C:\Users\janani\Desktop\RGU.PNGA close up of a sign

Description automatically generated

**BSc (Hons) Artificial Intelligence and Data Science**

**Module: CM2601 Machine Learning**

**Report**

**Module Leader: Mr. Sahan**

**RGU Student ID : 2330923**

**IIT Student ID : 20230976**

**Student Name : Abdullah Nazly**

Acknowledgment

I want to extend my sincere appreciation to all those who contributed and provided steadfast support throughout this coursework. A special acknowledgment goes to Mr.   
Sahan Priyanaya, the module leader, for her guidance that led to the successful completion of this coursework.

Additionally, I express extra gratitude to Mr. Pushpika Prasad Liyanaarachchi assistant lecturer, for assisting throughout the tutorials and addressing doubts, which facilitated the smooth execution of the coursework. I genuinely appreciate these individuals for their assistance in the successful completion of the coursework.

Executive Summary

This course work aims to build two machine learning models using python. Implementation is focused on understanding the key concepts of data preprocessing and training a model. A bank dataset is used to train the model here. Two models were used. One is a neural network model, and the other is a Random Forest model. For both models, the same way of data preprocessing is done. Though both got good predicting accuracy.

Contents

[1. Introduction 5](#_Toc185626032)

[2. Neural Network Model 5](#_Toc185626033)

[2.1. Data Preprocessing 5](#_Toc185626034)

[2.2. Checking NULL values and duplicates. 6](#_Toc185626035)

[2.3. Data format analysis. 7](#_Toc185626036)

[2.4. Getting the values of the features 8](#_Toc185626037)

[2.5. Checking the Unique values for each column 9](#_Toc185626038)

[2.6. Analyzing feature contact 9](#_Toc185626039)

[2.7. Analyzing poutcome feature 10](#_Toc185626040)

[2.8. Encoding Y variable and analyzing numerical columns. 10](#_Toc185626041)

[2.9. Removing unnecessary columns. 11](#_Toc185626042)

[2.10. Plotting on boxplot to identify extreme values and outliers 12](#_Toc185626043)

[2.10.1. Plotting duration column. 12](#_Toc185626044)

[2.10.2. Plotting age column. 12](#_Toc185626045)

[2.10.3. Plotting balance column. 13](#_Toc185626046)

[2.10.4. Plotting previous column. 13](#_Toc185626047)

[2.11. Applying Label encoding. 14](#_Toc185626048)

# Introduction

The Machine Learning course work is processed on simple approach. Starting with data preprocessing and going up to training the model. First the data is processed accordingly to train the neural network model. First the null values checked, the duplicates checked and problems were solved. Then unique values were checked to make decisions on encoding the categorical variables. Outliers and extreme values were checked for numerical values and no problems were found. Did EDA (Exploitary Data Analysis) for feature selection. For some features we must consider are influencing the target variable or not. No standardizations were made to test whether this processed data is enough to train the model. As the test the model showed a great performance to both models. The conclusion made to keep the data as it is without further standardization and improvements. Finally the models went through a hyper parameter tuning and evaluated.

# Neural Network Model

## Data Preprocessing

Bank detail dataset is used here to train the model. As always the dataset has to be preprocessed. The process starts with importing the library to manipulate dataset which is pandas. This dataset is a little different in csv format. Usually, it will be comma separated file. But in this dataset values are separated by semicolons.

"age";"job";"marital";"education";"default";"balance";"housing";"loan";**"contact";**"day";"month";"duration";"campaign";"pdays";"previous";"poutcome";"y"

58;"management";"married";"tertiary";"no";2143;"yes";"no";**"unknown";**5;"may";261;1;-1;0;"unknown";"no"

44;"technician";"single";"secondary";"no";29;"yes";"no";**"unknown";**5;"may";151;1;-1;0;"unknown";"no"

33;"entrepreneur";"married";"secondary";"no";2;"yes";"yes";**"unknown";**5;"may";76;1;-1;0;"unknown";"no"

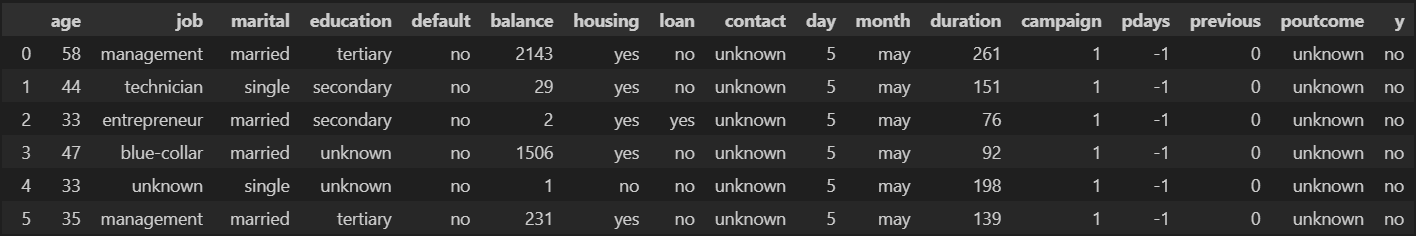
47;"blue-collar";"married";"unknown";"no";1506;"yes";"no";**"unknown";**5;"may";92;1;-1;0;"unknown";"no"

So as a result. When we try to read it with normal csv format, it won’t be able to read properly. Therefor another parameter is passed to divide the semicolons and read the csv file properly

df = pd.read\_csv('./Dataset/bank-full.csv', delimiter=';')

df.head(20)

Then we get the head output as usual.



If we didn’t use the delimiter parameter, the output would be like this.

A screenshot of a computer

Description automatically generated

## Checking NULL values and duplicates.

The next step is to check whether there’s any null values or duplicate values in the dataset. This is an important step to be considered before training the model. Checking the null values is performed using the below code which it gets the null values for every column in the dataset.

print(df.isnull().sum())

The output I got gave me the result that it doesn’t have any null value.

A black screen with white text

Description automatically generated

Next identifying the duplicate value is performed in the code. In this code also. The duplication is checked for all the columns at once by using this code.

print(df.duplicated().sum())

The result was zero which means there’s no duplicate values either.As a result, the part of data preprocessing becomes easier. We don’t have to handle missing values or duplicate values. We can directly go for the other areas of data preprocessing.

## Data format analysis.

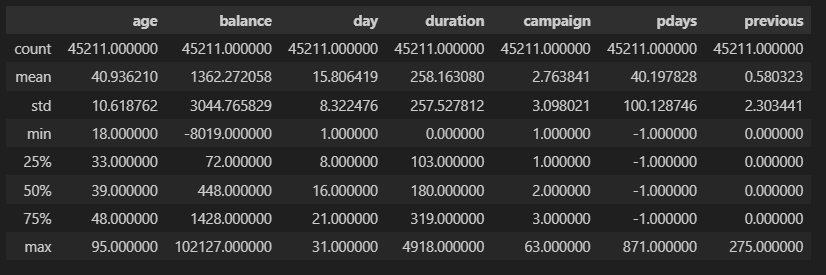
The usual format of checking for format is performed here, with .info() code the Dtypes including null counts are displayed.

df.info()



Then we get a statistic of the features to analyze the mean values, min-max values and other values as well.

df.describe()



By this we can understand that They are acceptable values. Theres no extreme values in these numerical features.

## Getting the values of the features

First the code below is run to get all the numerical features which doesn’t need data transformation currently.

numeric\_columns = df.select\_dtypes(include=['int64', 'float64']).columns

print("Numeric Columns:", numeric\_columns)

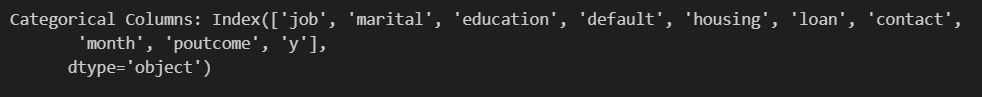


and the output given out put had all the numerical values columns. Next to get all the categorical valued features, the code below is run.

categorical\_columns = df.select\_dtypes(include=['object']).columns

print("Categorical Columns:", categorical\_columns)

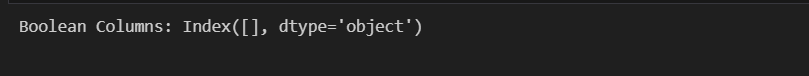
The output gave all the columns which has categorical values.



Finally, checking whether if any Boolean valued features are available using the below code  
  
boolean\_columns = df.select\_dtypes(include=['bool']).columns

print("Boolean Columns:", boolean\_columns)

but there wasn’t any Boolean valued features available.



## Checking the Unique values for each column

The unique values of each column are printed to make decision on encoding. Only categorical values should be considered under encoding, so categorical valued features are chosen here.

df['job'].unique()

df['marital'].unique()

df['education'].unique()

df['default'].unique()

df['housing'].unique()

df['contact'].unique()

df['month'].unique()

df['poutcome'].unique()

df['y'].unique()

df['campaign'].unique()

Codes were ran multiple times to identify the unique values of each columns to take the decision for encoding.

## Analyzing feature contact

When we consider the feature contact it seems like y variable doesn’t actually influenced by it. To make sure some EDA is done to the feature. We used chi-square test to evaluate whether there is a statistical relationship between two categorical variables. It will help to identify whether the feature is potentially important for the predictive model. This is done by the codes below.

from scipy.stats import chi2\_contingency

# If p value is < 0.05, the feature is influencing the targeted variable

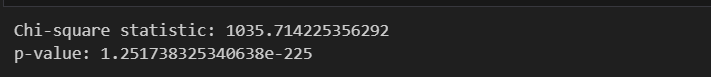
crosstab = pd.crosstab(df['contact'], df['y'])

chi2, p, dof, expected = chi2\_contingency(crosstab)

print("Chi-square statistic:", chi2)

print("p-value:", p)

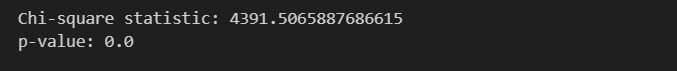
scipy library is imported to do statistical tests for the categorical variables. This is the output that was given to the above code.



As we can see, the p-value is extremely low and chi-square value is a bit higher. Therefore, the contact feature makes a huge influence in the prediction model.

## Analyzing poutcome feature

The same way of chi-square test is used to evaluate the relationship statistics between target variable and poutcome. This resulted in that this feature also deeply influences the target variable as the p-value is exactly equal to zero and chi-square value has a higher value.



## Encoding Y variable and analyzing numerical columns.

In this step we evaluate the relationship between numerical features and the binary target variable y using Point-Biserial Correlation. By this we can determine the significance of association between these features. First Label encoding is done for the y variable. Then numerical columns are selected. Then the point-biserial correlation is performed for each feature. Then the result is printed. These are the codes which used to calculate the correlation.

from scipy.stats import pointbiserialr

# Step 1: Convert 'y' to numeric (binary)

df['y'] = df['y'].map({'no': 0, 'yes': 1})

# Step 2: Define numerical features

numerical\_features = ['age', 'balance', 'day', 'duration', 'campaign', 'pdays', 'previous']

# Step 3: Calculate Point-Biserial Correlation for each feature

correlation\_results = []

for col in numerical\_features:

    corr, p\_value = pointbiserialr(df[col], df['y'])

    correlation\_results.append((col, corr, p\_value))

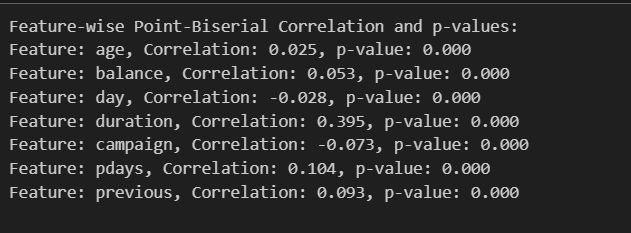
# Step 4: Print results

print("Feature-wise Point-Biserial Correlation and p-values:")

for feature, corr, p\_value in correlation\_results:

    print(f"Feature: {feature}, Correlation: {corr:.3f}, p-value: {p\_value:.3f}")

The output we got is below.



As we can see, the day column has a weak negative correlation. So, it might be less meaningful to the model. Age column also has a less correlation value, even though when considering real life scenarios, it may influence prediction. Campaign column also has a weak negative correlation. Considering the dimensionality of the dataset it proceeded to be removed.

## Removing unnecessary columns.

So according to analysis, the month column also removed. Because both day and month have similar characteristics. As the day has less influence, the month is also considered to be removed. Columns were dropped using this code

df = df.drop(columns=['day', 'month', 'campaign'])

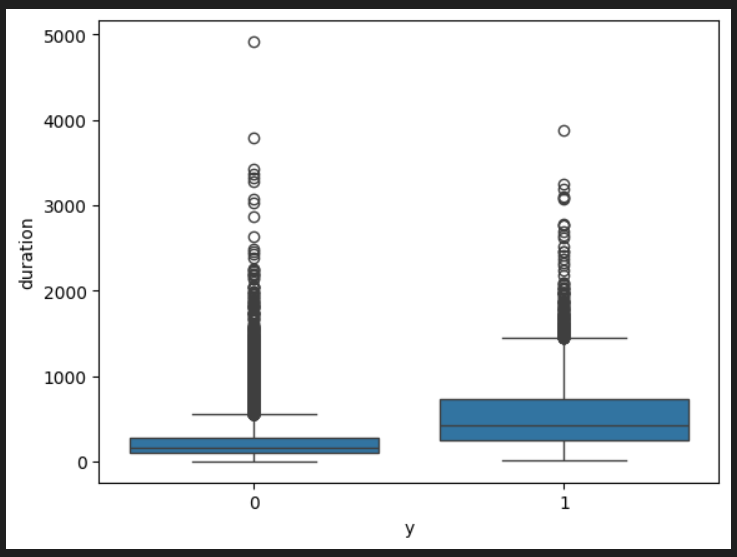
## Plotting on boxplot to identify extreme values and outliers

Box plotting is performed between numerical values and target values to analyse the extreme values or incorrect values. For every column similar format of this code is performed for plotting

sns.boxplot(x='y', y='duration', data=df)

### 2.10.1. Plotting duration column.

The result we got is looked like this.



As we can see for the duration theres no negative values which it can be incorrect format. So, this feature is good to go.

### 2.10.2. Plotting age column.

A diagram of a graph

Description automatically generated with medium confidence

As we can see there are some extreme values in the above plotting. Though these values are acceptable. These can happen in real life scenario as well. So, this column also good to go.

### 2.10.3. Plotting balance column.

A graph of a graph with numbers and lines

Description automatically generated with medium confidence

As we can see the plotting, we can see that there are some extreme values. But those can rarely happen. There are negative value as well. These could give a false assumption that there cannot be negative values for balance. But we can assume that the client could be under credit rather than having balance. Therefore, no need to do cleanup or transformation here.

### Plotting previous column.

A graph of a graph with numbers

Description automatically generated with medium confidence

As we can see the image there is one particular data point which unusually extreme. It’s better to remove that record. It is performed by the code below.

# Identify the record with the extreme value in 'previous'

outlier\_row = df[df['previous'] > 250]

# Display the details of the record

print("Outlier row details:")

print(outlier\_row)

# Drop the specific row

df = df.drop(outlier\_row.index)

# Verify the row is removed

print(f"Updated dataset shape: {df.shape}")

## Applying Label encoding.

Categorical features like default, housing and loan have only yes and no as their values which is binary datatype. So doing label encoding will reduce the dimension of the dataset. Also, it won’t be a huge effect for model prediction. Here we didn’t use any libraries. This can be achieved easily as they only have yes and no. The code below is used to encode the columns.

# List of columns to apply Label Encoding to (yes/no columns)

yes\_no\_columns = ['default', 'housing', 'loan']

# Apply Label Encoding to each of the columns in the list

df[yes\_no\_columns] = df[yes\_no\_columns].replace({'yes': 1, 'no': 0})

df.head()