Higher National Diploma in Data Science.

Machine Learning.

Module leader: Eng. Chameera De Silva.

COHNDDS23.1F

30th May 2023

Riley Douglas - COHNDDS23.1F -001

Acknowledgment

First and foremost, I would like to show my appreciation towards my beloved parents for motivating me and pushing me through.

Second of all I would like to express my special gratitude to my module leader and mentor Mr. Chameera De Silva for his tremendous support, dedication, passion, and amazing teaching skills and for guiding me and my colleagues through this subject. Your patience and consistency gave me and my classmates strength and motivation to keep going and pushing ourselves. Thank you for going above and beyond to make this module entertaining and enjoyable.

I would also like to thank our course director Mr. Balakumar for guiding me with advice, feedbacks, and tremendous support throughout the machine learning module. I am also grateful for my classmates and my beloved parents for the tremendous support throughout this journey.

Table of contents

Executive summary

This project aims to develop a decision tree classifier and logistic regression model in order to accurately determine whether a client will subscribe to term deposit based on their demographical features and previous banking history. the project below has gone through data preprocessing, data analysis, selecting the models, training the models, testing the model based on the test data and will be finally evaluated using confusion matrix to determine and further understand the model's predictions and outcomes.

The dataset used for this particular project is a historical dataset which has 45211 rows and 17 columns and consists of various customer demographic factors and previous banking data history such as age, education level, marital status, job status of the customers, outcomes of previous marketing campaigns, whether or not customers will subscribe to the term deposit and number of contacts with the bank and many more variables that ca help us build a model to identify binary outcomes

Step by step code explanation

Chapter 01 Data – preprocessing

This report will be showcasing two machine learning models using python to predict whether a customer will subscribe to a term deposit based on their demographic features such as age, job, marital status, education, and previous banking history such as number of contacts with the bank, outcome of previous marketing campaigns, etc.

Given below are the step-by-step code explanations,

Importing necessary libraries

```
#importing the libraries
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report
import matplotlib.pyplot as plt
import seaborn as sns
```

NumPy, pandas, seaborn and matplotlib we'll be used to analyze and further study the dataset. The first step would be to import all the necessary libraries and files to conduct the analysis.

Importing the dataset

```
#importing dataset
Data = pd.read_csv("/Users/rileydouglas/Downloads/bank-full.csv")
Data
```

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign	pdays	previous	poutcome	Target
0	58	management	married	tertiary	no	2143	yes	no	unknown	5	may	261	1	-1	0	unknown	no
1	44	technician	single	secondary	no	29	yes	no	unknown	5	may	151	1	-1	0	unknown	no
2	33	entrepreneur	married	secondary	no	2	yes	yes	unknown	5	may	76	1	-1	0	unknown	no
3	47	blue-collar	married	unknown	no	1506	yes	no	unknown	5	may	92	1	-1	0	unknown	no
4	33	unknown	single	unknown	no	1	no	no	unknown	5	may	198	1	-1	0	unknown	no
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
45206	51	technician	married	tertiary	no	825	no	no	cellular	17	nov	977	3	-1	0	unknown	yes
45207	71	retired	divorced	primary	no	1729	no	no	cellular	17	nov	456	2	-1	0	unknown	yes
45208	72	retired	married	secondary	no	5715	no	no	cellular	17	nov	1127	5	184	3	success	yes
45209	57	blue-collar	married	secondary	no	668	no	no	telephone	17	nov	508	4	-1	0	unknown	no
45210	37	entrepreneur	married	secondary	no	2971	no	no	cellular	17	nov	361	2	188	11	other	no

45211 rows × 17 columns

The second step is to import the csv file by using 'pd.read_csv' function and pasting the path name of the data source.

Exploring the dataset

```
Data.head()
```

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign	pdays	previous	poutcome	Target
0	58	management	married	tertiary	no	2143	yes	no	unknown	5	may	261	1	-1	0	unknown	no
1	44	technician	single	secondary	no	29	yes	no	unknown	5	may	151	1	-1	0	unknown	no
2	33	entrepreneur	married	secondary	no	2	yes	yes	unknown	5	may	76	1	-1	0	unknown	no
3	47	blue-collar	married	unknown	no	1506	yes	no	unknown	5	may	92	1	-1	0	unknown	no
4	33	unknown	single	unknown	no	1	no	no	unknown	5	may	198	1	-1	0	unknown	no

```
Data.shape
```

```
(45211, 17)
```

During this step of the analysis process **Data.head()** is a method used to display the first five rows of a data frame in Python. By calling **Data.head()**, you can inspect the top portion of your dataset to get a glimpse of its structure and the values it contains.

Data.shape depicts the shape of the dataset and shows the number of rows and columns that are present in the dataset.

Descriptive statistical analysis

```
#discriptive statistics of variables
Data.describe()
```

	age	balance	day	duration	campaign	pdays	previous
count	45211.000000	45211.000000	45211.000000	45211.000000	45211.000000	45211.000000	45211.000000
mean	40.936210	1362.272058	15.806419	258.163080	2.763841	40.197828	0.580323
std	10.618762	3044.765829	8.322476	257.527812	3.098021	100.128746	2.303441
min	18.000000	-8019.000000	1.000000	0.000000	1.000000	-1.000000	0.000000
25%	33.000000	72.000000	8.000000	103.000000	1.000000	-1.000000	0.000000
50%	39.000000	448.000000	16.000000	180.000000	2.000000	-1.000000	0.000000
75%	48.000000	1428.000000	21.000000	319.000000	3.000000	-1.000000	0.000000
max	95.000000	102127.000000	31.000000	4918.000000	63.000000	871.000000	275.000000

This step is important to identify the statistical description of the numerical variables in the dataset by using the function **Data.describe().**

accordingly, the statistics for the age column indicates that the mean age is approximately 40.94, with the standard deviation of 10.62 and the minimum and maximum ages are 18 and 95 respectively.

The mean balance is approximately 1362.27, with a standard deviation of 3044.77. The minimum balance is -8019, and the maximum balance is 102,127. The mean day is 15.81, with a standard deviation of 8.32. The minimum day is 1, and the maximum day is 31.

The mean duration is approximately 258.16, with a standard deviation of 257.53. The minimum duration is 0, and the maximum duration is 4918.

The mean campaign is approximately 2.76, with a standard deviation of 3.10. The minimum campaign is 1, and the maximum campaign is 63.

The mean pdays is 40.20, with a standard deviation of 100.13. The minimum pdays is -1, and the maximum pdays is 871.

The mean previous is approximately 0.58, with a standard deviation of 2.30. The minimum previous is 0, and the maximum previous is 275. Henceforth this statistical summary provides an insight of the numerical variables in the dataset.

Identifying data types of the variable

```
#Identifying the data types of the variable
Data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 45211 entries, 0 to 45210
Data columns (total 17 columns):
 #   Column     Non-Null Count  Dtype 
---  ------     --------------  ----- 
 0   age        45211 non-null  int64 
 1   job        45211 non-null  object
 2   marital    45211 non-null  object
 3   education  45211 non-null  object
 4   default    45211 non-null  object
 5   balance    45211 non-null  int64 
 6   housing    45211 non-null  object
 7   loan       45211 non-null  object
 8   contact    45211 non-null  object
 9   day        45211 non-null  int64 
 10  month      45211 non-null  object
 11  duration   45211 non-null  int64 
 12  campaign   45211 non-null  int64 
 13  pdays      45211 non-null  int64 
 14  previous   45211 non-null  int64 
 15  poutcome   45211 non-null  object
 16  Target     45211 non-null  object
dtypes: int64(7), object(10)
memory usage: 5.9+ MB
```

As shown on the figure above we will be identifying the data types of each variable to get a further understanding about the dataset that will be used in the prediction.

Checking for null and unique values

```
#Checking for any null values in the dataset
Data.isnull().sum()
```

```
age          0
job          0
marital      0
education    0
default      0
balance      0
housing      0
loan         0
contact      0
day          0
month        0
duration     0
campaign     0
pdays        0
previous     0
poutcome     0
Target       0
dtype: int64
```

Accordingly, we can proceed with the analysis process by checking for null values in the dataset, in this scenario there isn't any null values. In order to find the null values, we can use the function 'isnull ()' as shown above.

```
#identifying the unique values in the columns
Data.nunique()
```

```
age            77
job            12
marital         3
education       4
default         2
balance      7168
housing         2
loan            2
contact         3
day            31
month          12
duration     1573
campaign       48
pdays         559
previous       41
poutcome        4
Target          2
dtype: int64
```

Since now there aren't any null values as shown above, we can proceed by identifying the number of unique values in the chosen dataset. In order to do that we can use the function 'nunique **()'** this will print out the number of unique values in each column as shown above.

```python
#checking unique values per column
a = Data['age'].unique()
print(a)

b = Data['job'].unique()
print(b)

c = Data['marital'].unique()
print(c)

d = Data['education'].unique()
print(d)

e = Data['housing'].unique()
print(e)

f = Data['loan'].unique()
print(f)

g = Data['contact'].unique()
print(g)
```

```
[58 44 33 47 35 28 42 43 41 29 53 57 51 45 60 56 32 25 40 39 52 46 36 49
 59 37 50 54 55 48 24 38 31 30 27 34 23 26 61 22 21 20 66 62 83 75 67 70
 65 68 64 69 72 71 19 76 85 63 90 82 73 74 78 80 94 79 77 86 95 81 18 89
 84 87 92 93 88]
['management' 'technician' 'entrepreneur' 'blue-collar' 'unknown'
 'retired' 'admin.' 'services' 'self-employed' 'unemployed' 'housemaid'
 'student']
['married' 'single' 'divorced']
['tertiary' 'secondary' 'unknown' 'primary']
['yes' 'no']
['no' 'yes']
['unknown' 'cellular' 'telephone']
```

You can get a greater understanding of the data quality, identify potential problems with the data, and make informed choices on preprocessing activities like imputation, controlling values that are missing, encoding categorical variables, and choosing features by examining null values and unique values in the dataset.

Determining categorical features of the dataset

```
: termining the categorical features of the dataset
 egorical_features = [feature for feature in Data.columns if ((Data[feature].dtypes=='O') & (feature not in ['deposit' ])
 egorical_features

: ['job',
  'marital',
  'education',
  'default',
  'housing',
  'loan',
  'contact',
  'month',
  'poutcome',
  'Target']
```

The variable categorical_features are being populated with the column names of the categorical features in the DataFrame. Categorical features are typically non-numeric variables that represent discrete categories or labels.

The code 'Data[feature].dtypes=='O'' checks if the data type of each column in data is object a string or categorical data type. The condition feature not in ['deposit'] is used to exclude specific columns from the list of categorical features. by iterating over the columns in data and applying these conditions, the above code generates a list of column names that satisfy both criteria which is non-numeric data type and not in the exclusion list. These column names represent the categorical features in the dataset which is being used.

Furthermore, we can use the categorical features variable in upcoming steps to analyze the data set by visualizing it and thoroughly studying it.

Visualizing the data

In this step we will be visualizing the categorical data to further study and analyze the variables that are present in the dataset, by conducting a visualization we can identify any trends and patterns in the various demographical features and gather further information about the variables that we need to use in order to predict the customer subscription.

```
#understanding the categorical variables by visualizing them using plots
plt.figure (figsize=(20,80), facecolor='white')
plotnumber =1
for categorical_feature in categorical_features:
    ax = plt.subplot(12,3,plotnumber)
    sns.countplot(y=categorical_feature,data=Data)
    plt.xlabel(categorical_feature)
    plt.title(categorical_feature)
    plotnumber+=1
plt.show()
```

The above code is intended to visualize the categorical variables in the data set using countplots. We can visualize the distribution of each categorical variable in the data set. It helps in understanding the frequency and proportions of different categories within each variable, which can provide insights into the data distribution and potential relationships between variables.


job
management
technician
entrepreneur
blue-collar
unknown
retired
admin.
services
self-employed
unemployed
housemaid
student
0 2000 4000 6000 8000 10000
marital
married
single
divorced
0 5000 10000 15000 20000 25000
education
tertiary
secondary
unknown
primary
0 5000 10000 15000 20000
default
no
yes
0 10000 20000 30000 40000
housing
yes
no
0 5000 10000 15000 20000 25000
loan
no
yes
0 5000 10000 15000 20000 25000 30000 35000
contact
unknown
cellular
telephone
0 5000 10000 15000 20000 25000 30000
month
may
jun
jul
aug
oct
nov
dec
jan
feb
mar
apr
sep
0 2000 4000 6000 8000 10000 12000 14000
poutcome
unknown
failure
other
success
0 5000 10000 15000 20000 25000 30000 35000
Target
no
yes
0 5000 10000 15000 20000 25000 30000 35000 40000

Accordingly, we can visualize the categorical data in such countplots, we can gather useful information about the data set, such as the relationship between the various demographic features that affect the customers to subscribe to the term deposit.

Converting categorical variables into numerical representations

```python
from sklearn.preprocessing import LabelEncoder

# Create an instance of LabelEncoder
le = LabelEncoder()

# Perform label encoding on the "Category" column
Data["job_new"] = le.fit_transform(Data["job"])

Data["marital_new"] = le.fit_transform(Data["marital"])

Data["education_new"] = le.fit_transform(Data["education"])

Data["Target_new"] = le.fit_transform(Data["Target"])
```

```python
Data.head()
```

	age	job	marital	education	default	balance	housing	loan	contact	day	...	duration	campaign	pdays	previous	poutcome	Target	job_new
0	58	management	married	tertiary	no	2143	yes	no	unknown	5	...	261	1	-1	0	unknown	no	4
1	44	technician	single	secondary	no	29	yes	no	unknown	5	...	151	1	-1	0	unknown	no	9
2	33	entrepreneur	married	secondary	no	2	yes	yes	unknown	5	...	76	1	-1	0	unknown	no	2
3	47	blue-collar	married	unknown	no	1506	yes	no	unknown	5	...	92	1	-1	0	unknown	no	1
4	33	unknown	single	unknown	no	1	no	no	unknown	5	...	198	1	-1	0	unknown	no	11

During this step we will be converting the categorical variables in order to further complete the machine learning models. To achieve this, we will be using Label encoder, label encoding is a technique to convert categorical variables into numerical representations.

Above we have used “LabelEncoder ()” to encode the "job", "marital", "education", and "Target" columns and store the encoded values in new columns with "_new" appended to the column names.

After converting the categorical feature to numerical representations, we can use “Data.head ()” to represent the first five rows of the updated data set.

Correlation and heat map

In this step we would be identifying the correlation of the updated columns in order to further understand the variables and the relationships among them.

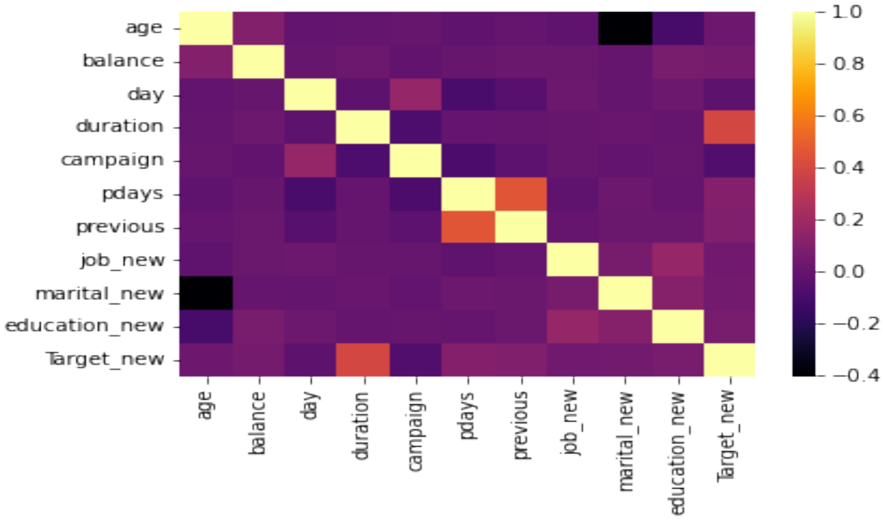
```
correlation = Data.corr()
correlation
```

	age	balance	day	duration	campaign	pdays	previous	job_new	marital_new	education_new	Target_new
age	1.000000	0.097783	-0.009120	-0.004648	0.004760	-0.023758	0.001288	-0.021868	-0.403240	-0.106807	0.025155
balance	0.097783	1.000000	0.004503	0.021560	-0.014578	0.003435	0.016674	0.018232	0.002122	0.064514	0.052838
day	-0.009120	0.004503	1.000000	-0.030206	0.162490	-0.093044	-0.051710	0.022856	-0.005261	0.022671	-0.028348
duration	-0.004648	0.021560	-0.030206	1.000000	-0.084570	-0.001565	0.001203	0.004744	0.011852	0.001935	0.394521
campaign	0.004760	-0.014578	0.162490	-0.084570	1.000000	-0.088628	-0.032855	0.006839	-0.008994	0.006255	-0.073172
pdays	-0.023758	0.003435	-0.093044	-0.001565	-0.088628	1.000000	0.454820	-0.024455	0.019172	0.000052	0.103621
previous	0.001288	0.016674	-0.051710	0.001203	-0.032855	0.454820	1.000000	-0.000911	0.014973	0.017570	0.093236
job_new	-0.021868	0.018232	0.022856	0.004744	0.006839	-0.024455	-0.000911	1.000000	0.062045	0.166707	0.040438
marital_new	-0.403240	0.002122	-0.005261	0.011852	-0.008994	0.019172	0.014973	0.062045	1.000000	0.108576	0.045588
education_new	-0.106807	0.064514	0.022671	0.001935	0.006255	0.000052	0.017570	0.166707	0.108576	1.000000	0.066241
Target_new	0.025155	0.052838	-0.028348	0.394521	-0.073172	0.103621	0.093236	0.040438	0.045588	0.066241	1.000000

The above image shows the calculated correlation matrix for the numerical columns in the data frame. The correlation coefficients between -1 and 1, where value of -1 indicates a perfect negative correlation, 0 indicates no correlation, and 1 indicates a perfect positive correlation. The above matrix helps us understand the relationships between different variables and can provide insights into the strengths and their linear relationships.

```
sns.heatmap(correlation,xticklabels=correlation.columns,yticklabels=correlation.columns,annot=False, cmap='inferno')
```

To further understand the linear relationships between the variables we have used a correlation heatmap.



An essential part of data preparation is creating a correlation heatmap, which can be used for feature selection as well as for identifying dependencies and visualizing the correlation coefficient between numerical variables in the data frame. In the above heatmap we can see that there are no strong positive between any variables, but we can see that there is fairly significant positive relationship between duration and target_new, pdays and previous variables. We can also see that there is a strong negative relationship between marital status and age variables. (Szabo, 2020)

Chapter 03 - Model selection

Splitting the data

Splitting the data into raining and testing is an essential step in machine learning and data analysis. It is important because by splitting the data we can assess the performance of the model we use on unseen data. Splitting the data also enables us to fine tune the hyperparameters of the model, hyperparameters are configuration settings that that are not learned from the data but are set by the user before training the model. (Galarnyk, 2022)

Henceforth by splitting the data into test and training sets is crucial to assess model performance, tuning hyperparameters and estimating the model's performance on unseen data.

```python
# split the data into features (X) and target variable (y)
X = Data[['age','duration','job_new','education_new', 'marital_new','campaign','pdays','previous']]
y = Data['Target_new']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=40)
```

We have taken the columns 'age', 'duration', 'job_new', 'education_new' 'pdays', 'previous', 'campaign' and 'marital_status' as the independent variable which is 'x' and target as the 'y' or dependent variable. As seen above the parameter is set to 0.2 which means that 20% of the data will be used for testing and the random state parameter is set to 40 to ensure reproducibility of the split.

Logistic regression

A statistical model known as logistic regression is used to predict the probability of a binary event based on one or more independent factors. It's a classification process that's commonly applied to binary classification issues when the dependent variable has two classes, like "yes" or "no". (Swaminathan, 2018)

```
# Create an instance of the logistic regression model
model = LogisticRegression()

# Train the model on the training data
model.fit(X_train, y_train)

# Make predictions on the testing data
y_pred = model.predict(X_test)

# Evaluate the model's performance
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.90      0.98      0.94      7976
           1       0.56      0.18      0.27      1067

    accuracy                           0.89      9043
   macro avg       0.73      0.58      0.60      9043
weighted avg       0.86      0.89      0.86      9043
```

The above classification report provides the summary of the model's performance on the testing data. Out of all incidents that were predicted to be positive, it shows the percentage of positive instances (class 1) that were accurately predicted. For class 1, the accuracy of 0.56 indicates that only 56% of the predicted cases are true positives. The model's total accuracy in predicting both classes. With an accuracy of 0.89, the model accurately predicts the class for 89% of the test set cases.

```
#Calculating the accuracy score
from sklearn.metrics import accuracy_score

accuracy_score(y_test,y_pred)
print("Training Accuracy", model.score(X_train, y_train))
print("Accuracy:", accuracy)
```

```
Training Accuracy 0.8877184251271842
Accuracy: 0.8847727524051753
```

In this we would be representing the training accuracy score of the logistic regression model, based on the model the training accuracy f the logistic regression model is 0.8877 which is 88.77%nd the accuracy of the testing data is 0.8848 which is 88.48%

The model's performance on the training data is shown by the training accuracy, which demonstrates how well the model corresponds its training data. It indicates that the logistic regression model in this instance had a training set accuracy of 88.77%.

The model's performance on the testing data is likely comparable to its performance on the training data given that both the training accuracy and testing accuracy are fairly comparable.

```
#testing model accuracy
print("Model accuracy",model.score(X_test, y_test))
```

```
Model accuracy 0.8866526595156474
```

During this step we would be testing the accuracy of the model, hence the model accuracy is 0.8867 which is 88.67%. According to the given testing data (X_test and y_test), the logistic regression model has an accuracy of 88.67%, as shown by this. The percentage of accurately predicted instances (both positive and negative) out of all the examples in the testing set is represented by the accuracy score.

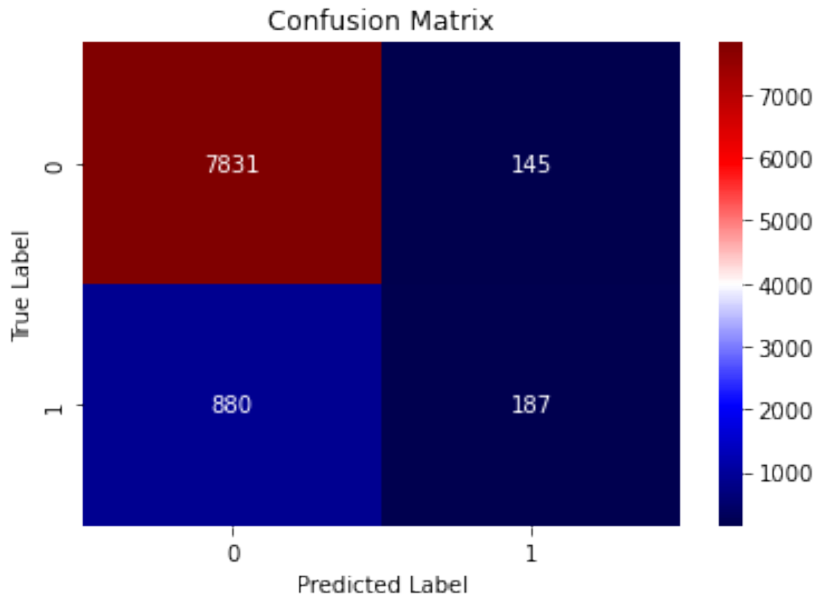
Therefore, we can conclude that the logistic regression model is able to predict the target new variable with an accuracy of approximately 88.67%on unseen data, suggesting that it performs reasonably well in classifying instances based on the given features.

```python
from sklearn.metrics import confusion_matrix

# compute confusion matrix
cm = confusion_matrix(y_test, y_pred)

# display the confusion matrix as a heatmap
sns.heatmap(cm, annot=True, fmt="d", cmap="seismic")

# customize the plot
plt.title("Confusion Matrix")
plt.xlabel("Predicted Label")
plt.ylabel("True Label")
plt.show()
```

A confusion matrix is a table that is used to evaluate the performance of a classification model. By comparing the predictions to the actual labels of the data, it describes the results of predictions generated by a model on a classification issue.

Alongside accuracy, the confusion matrix offers a more thorough analysis of the model's performance. It enables us to assess the model's accuracy, recall, and other assessment metrics as well as the different kinds of errors it makes. The confusion matrix summarizes the true positive(TP) , true negative(TN) , false positive (FP), and false negative (FN) counts. (Narkhede, 2018)

We can learn more about the model's performance in terms of accurately and incorrectly classifying instances by displaying the confusion matrix. This may also help us find areas where the model's predictions could be improved.



In this case the matrix shows that we have correctly predicted 7831 which indicates a true positive and predicted true negative 187 which means that the model was moderately performing well, and it shows that we have predicted false positive and false negative 880 and 145 times respectively, this is an indication that the machine learning model is performing reasonably well.

Decision tree

Decision tree is a supervised machine learning technique that is used for both classification and regression problems. It is a model that resembles a flowchart, with internal nodes standing in for features or attributes, branches for decisions, and leaf nodes for the outcomes or class labels. (Navlani, 2023)

The data is recursively split depending on the values of various attributes using the decision tree method, which then forms the tree. In order to maximize information gain or reduce impurities in the resulting subsets, it selects the most suitable attribute at each stage of the data splitting process. Until a stopping requirement, such as reaching a maximum depth or a minimum number of samples, is satisfied, the splitting process continues.

The majority class at a leaf node is chosen as the projected class for cases falling into that leaf in classification tasks, and each leaf node of the decision tree represents a class label. The shown continuous values in regression tasks are represented by the leaf nodes based on the mean or median of the target variable in that area.

During this process we will be taking a look on how well the features can be predicted using decision tree method. Given below is the code created in order to fit the training data using scikit learn library.

```
from sklearn.tree import DecisionTreeClassifier

#creating an object of Decision tree
clf = DecisionTreeClassifier(max_depth=4, random_state=0)

#fitting the model
clf.fit(X_train, y_train)
```

First and foremost, we must import DecisionTreeClassifier and the create parameter for the model, DecisionTreeClassifier class is created with the specified parameters, max_depth=4 sets the maximum depth of the decision tree to 4, which limits the number of levels or splits the tree can have. random_state=0 sets the random seed to ensure reproducibility of the results.

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

```
array([0, 0, 0, ..., 0, 0, 0])
```

The predict method is used to make predictions on new data using the trained decision tree classifier. In the above given code, the predict method is applied to the test data (X_test) using the trained classifier (clf). The predicted labels for the test data are assigned to the variable prediction.

Based on the decision tree classifier, the outcome's prediction array includes the predicted labels for each instance in the test data. Each prediction is represented by a binary class label (0 or 1 in this instance), which denotes the expected result.

```
DTac = accuracy_score(y_test,prediction)
print("Accuracy:", DTac)
```

```
Accuracy: 0.8879796527700984
```

Accordingly, the above code given calculates the accuracy score of the decision tree classifier predictions, the accuracy of the decision tree classifier on the test data 0.8879 which means that the classifier has correctly predicted he target variable for approximately 88.8%of the instances in the test data.

```
print(classification_report(y_test, prediction))
```

```
              precision    recall  f1-score   support

           0       0.90      0.98      0.94      7976
           1       0.57      0.21      0.30      1067

    accuracy                           0.89      9043
   macro avg       0.74      0.59      0.62      9043
weighted avg       0.86      0.89      0.86      9043
```

The classification_report function from the scikit-learn library is used to generate a detailed classification report for the performance evaluation of the decision tree classifier on the test data. It provides various metrics such as precision, recall, F1-score, and support for each class.

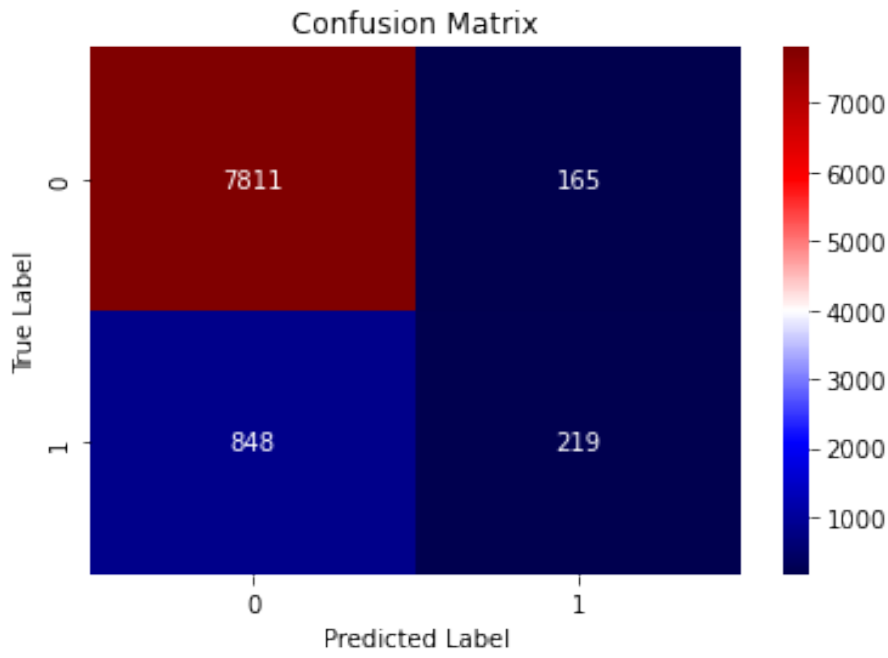
Based on the above classification report we can see that precision for class 0 is 0.90, indicating that 90% of instances predicted as class 0 were actually class 0, for class 1 the precision is 0.57 which means that 57% of the time it was predicted as class 1 it was class 1. The recall for class 0 is 0.98 which means that 98% of actual class instances was predicted correct, whereas for class 1 the recall was 0.21 which means that only 21% of actual class 1 instances was predicted correctly.

The overall accuracy of the decision tree classifier on the test data is 0.89 indicating that the classifier correctly predicted 89% of the instances of the test data.

By understanding the above classification report we can see that the report provides a comprehensive evaluation of the performance of the model, considering multiple metrics for each class and overall accuracy. According to the report, the model predicts class 0 well with high precision and recall but predicts class 1 less accurately with lower precision and recall.

```
cm = confusion_matrix(y_test, prediction)

# display the confusion matrix as a heatmap
sns.heatmap(cm, annot=True, fmt="d", cmap="seismic")

# customize the plot
plt.title("Confusion Matrix")
plt.xlabel("Predicted Label")
plt.ylabel("True Label")
plt.show()
```

In this step we will be conducting a confusion matrix to further analyze and understand the decision tree classifier on the test data.



In this model we have correctly predicted 7811 instances where its true positive and 291 instances where its true negative which indicates that classifier was performing moderately well, it also represents that it has incorrectly predicted class 0, 848 instances and false positive which is class 1, 165 instances. Overall, the model has performed well in predicting class 0 correctly and true negative which is class 1 respectively lower that of class 0. (Kundu, 2022)

References

Navlani, A., 2023. *datacamp.* [Online]
Available at: https://www.datacamp.com/tutorial/decision-tree-classification-python
[Accessed 30 May 2023].
Narkhede, S., 2018. *towardsdatascience.* [Online]
Available at: https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62
[Accessed 29 May 2023].
Swaminathan, S., 2018. *towardsdatascience.* [Online]
Available at: https://towardsdatascience.com/logistic-regression-detailed-overview-46c4da4303bc
[Accessed 30 May 2023].
Galarnyk, M., 2022. *builtin.* [Online]
Available at: https://builtin.com/data-science/train-test-split
[Accessed 30 May 2023].
Szabo, B., 2020. *medium.* [Online]
Available at: https://medium.com/@szabo.bibor/how-to-create-a-seaborn-correlation-heatmap-in-python-834c0686b88e
[Accessed 30 May 2023].
Kundu, R., 2022. *v7labs.* [Online]
Available at: https://www.v7labs.com/blog/confusion-matrix-guide
[Accessed 30 May 2023].