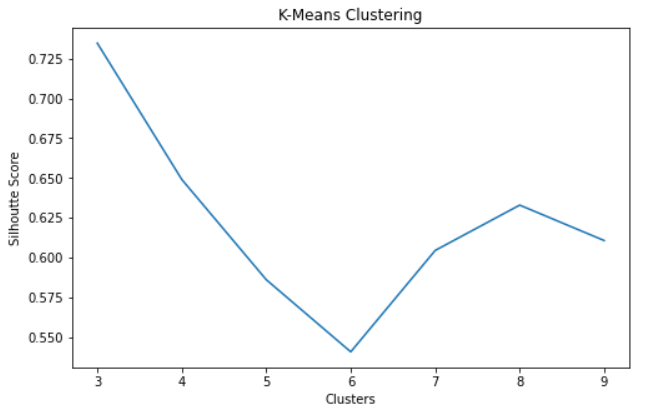
# Key Recommendations

Based on the results of the final model, it is evident that:

1. Higher the duration of the call, the higher the likelihood of the individuals converting into a sale. Which means that either the individuals are showing interest in the product and are asking more information or that the caller has been able to engage the individual. Making sure that the calls are engaging can lead to higher conversion rates
2. Market conditions or economic conditions can significantly affect the conversion rates. Euribor 3-month rate which is an indicator of economic conditions coupled with the number of employees in the last quarter both are significant predictors of the conversion rates. Higher Euribor 3-month rate can lead to higher conversion. So, it is recommended to conduct such marketing campaigns when the economic conditions are better.

# K-mean Clustering

K-Means is a clustering algorithm that aims to minimize the distance between datapoints and the cluster center. The algorithm stops when the data points and positions of the centroids stop changing. In K-Means it is necessary to run the algorithms with different initialized centers as in the first run these centers are chosen randomly, a solution with the lowest sum of distances is chosen.



For the implementation of K-Means the top 5 most important variables were selected to build clusters on namely ‘duration’, ’nr\_employed’, ’month\_index’, ’euribor3m’, ’job\_index’. The number of clusters were selected based on the silhouette scores for each number of clusters.

Silhouette Score for 3 clusters = 0.73

Silhouette Score for 4 clusters = 0.64

Silhouette Score for 5 clusters = 0.58

Silhouette Score for 6 clusters = 0.54

Silhouette Score for 7 clusters = 0.60

Silhouette Score for 8 clusters = 0.63

Silhouette Score for 9 clusters = 0.61

Based on this, 3 clusters were chosen as the optimal number of clusters for the K-means clustering algorithm.

# Appendix

## Initialization Code

spark=SparkSession.builder.master ("local[\*]").appName("MiniProject").getOrCreate()

sc=spark.sparkContext

sqlContext=SQLContext(sc)

df = spark.read.option('header','True').option("InferSchema",'True').csv('XYZ\_Bank\_Deposit\_Data\_Classification.csv', sep = ";")

df = df.withColumnRenamed("emp.var.rate","emp\_var\_rate").withColumnRenamed('cons.price.idx','cons\_price\_idx').withColumnRenamed("cons.conf.idx",'cons\_conf\_idx')

df = df.withColumnRenamed('nr.employed','nr\_employed')

numeric = []

categorical = []

for x,y in df.dtypes:

if y in ['int','double']:

numeric.append(x)

else:

categorical.append(x)

## Exploratory Data Analysis Code

numeric\_features = [t[0] for t in df.dtypes if t[1] == 'int' or t[1] == 'double']

print(df.select(numeric\_features).describe().toPandas().transpose())

print(df.select(numeric\_features).toPandas().corr())

df.groupby("y").count().show()

#Variable distributions

fig = plt.figure(figsize=(25,15)) ## Plot Size

st = fig.suptitle("Distribution of Features", fontsize=50,

verticalalignment='center') # Plot Main Title

for col,num in zip(df.toPandas().describe().columns, range(1,11)):

ax = fig.add\_subplot(3,4,num)

ax.hist(df.toPandas()[col])

# plt.style.use('dark\_background')

plt.grid(False)

plt.xticks(rotation=45,fontsize=20)

plt.yticks(fontsize=15)

plt.title(col.upper(),fontsize=20)

plt.tight\_layout()

st.set\_y(0.95)

fig.subplots\_adjust(top=0.85,hspace = 0.4)

plt.show()

## Predictive Modeling Code

#Logistic Regression

scaler = StandardScaler(inputCol= 'features\_s', outputCol="scaled\_features", withStd=True, withMean=True)

assembler2 = VectorAssembler().setInputCols(numeric).setOutputCol("features\_s")

VA = VectorAssembler(inputCols=df\_x.columns, outputCol='features')

log\_reg = LogisticRegression(featuresCol='features', labelCol='y\_index')

# Creating the pipeline

pipex = Pipeline(stages=[assembler2, scaler, VA, log\_reg])

train\_data, test\_data = df\_converted.randomSplit([0.6, 0.4])

# Fitting the model on training data

fit\_model = pipex.fit(train\_data)

fit\_model.write().overwrite().save('logreg')

results = fit\_model.transform(test\_data)

res = BinaryClassificationEvaluator(labelCol="y\_index", rawPredictionCol="rawPrediction", metricName="areaUnderROC")

ROC\_AUC = res.evaluate(results)

print('ROC-AUC for logistic Regression:',ROC\_AUC)

res = BinaryClassificationEvaluator(labelCol="y\_index", rawPredictionCol="rawPrediction", metricName="areaUnderPR")

PR = res.evaluate(results)

print('Precision-Recall for logistic Regression:',PR)

#Random Forest depth 5

rf\_classifier = RandomForestClassifier(featuresCol="features", labelCol="y\_index", numTrees=100, maxDepth=5)

pipe2 = Pipeline(stages=[VA, rf\_classifier])

fit\_model = pipe2.fit(train\_data)

fit\_model.write().overwrite().save('RF1')

results2 = fit\_model.transform(test\_data)

res2 = BinaryClassificationEvaluator(labelCol="y\_index", rawPredictionCol="rawPrediction", metricName="areaUnderROC")

ROC\_AUC = res2.evaluate(results2)

print('ROC-AUC for Random Forest:',ROC\_AUC)

res = BinaryClassificationEvaluator(labelCol="y\_index", rawPredictionCol="rawPrediction", metricName="areaUnderPR")

PR = res.evaluate(results2)

print('Precision-Recall for Random Forest:',PR)

#Gradient Boosted Tree

GBT\_classifier = GBTClassifier(featuresCol="features", labelCol="y\_index", maxDepth=5, maxIter=25, stepSize=0.01)

pipe3 = Pipeline(stages=[VA, GBT\_classifier])

fit\_model = pipe3.fit(train\_data)

fit\_model.write().overwrite().save('GBT-Classifier')

results3 = fit\_model.transform(test\_data)

res3 = BinaryClassificationEvaluator(labelCol="y\_index", rawPredictionCol="rawPrediction", metricName="areaUnderROC")

ROC\_AUC = res3.evaluate(results3)

print("ROC-AUC for Gradient Boosted Tree is:",ROC\_AUC)

res = BinaryClassificationEvaluator(labelCol="y\_index", rawPredictionCol="rawPrediction", metricName="areaUnderPR")

PR = res.evaluate(results3)

print('Precision-Recall for Gradient Boosted Tree is:',PR)

#Support Vector Classifier

svc = LinearSVC(featuresCol="features", labelCol="y\_index", maxIter=10, regParam=0.1)

pipe4 = Pipeline(stages=[VA, svc])

fit\_model = pipe4.fit(train\_data)

fit\_model.write().overwrite().save('Linear-SVC')

results4 = fit\_model.transform(test\_data)

res4 = BinaryClassificationEvaluator(labelCol="y\_index", rawPredictionCol="rawPrediction", metricName="areaUnderROC")

ROC\_AUC = res4.evaluate(results4)

print("ROC-AUC for Support Vector Classifier is:", ROC\_AUC)

res = BinaryClassificationEvaluator(labelCol="y\_index", rawPredictionCol="rawPrediction", metricName="areaUnderPR")

PR = res.evaluate(results4)

print('Precision-Recall for Support Vector Classifier is:',PR)

#Decision Tree Classifier

dt\_classifier = DecisionTreeClassifier(featuresCol="features", labelCol="y\_index", maxDepth=6)

pipe5 = Pipeline(stages=[VA, dt\_classifier])

fit\_model = pipe5.fit(train\_data)

fit\_model.write().overwrite().save('Decision\_Tree')

results5 = fit\_model.transform(test\_data)

res5 = BinaryClassificationEvaluator(labelCol="y\_index", rawPredictionCol="rawPrediction", metricName="areaUnderROC")

ROC\_AUC = res5.evaluate(results5)

print('ROC-AUC for Decision Tree is:',ROC\_AUC)

res = BinaryClassificationEvaluator(labelCol="y\_index", rawPredictionCol="rawPrediction", metricName="areaUnderPR")

PR = res.evaluate(results5)

print('Precision-Recall for Decision Tree is:',PR)

## K-Means Clustering Code

#K-Means

df\_k = df\_x.select('duration','nr\_employed','month\_index','euribor3m','job\_index')

assemble = VectorAssembler(inputCols=df\_k.columns, outputCol='features')

sil\_score = []

inertial = []

for x in range(3,10):

k = x

kmeans = KMeans(k=k, seed=1234)

kpipeline = Pipeline(stages=[assemble, kmeans])

model = kpipeline.fit(df\_k)

centers = model.stages[-1].clusterCenters()

predictions = model.transform(df\_k)

evaluator = ClusteringEvaluator()

silhouette = evaluator.evaluate(predictions)

print("Silhouette Score for {0} = {1}".format(k,silhouette))

sil\_score.append(silhouette)