University for the Creative Arts

BERLIN SCHOOL OF BUSINESS & INNOVATION

|  |  |
|---|---|
| Assignment Title | Problematisation of a Data-based Scenario and using Python for Problem Solving |
| Programme title | MSc Data Analytics |
| Name | Sachin Mehta |
| Year | 2023 |

## CONTENTS

## Statement of compliance with academic ethics and the avoidance of plagiarism

I honestly declare that this dissertation is entirely my own work and none of its part has been copied from printed or electronic sources, translated from foreign sources and reproduced from essays of other researchers or students. Wherever I have been based on ideas or other people texts I clearly declare it through the good use of references following academic ethics.

(In the case that is proved that part of the essay does not constitute an original work, but a copy of an already published essay or from another source, the student will be expelled permanently from the postgraduate program).

Name and Surname (Capital letters):

**SACHIN MEHTA**

Date: **2023/06/20**

## INTRODUCTION

Understanding the fundamentals of data management, predictive analytics, and putting machine learning models into practice is essential for resolving practical business issues in the field of data science and analytics. In this introduction, the main learning objectives for these ideas will be briefly discussed.

The first learning outcome (LO1) is concerned with creating a thorough understanding of how to work with various data kinds, such as ordinal and categorical data. It highlights how crucial it is to use data collecting, storage, and preparation methods like encoding to make sure that the data is ready for analysis and modeling. The second learning outcome (LO2) delves into the clustering, classification, and regression models that make up predictive analytics. Students learn how to analyze and understand data to efficiently handle challenging business situations by investigating these models. While classification models enable the categorization of data based on specified classifications, regression models make it easier to anticipate numerical results. On the other hand, clustering models combine comparable data elements. The third learning objective (LO3) entails using diverse models and project life cycles practically. Students are introduced to real-world situations where they use artificial intelligence and machine learning techniques to address analytical issues. This result promotes a comprehensive approach to problem-solving, starting with the identification of the business issue, followed by the selection of suitable models, training, and evaluation, and finally the delivery of a resolution that is in line with company goals.

# CHAPTER ONE: Task 1

The goal of Task 1 is to choose a good machine learning method and locate or prepare a sufficient dataset for that algorithm. We stress the significance of maintaining dataset cleanliness and give justifications for it. This job relates to learning objectives LO1 and LO2, which deal with comprehending various data types, preparing data, and utilizing predictive analytics models. The dataset for credit card customers can be found at Kaggle ***"BankChurners.csv"***. It includes details on clients, including their demographic traits, credit card usage habits, and churn status. The use of this dataset for numerous predictive analytics activities involving consumer behaviour, credit card usage, and churn prediction is possible (Acciarini *et al*. 2023).

The dataset contains a number of pertinent characteristics, including information about the customer's age, gender, educational background, income level, marital status, and credit card information, including transaction history, average utilisation ratio, and credit limit. These highlights give valuable data on client profiles and monetary way of behaving, which makes it proper for dissecting real organization issues with Visa utilization and client turnover.

Organizations can involve this data for relapse investigation to gauge charge card ways of managing money in light of client socioeconomics and different factors (Doran *et al*. 2022).

It can also be applied to classification jobs to determine the variables affecting customer churn and forecast whether a client will leave or stay. Furthermore, clustering algorithms can be used to divide up the client base into groups according to their behaviour and preferences, enabling targeted marketing plans and personalised advertising.

The dataset is additionally well-maintained and prepared for study thanks to preprocessing. It has a huge quantity of records, making it possible to train and evaluate robust models. It is useful for developing comprehensive predictive models and obtaining valuable insights because it is possible to access a variety of customer attributes and data on credit card usage. Organizations and researchers can undertake predictive analytics tasks relating to credit card usage, consumer behavior, and churn prediction using the credit card customer dataset supplied in the link. It is a useful tool for examining actual business issues in the credit card sector because of its thoroughness, cleanliness, and adaptability for regression, classification, and clustering (Fathi *et al*. 2022).

The first task in Task 1 entails choosing a suitable machine-learning method depending on the current issue. The algorithm chosen will vary depending on the nature of the problem, the data at hand, and the intended results. For instance, regression issues where the goal is to predict a continuous numerical value may be ideal for linear regression, decision trees, or random forests. On the other hand, classification problems might be better served by logistic regression, support vector machines, or neural networks. The same is true for clustering issues; clustering approaches like k-means or hierarchical clustering can be taken into account. Finding an existing dataset or creating one that is appropriate for training and evaluating the chosen machine learning algorithm is the next step after choosing the method. Datasets can be obtained in a variety of ways, including openly accessible datasets, company-owned data, or synthetic datasets made specifically for experiments (Prakash *et al*. 2022).

Reasons to guarantee clean datasets:

To obtain accurate and trustworthy findings, it is essential to make sure the dataset is clean. The significance of clean datasets is underscored by a number of important factors:

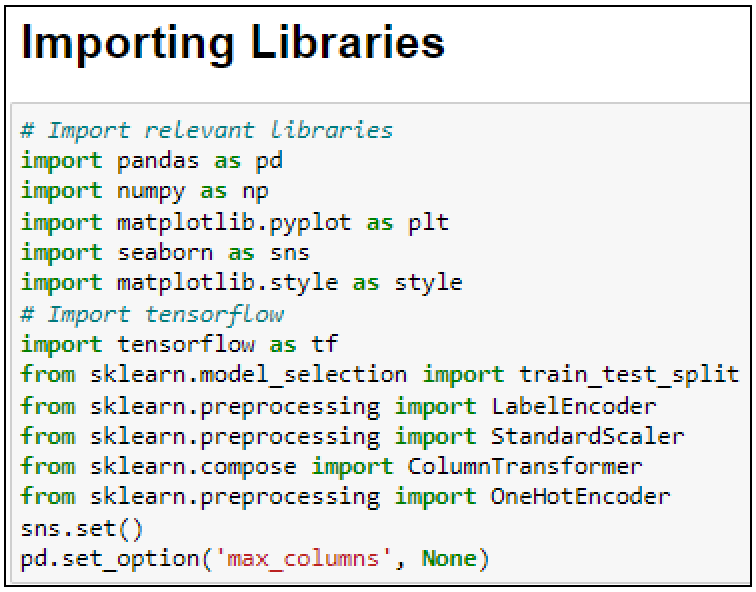
- Results are more accurate and dependable when the presence of mistakes and inconsistencies is minimized, which is achieved by using a clean dataset. The machine learning model can effectively identify important patterns and produce accurate predictions when the data is free of errors, missing values, or outliers.
- Dataset cleanliness helps to reduce biases that may result from inaccurate or missing data. The performance and fairness of the machine learning model can be severely impacted by biases, which can lead to biassed predictions and potentially bad decision-making.
- Consistent Data Representation: The standardization of data kinds, units, and formats is made possible by clean data, which guarantees consistent data representation. The capacity to compare results across datasets and simplify data processing and analysis are all made possible by consistency.
- Clean datasets help with effective model training by cutting down on the processing time and computational resources needed. The machine learning model can concentrate on discovering pertinent patterns and relationships when the data is clean rather than dedicating resources to handle missing values or outliers (Akter *et al*. 2022).

- Generalization to New Data which is the model's capacity to generalize well to new data is improved by a clean dataset. A model that has been trained on a clean dataset will have learned robust features and patterns that are more likely to hold true when faced with brand-new, unexplored data instances.
- Improved Interpretability is a clean dataset that makes the machine learning model easier to understand. The interpretation and justification of the model's predictions and insights to stakeholders are facilitated by clean, well-documented data.

# CHAPTER TWO: Task 2

The development of multiple machine learning models using predictive analytics is shown in the given Python code. The code relates to the learning objectives for comprehending various data kinds, using predictive analytics models, and managing a whole project lifetime (Yin *et al.*, 2021).

The code starts by importing the essential libraries, including matplotlib, seaborn, pandas, and numpy. These libraries include features for model creation, data processing, and visualization. The code demonstrates the application of the ideas related to dealing with various forms of data by utilising these libraries.

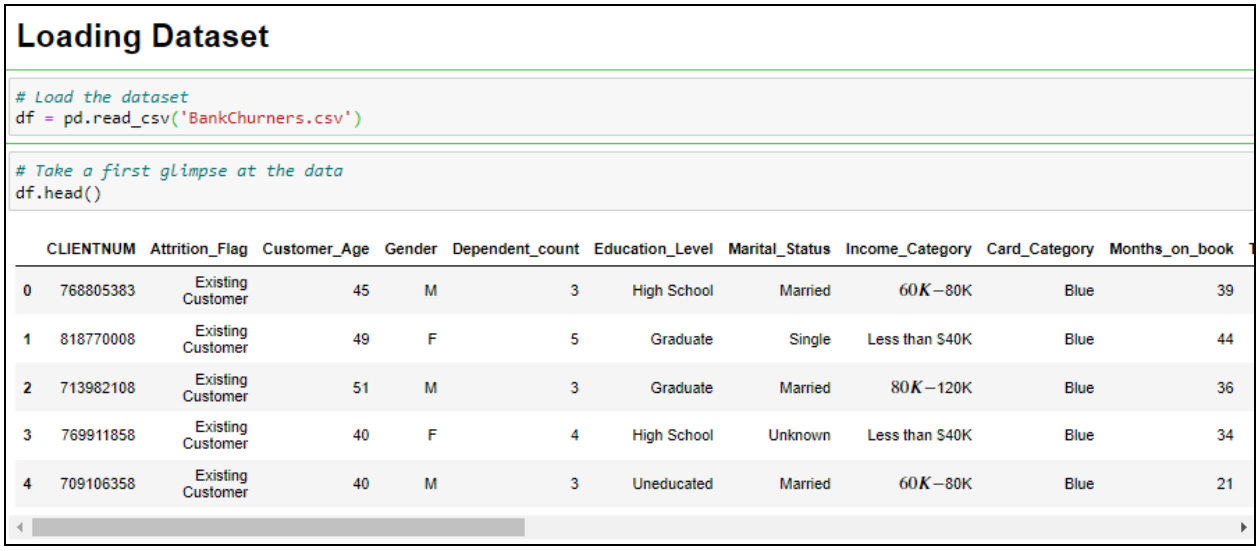


## Importing Libraries

```
# Import relevant libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import matplotlib.style as style
# Import tensorflow
import tensorflow as tf
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import StandardScaler
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
sns.set()
pd.set_option('max_columns', None)
```

**Figure 1: Importing libraries**

Next, the pd.read_csv() method loads a dataset from a CSV file. This action fits with how data collection and storage are understood in the processing process. The dataset.head() method displays a preview of the data after it has been loaded, giving users a quick overview of the dataset's structure and contents.



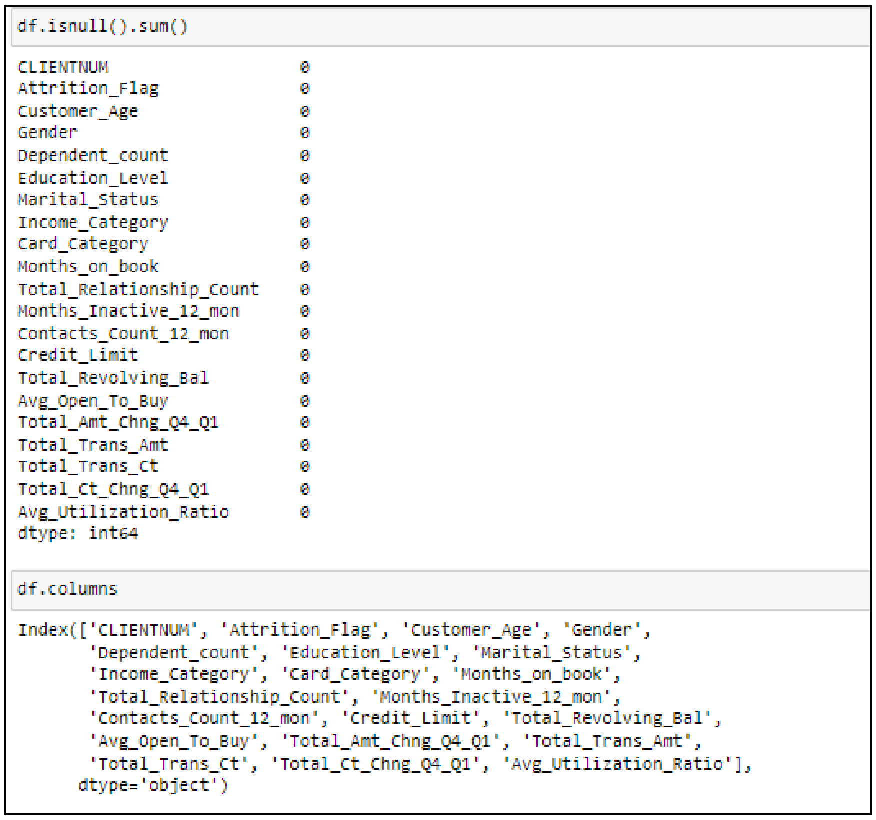
## Loading Dataset

```
# Load the dataset
df = pd.read_csv('BankChurners.csv')
```

```
# Take a first glimpse at the data
df.head()
```

| | CLIENTNUM | Attrition_Flag | Customer_Age | Gender | Dependent_count | Education_Level | Marital_Status | Income_Category | Card_Category | Months_on_book |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 768805383 | Existing Customer | 45 | M | 3 | High School | Married | 60$K-$80K | Blue | 39 |
| 1 | 818770008 | Existing Customer | 49 | F | 5 | Graduate | Single | Less than $40K | Blue | 44 |
| 2 | 713982108 | Existing Customer | 51 | M | 3 | Graduate | Married | 80$K-$120K | Blue | 36 |
| 3 | 769911858 | Existing Customer | 40 | F | 4 | High School | Unknown | Less than $40K | Blue | 34 |
| 4 | 709106358 | Existing Customer | 40 | M | 3 | Uneducated | Married | 60$K-$80K | Blue | 21 |

**Figure 2: Loading the dataset**



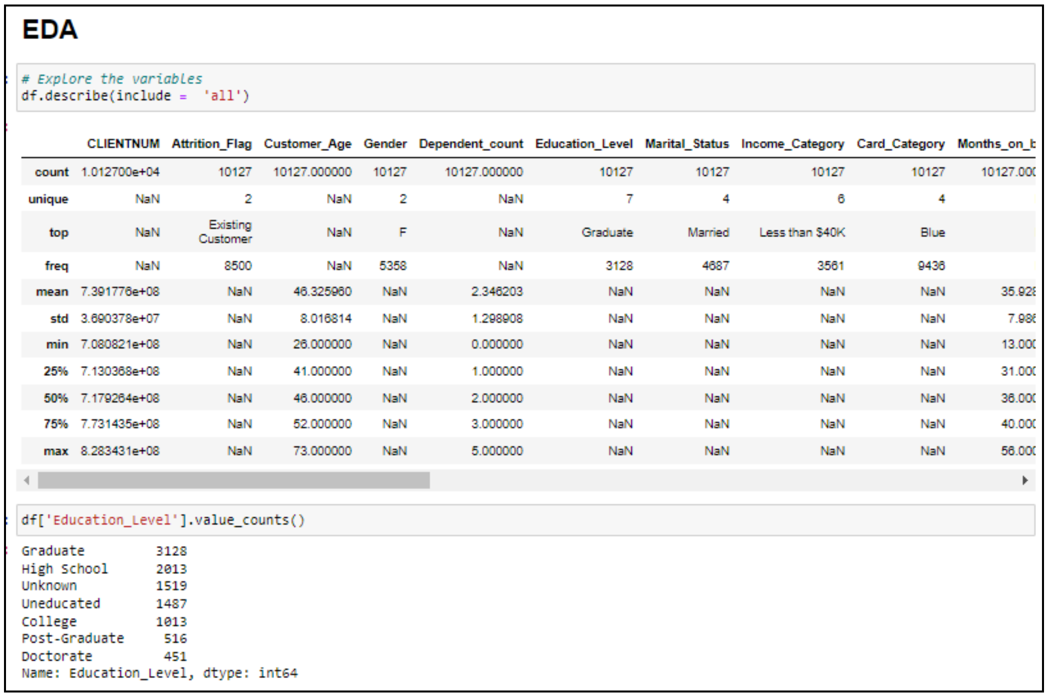
```
df.isnull().sum()
```

```
CLIENTNUM                   0
Attrition_Flag              0
Customer_Age                0
Gender                      0
Dependent_count             0
Education_Level             0
Marital_Status              0
Income_Category             0
Card_Category               0
Months_on_book              0
Total_Relationship_Count    0
Months_Inactive_12_mon      0
Contacts_Count_12_mon       0
Credit_Limit                0
Total_Revolving_Bal         0
Avg_Open_To_Buy             0
Total_Amt_Chng_Q4_Q1        0
Total_Trans_Amt             0
Total_Trans_Ct              0
Total_Ct_Chng_Q4_Q1         0
Avg_Utilization_Ratio       0
dtype: int64
```

```
df.columns
```

```
Index(['CLIENTNUM', 'Attrition_Flag', 'Customer_Age', 'Gender',
       'Dependent_count', 'Education_Level', 'Marital_Status',
       'Income_Category', 'Card_Category', 'Months_on_book',
       'Total_Relationship_Count', 'Months_Inactive_12_mon',
       'Contacts_Count_12_mon', 'Credit_Limit', 'Total_Revolving_Bal',
       'Avg_Open_To_Buy', 'Total_Amt_Chng_Q4_Q1', 'Total_Trans_Amt',
       'Total_Trans_Ct', 'Total_Ct_Chng_Q4_Q1', 'Avg_Utilization_Ratio'],
      dtype='object')
```

**Figure 3: Checking the missing values of the dataset**



EDA

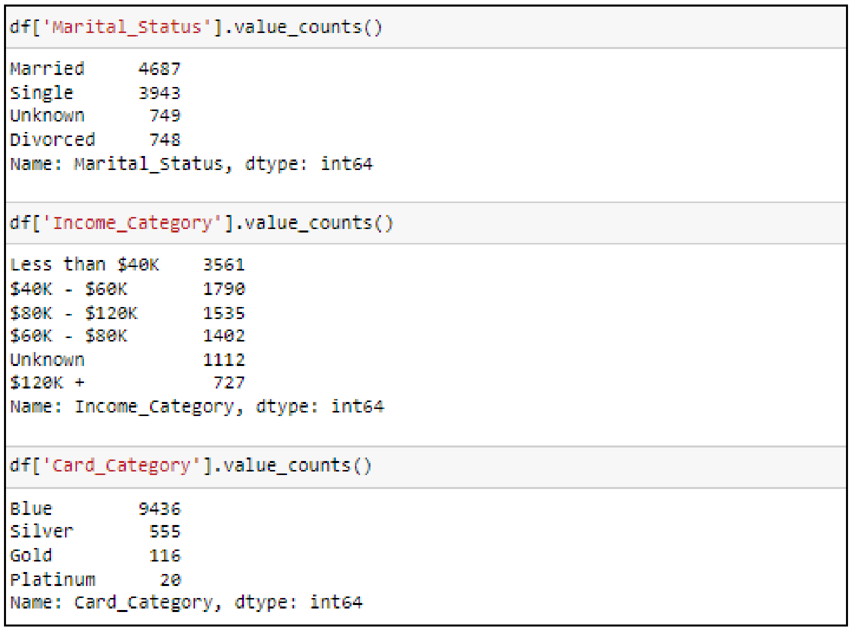
```
# Explore the variables
df.describe(include =  'all')
```

| | CLIENTNUM | Attrition_Flag | Customer_Age | Gender | Dependent_count | Education_Level | Marital_Status | Income_Category | Card_Category | Months_on_b |
|---|---|---|---|---|---|---|---|---|---|---|
| count | 1.012700e+04 | 10127 | 10127.000000 | 10127 | 10127.000000 | 10127 | 10127 | 10127 | 10127 | 10127.00 |
| unique | NaN | 2 | NaN | 2 | NaN | 7 | 4 | 6 | 4 | |
| top | NaN | Existing Customer | NaN | F | NaN | Graduate | Married | Less than $40K | Blue | |
| freq | NaN | 8500 | NaN | 5358 | NaN | 3128 | 4687 | 3561 | 9436 | |
| mean | 7.391776e+08 | NaN | 46.325960 | NaN | 2.346203 | NaN | NaN | NaN | NaN | 35.92 |
| std | 3.690378e+07 | NaN | 8.016814 | NaN | 1.298908 | NaN | NaN | NaN | NaN | 7.98 |
| min | 7.080821e+08 | NaN | 26.000000 | NaN | 0.000000 | NaN | NaN | NaN | NaN | 13.00 |
| 25% | 7.130368e+08 | NaN | 41.000000 | NaN | 1.000000 | NaN | NaN | NaN | NaN | 31.00 |
| 50% | 7.179264e+08 | NaN | 46.000000 | NaN | 2.000000 | NaN | NaN | NaN | NaN | 36.00 |
| 75% | 7.731435e+08 | NaN | 52.000000 | NaN | 3.000000 | NaN | NaN | NaN | NaN | 40.00 |
| max | 8.283431e+08 | NaN | 73.000000 | NaN | 5.000000 | NaN | NaN | NaN | NaN | 56.00 |

```
df['Education_Level'].value_counts()
```

```
Graduate         3128
High School      2013
Unknown          1519
Uneducated       1487
College          1013
Post-Graduate     516
Doctorate         451
Name: Education_Level, dtype: int64
```

**Figure 4: Basic steps of EDA**



```
df['Marital_Status'].value_counts()
```

```
Married     4687
Single      3943
Unknown      749
Divorced     748
Name: Marital_Status, dtype: int64
```

```
df['Income_Category'].value_counts()
```

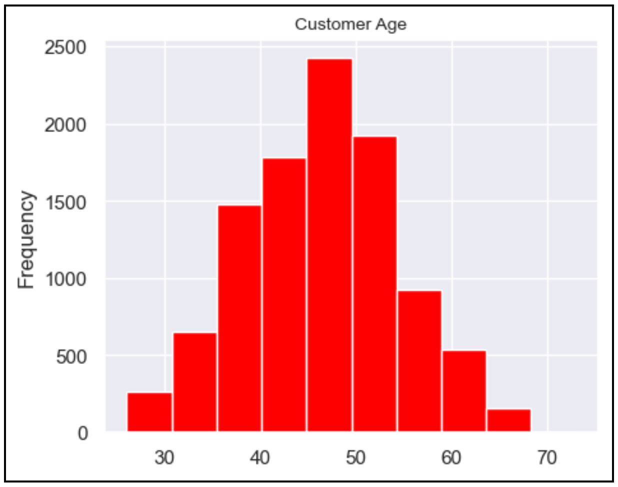
```
Less than $40K    3561
$40K - $60K       1790
$80K - $120K      1535
$60K - $80K       1402
Unknown           1112
$120K +            727
Name: Income_Category, dtype: int64
```

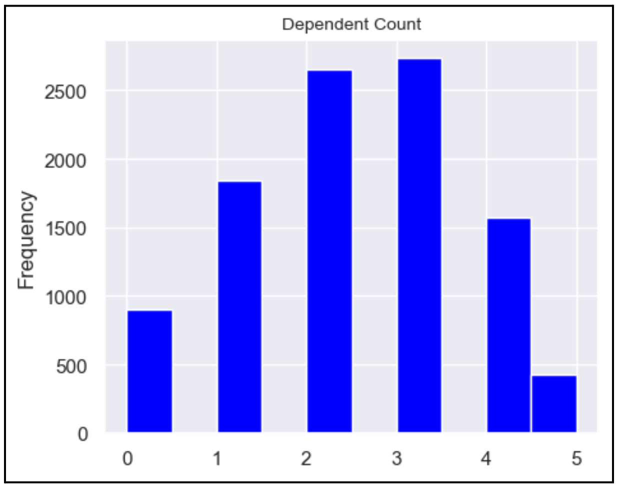
```
df['Card_Category'].value_counts()
```

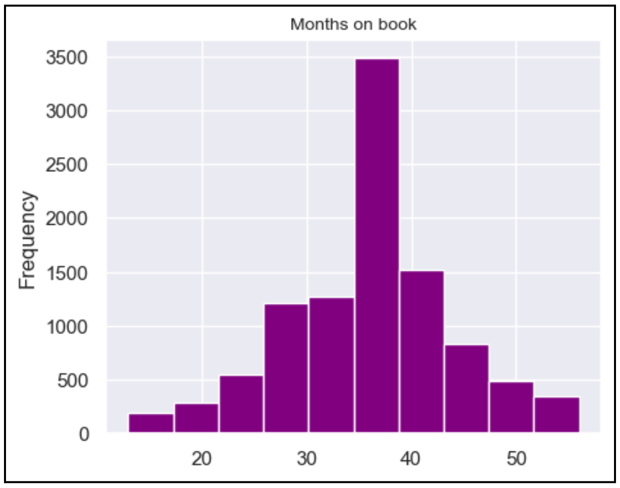
```
Blue        9436
Silver       555
Gold         116
Platinum      20
Name: Card_Category, dtype: int64
```

**Figure 5: Analyzing the features**

The program provides several visualizations utilizing histograms and bar graphs for "exploratory data analysis (EDA)". These visualizations aid in the analysis of the dataset's many characteristics, including client age, the number of dependents, credit limit, and more. This relates to the comprehension of fundamental ideas in data analysis and visualization.


Customer Age
Frequency
2500
2000
1500
1000
500
0
30
40
50
60
70


Dependent Count
Frequency
2500
2000
1500
1000
500
0
0
1
2
3
4
5


Months on book
Frequency
3500
3000
2500
2000
1500
1000
500
0
20
30
40
50

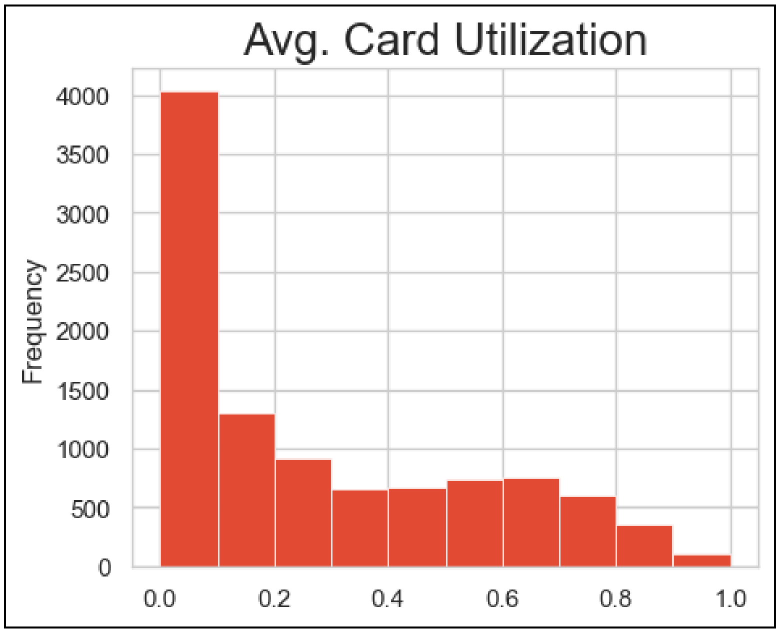
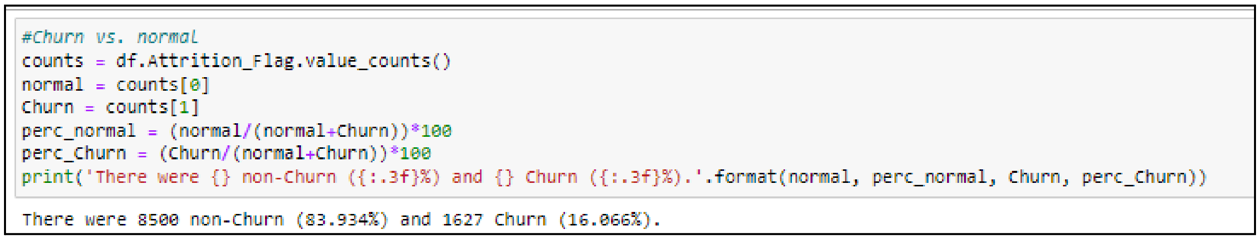


**Figure 6: Some important insights of the dataset**

The code computes and shows the counts of many categorical variables, including income category, marital status, degree of education, and credit card type. This level emphasizes the management of categorical data and aids in the comprehension of data encoding methods.



```
#Churn vs. normal
counts = df.Attrition_Flag.value_counts()
normal = counts[0]
Churn = counts[1]
perc_normal = (normal/(normal+Churn))*100
perc_Churn = (Churn/(normal+Churn))*100
print('There were {} non-Churn ({:.3f}%) and {} Churn ({:.3f}%).'.format(normal, perc_normal, Churn, perc_Churn))

There were 8500 non-Churn (83.934%) and 1627 Churn (16.066%).
```

**Figure 7: churned vs. non-churned**

The value_counts() method is used in the code to compute and display the ratio of lost customers to new ones. This offers information about the dataset's distribution of churned alongside non-churned consumers.

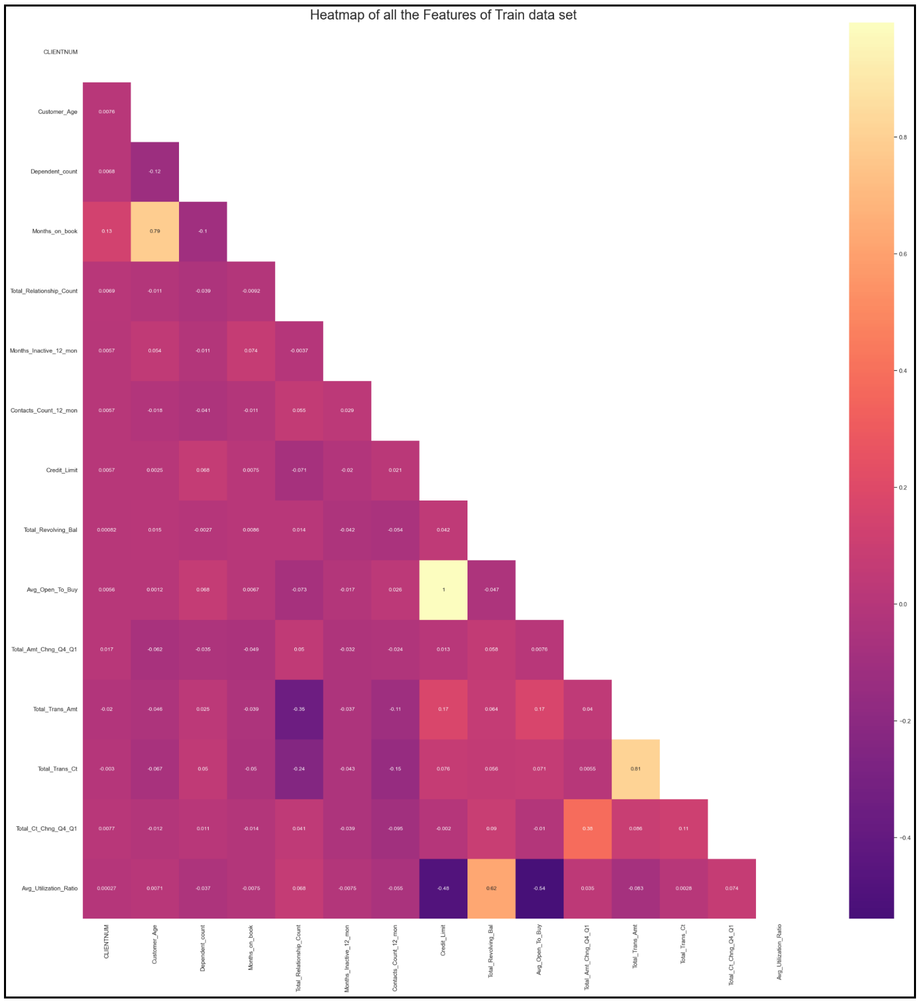
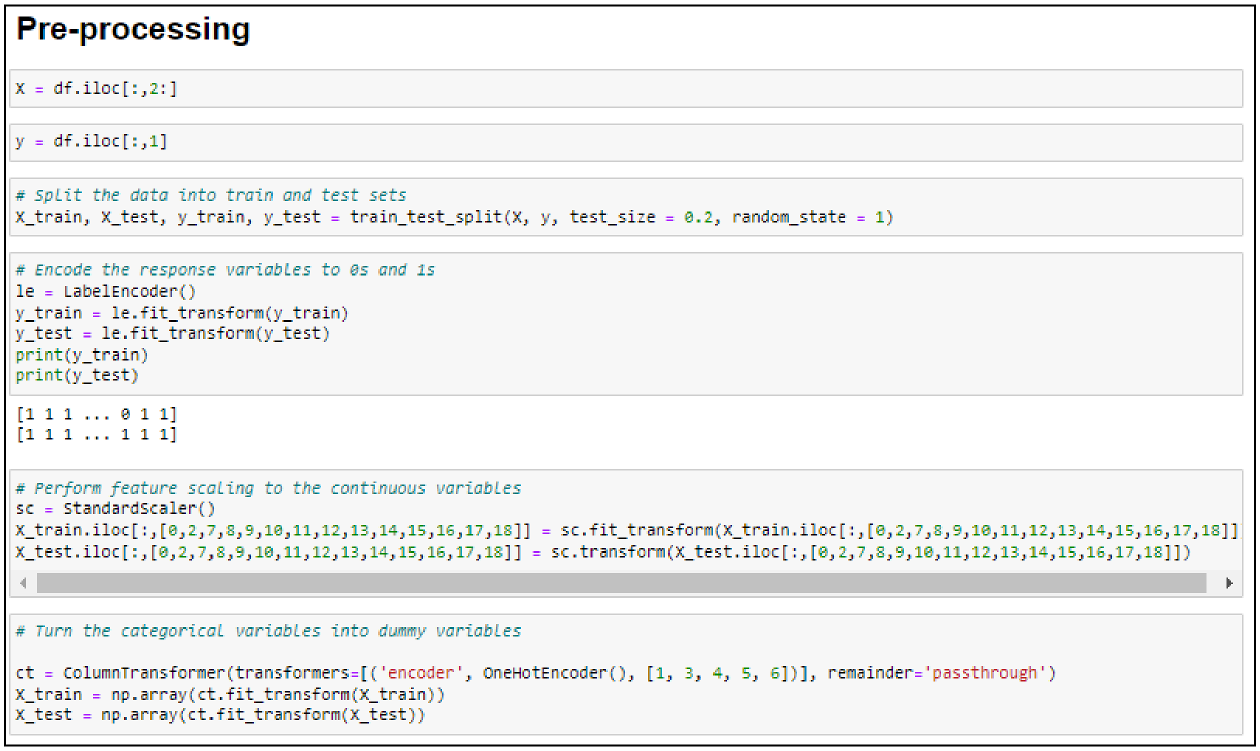


**Figure 8: Heat map of all the features**

The code divides the dataset into sets for both training and testing using the train_test_split() method from the sklearn.model_selection module before moving on to the model implementation section. For the training and assessment of models, this stage is crucial (Gayialis *et al.*, 2022).

The StandardScaler() method from the sklearn.preprocessing package is then used to apply feature scaling. Selected continuous variables are subjected to feature scaling to guarantee that their values are on an equivalent scale, which helps with model training.



Pre-processing

```
X = df.iloc[:,2:]
```

```
y = df.iloc[:,1]
```

```
# Split the data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 1)
```

```
# Encode the response variables to 0s and 1s
le = LabelEncoder()
y_train = le.fit_transform(y_train)
y_test = le.fit_transform(y_test)
print(y_train)
print(y_test)
```

```
[1 1 1 ... 0 1 1]
[1 1 1 ... 1 1 1]
```

```
# Perform feature scaling to the continuous variables
sc = StandardScaler()
X_train.iloc[:,[0,2,7,8,9,10,11,12,13,14,15,16,17,18]] = sc.fit_transform(X_train.iloc[:,[0,2,7,8,9,10,11,12,13,14,15,16,17,18]]
X_test.iloc[:,[0,2,7,8,9,10,11,12,13,14,15,16,17,18]] = sc.transform(X_test.iloc[:,[0,2,7,8,9,10,11,12,13,14,15,16,17,18]])
```

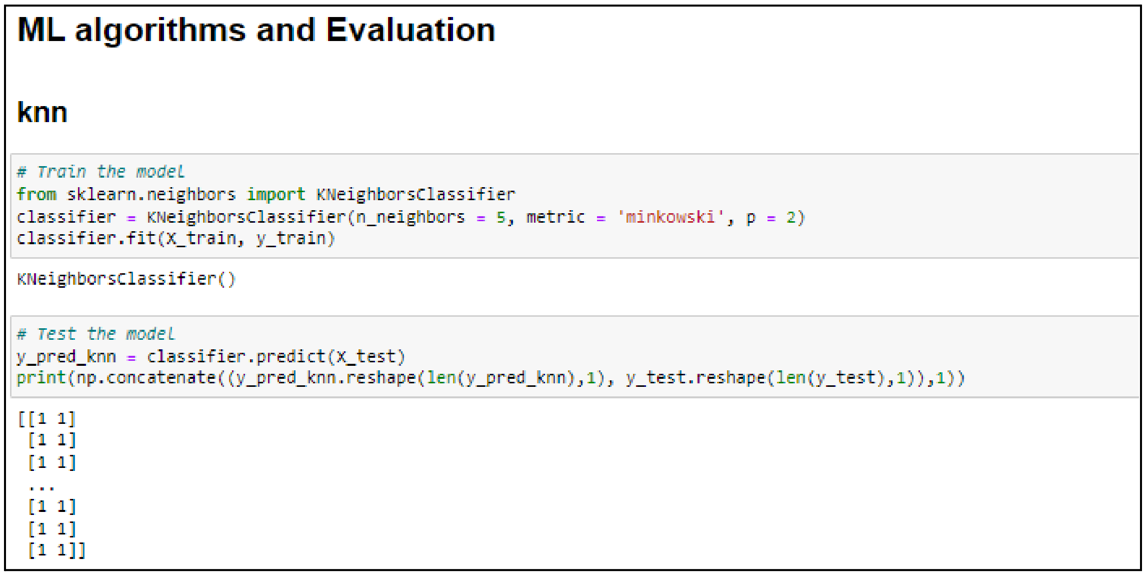
```
# Turn the categorical variables into dummy variables

ct = ColumnTransformer(transformers=[('encoder', OneHotEncoder(), [1, 3, 4, 5, 6])], remainder='passthrough')
X_train = np.array(ct.fit_transform(X_train))
X_test = np.array(ct.fit_transform(X_test))
```

**Figure 9: Pre-processing steps including train-test splitting, encoding, and standardizing the train and test sets**

Next, the algorithm converts category variables into dummy variables using one-hot encoding. The classes ColumnTransformer and OneHotEncoder from the sklearn.compose as well as sklearn.preprocessing modules, respectively, are used to do this transformation. Due to the fact that many machine learning methods require numerical input, this phase gets the data ready for model training. A number of machine learning models, like "k-Nearest Neighbours (k-NN)", "Logistic Regression", "Support Vector Machine (SVM)", "Decision Tree", "Random Forest", and "Artificial Neural Network (ANN)", are then implemented in the code. Each model is tested once it has been trained using the testing set. The confusion matrices and accuracy scores are computed and saved for analysis.

Several classifiers from the “sklearn.neighbors, sklearn.linear_model, sklearn.svm, sklearn.tree, sklearn.ensemble”, as well as “tensorflow.keras.models” modules are used in the code to show their application. This relates to the knowledge gained by using different predictive analytics algorithms.

The code then uses the tabulate function to deliver the results in a tabular fashion. A table showing the confusion matrices and accuracy scores for each model is presented. This process makes it simple to compare and assess the performance of the models.



ML algorithms and Evaluation

knn

```
# Train the model
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(n_neighbors = 5, metric = 'minkowski', p = 2)
classifier.fit(X_train, y_train)
```

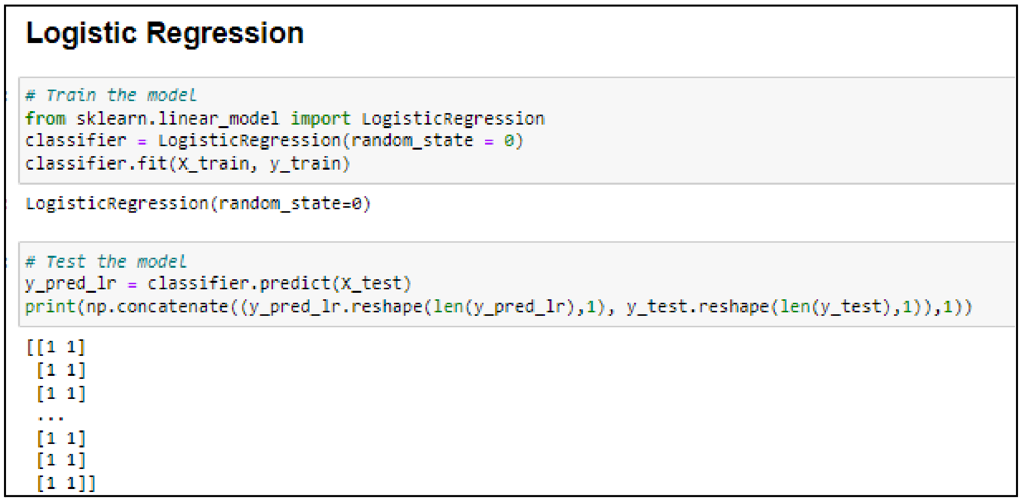
```
KNeighborsClassifier()
```

```
# Test the model
y_pred_knn = classifier.predict(X_test)
print(np.concatenate((y_pred_knn.reshape(len(y_pred_knn),1), y_test.reshape(len(y_test),1)),1))
```

```
[[1 1]
 [1 1]
 [1 1]
 ...
 [1 1]
 [1 1]
 [1 1]]
```

**Figure 10: Implementing KNN algorithm**

The k-NN technique is used in the code to do classification. Finding the k closest neighbours to a given data point and categorizing it according to the majority class of those neighbours is how this method operates. The performance of the model may be affected by the selection of k, the number of neighbours. The k-NN classifier is trained using the training set and the 'minkowski' distance metric with k set to 5. The testing set is then used to assess the model's predictions.



Logistic Regression

```
# Train the model
from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression(random_state = 0)
classifier.fit(X_train, y_train)
```

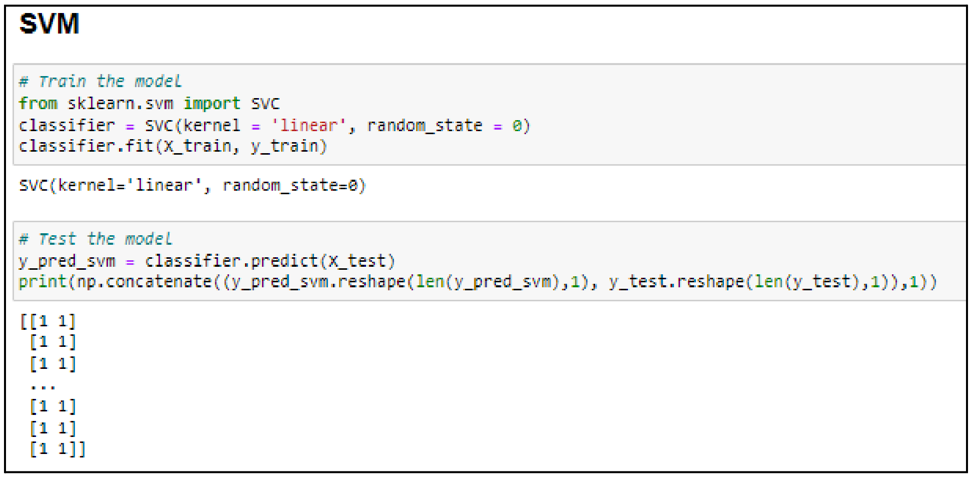
```
LogisticRegression(random_state=0)
```

```
# Test the model
y_pred_lr = classifier.predict(X_test)
print(np.concatenate((y_pred_lr.reshape(len(y_pred_lr),1), y_test.reshape(len(y_test),1)),1))
```

```
[[1 1]
 [1 1]
 [1 1]
 ...
 [1 1]
 [1 1]
 [1 1]]
```

**Figure 11: Implementing Logistic Regression**

For categorization, logistic regression is then used. An effective approach for binary classification is logistic regression. A logistic function is used to represent the likelihood that the target variable will belong to a particular class. The program trains a logistic regression model on the training set using the LogisticRegression class from the “sklearn.linear_model package”. By producing predictions on the testing set and contrasting them with the real labels, the performance of the model is evaluated.



SVM

```
# Train the model
from sklearn.svm import SVC
classifier = SVC(kernel = 'linear', random_state = 0)
classifier.fit(X_train, y_train)
```

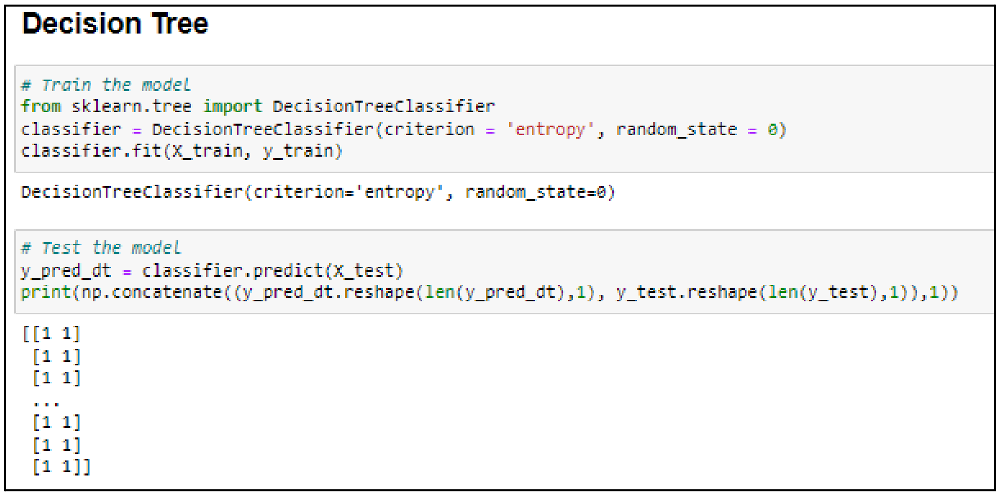
```
SVC(kernel='linear', random_state=0)
```

```
# Test the model
y_pred_svm = classifier.predict(X_test)
print(np.concatenate((y_pred_svm.reshape(len(y_pred_svm),1), y_test.reshape(len(y_test),1)),1))
```

```
[[1 1]
 [1 1]
 [1 1]
 ...
 [1 1]
 [1 1]
 [1 1]]
```

**Figure 12: Implementing SVM algorithm**

The SVM classifier using a linear kernel is also included in the code. SVM is a potent method for both regression and classification tasks. It looks for a hyperplane that divides the classes within the feature space as widely as possible. The SVM classifier in this code is tested on the testing set after being trained on the initial set using a linear kernel.



Decision Tree

```
# Train the model
from sklearn.tree import DecisionTreeClassifier
classifier = DecisionTreeClassifier(criterion = 'entropy', random_state = 0)
classifier.fit(X_train, y_train)
```

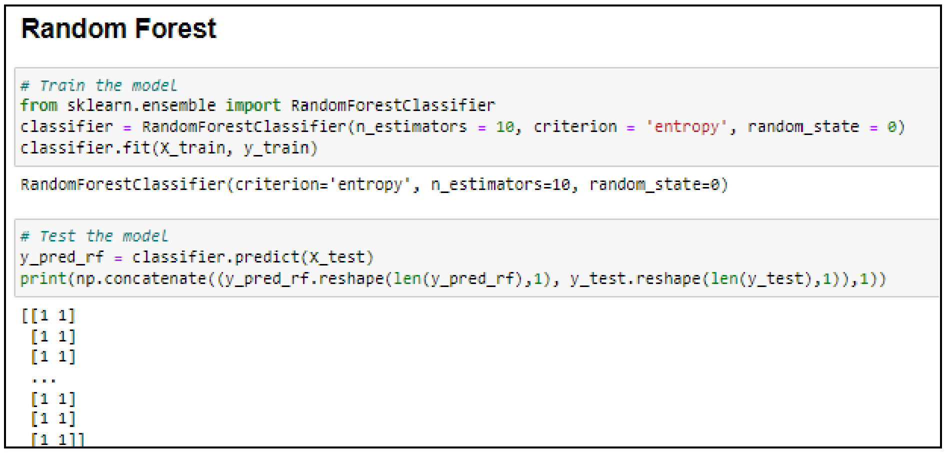
```
DecisionTreeClassifier(criterion='entropy', random_state=0)
```

```
# Test the model
y_pred_dt = classifier.predict(X_test)
print(np.concatenate((y_pred_dt.reshape(len(y_pred_dt),1), y_test.reshape(len(y_test),1)),1))
```

```
[[1 1]
 [1 1]
 [1 1]
 ...
 [1 1]
 [1 1]
 [1 1]]
```

**Figure 13: Implementing Decision Tree algorithm**

It uses the Decision Tree classifier. Decision trees are flexible models that infer from the data hierarchical decision rules. They divide the feature space according to certain criteria until they reach pure or almost pure leaf nodes. To train a decision tree model on the training set, the code makes use of the DecisionTreeClassifier class from the "sklearn.tree module". The effectiveness of the model is then evaluated on the testing set.



Random Forest

```
# Train the model
from sklearn.ensemble import RandomForestClassifier
classifier = RandomForestClassifier(n_estimators = 10, criterion = 'entropy', random_state = 0)
classifier.fit(X_train, y_train)
```

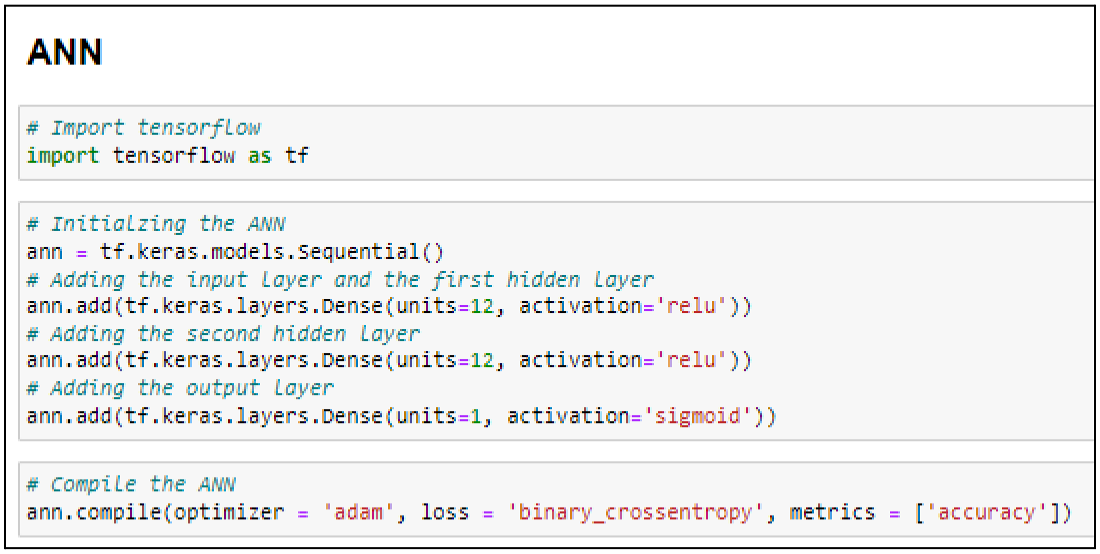
```
RandomForestClassifier(criterion='entropy', n_estimators=10, random_state=0)
```

```
# Test the model
y_pred_rf = classifier.predict(X_test)
print(np.concatenate((y_pred_rf.reshape(len(y_pred_rf),1), y_test.reshape(len(y_test),1)),1))
```

```
[[1 1]
 [1 1]
 [1 1]
 ...
 [1 1]
 [1 1]
 [1 1]]
```

**Figure 14: Implementing Random forest model**

The Random Forest classifier is included in the code. Multiple decision trees are used in Random Forest, an ensemble learning technique, to enhance performance and minimise overfitting. To produce the final categorization, it builds a group of decision trees and combines their predictions. The programme trains a random forest model on the training set using the RandomForestClassifier class from the "sklearn.ensemble package". The testing set is used to assess the model's effectiveness.



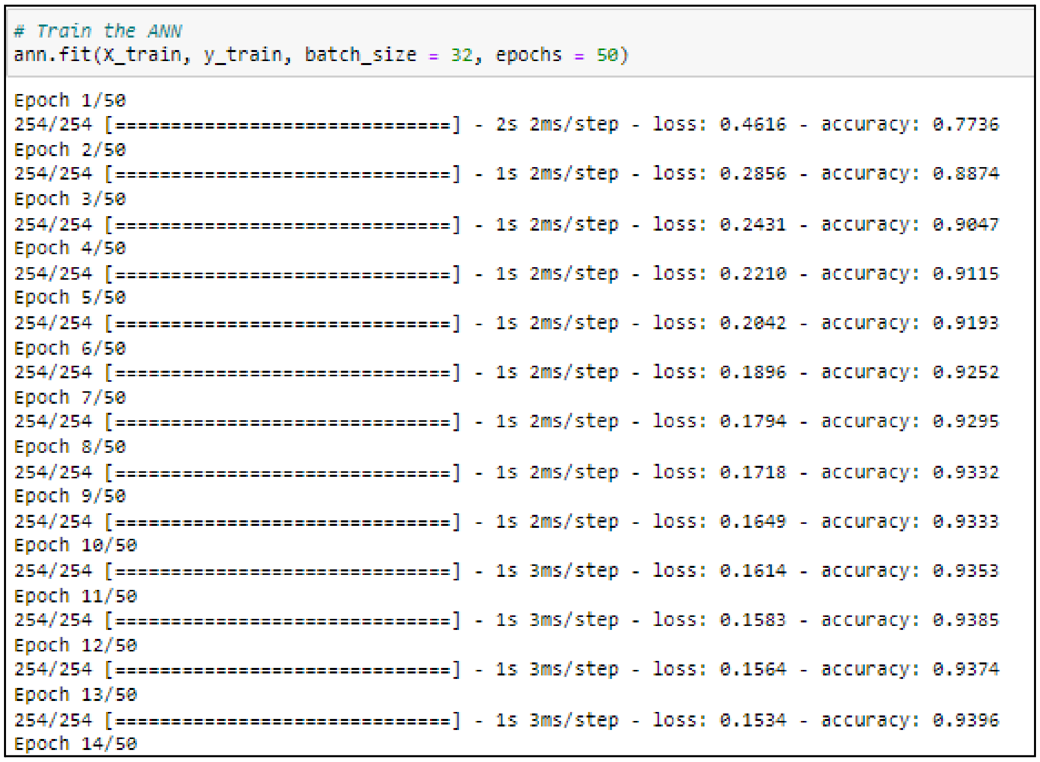
ANN

```
# Import tensorflow
import tensorflow as tf
```

```
# Initialzing the ANN
ann = tf.keras.models.Sequential()
# Adding the input layer and the first hidden layer
ann.add(tf.keras.layers.Dense(units=12, activation='relu'))
# Adding the second hidden layer
ann.add(tf.keras.layers.Dense(units=12, activation='relu'))
# Adding the output layer
ann.add(tf.keras.layers.Dense(units=1, activation='sigmoid'))
```

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

**Figure 15: Creating ANN model**



```
# Train the ANN
ann.fit(X_train, y_train, batch_size = 32, epochs = 50)
```

```
Epoch 1/50
254/254 [==============================] - 2s 2ms/step - loss: 0.4616 - accuracy: 0.7736
Epoch 2/50
254/254 [==============================] - 1s 2ms/step - loss: 0.2856 - accuracy: 0.8874
Epoch 3/50
254/254 [==============================] - 1s 2ms/step - loss: 0.2431 - accuracy: 0.9047
Epoch 4/50
254/254 [==============================] - 1s 2ms/step - loss: 0.2210 - accuracy: 0.9115
Epoch 5/50
254/254 [==============================] - 1s 2ms/step - loss: 0.2042 - accuracy: 0.9193
Epoch 6/50
254/254 [==============================] - 1s 2ms/step - loss: 0.1896 - accuracy: 0.9252
Epoch 7/50
254/254 [==============================] - 1s 2ms/step - loss: 0.1794 - accuracy: 0.9295
Epoch 8/50
254/254 [==============================] - 1s 2ms/step - loss: 0.1718 - accuracy: 0.9332
Epoch 9/50
254/254 [==============================] - 1s 2ms/step - loss: 0.1649 - accuracy: 0.9333
Epoch 10/50
254/254 [==============================] - 1s 3ms/step - loss: 0.1614 - accuracy: 0.9353
Epoch 11/50
254/254 [==============================] - 1s 3ms/step - loss: 0.1583 - accuracy: 0.9385
Epoch 12/50
254/254 [==============================] - 1s 3ms/step - loss: 0.1564 - accuracy: 0.9374
Epoch 13/50
254/254 [==============================] - 1s 3ms/step - loss: 0.1534 - accuracy: 0.9396
Epoch 14/50
```

**Figure 16: Training the ANN model**

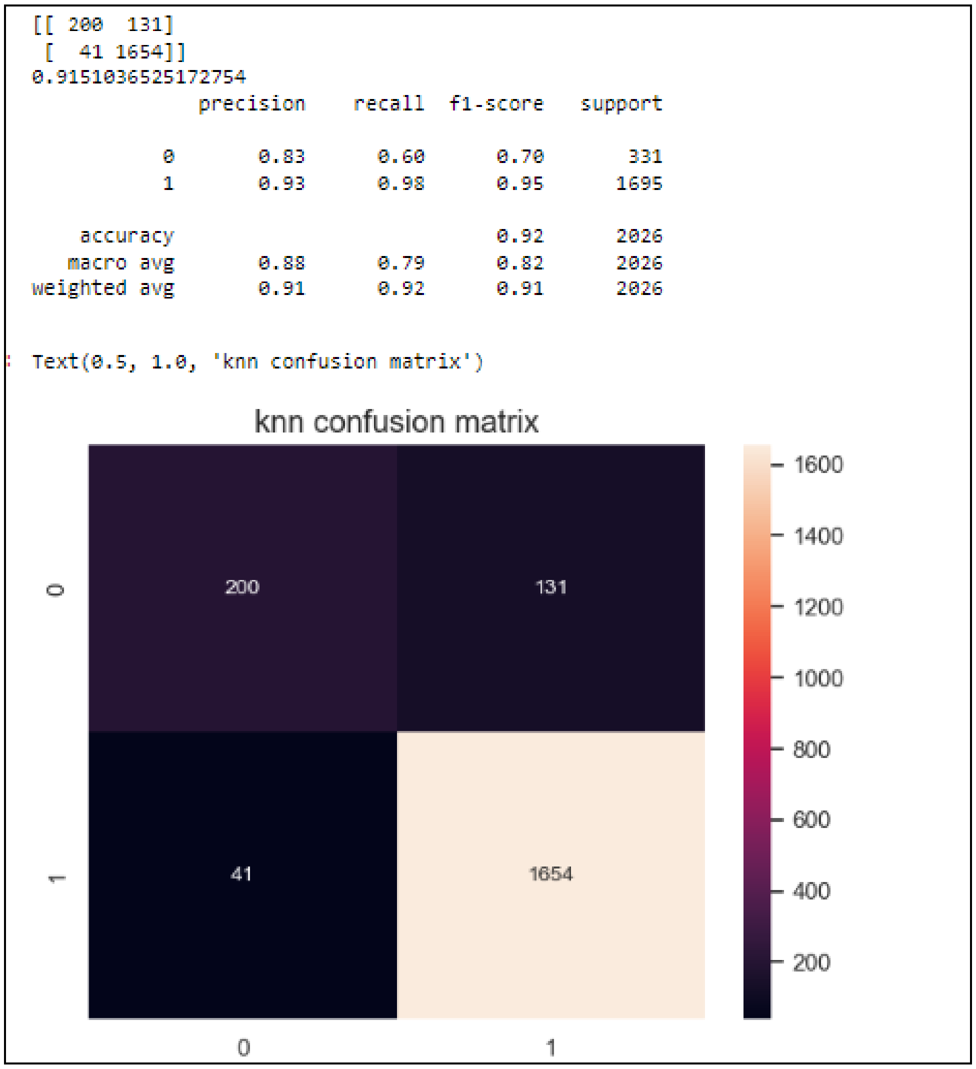
The code then uses the TensorFlow package to create an "Artificial Neural Network (ANN)". ANNs are potent models that draw inspiration from the design and operation of the human brain. They are made up of several linked layers of artificial neurons, and they can recognise intricate patterns in the data. The Sequential class is found in the tensorflow.keras.models module is used in this code to build a sequential model. The model is trained on the training set using the Adam optimizer and binary cross-entropy loss function. The model is then evaluated by comparing its predictions to the testing set's true labels.

k-NN, logistic regression, SVM, decision tree, random forest, and ANN are just a few of the machine learning methods for classification problems covered by the given Python code. The following dataset is used to implement and evaluate each method. Learners may gain a basic understanding of dealing with various types of data, implementing predictive analytics models, as well as going through the development lifecycle from data pretreatment to model assessment by comprehending the code and its components. By offering hands-on experience with actual business issues and showcasing the use of machine learning and other artificial intelligence approaches, the code successfully links to the course's learning objectives.

The Python code presented exemplifies the comprehension and implementation of fundamental ideas in data management, predictive analytics, and the project lifecycle. It provides examples of data exploration, model training, and assessment. The code is in line with the learning objectives of comprehending various data kinds, using predictive analytics, and working on whole projects.

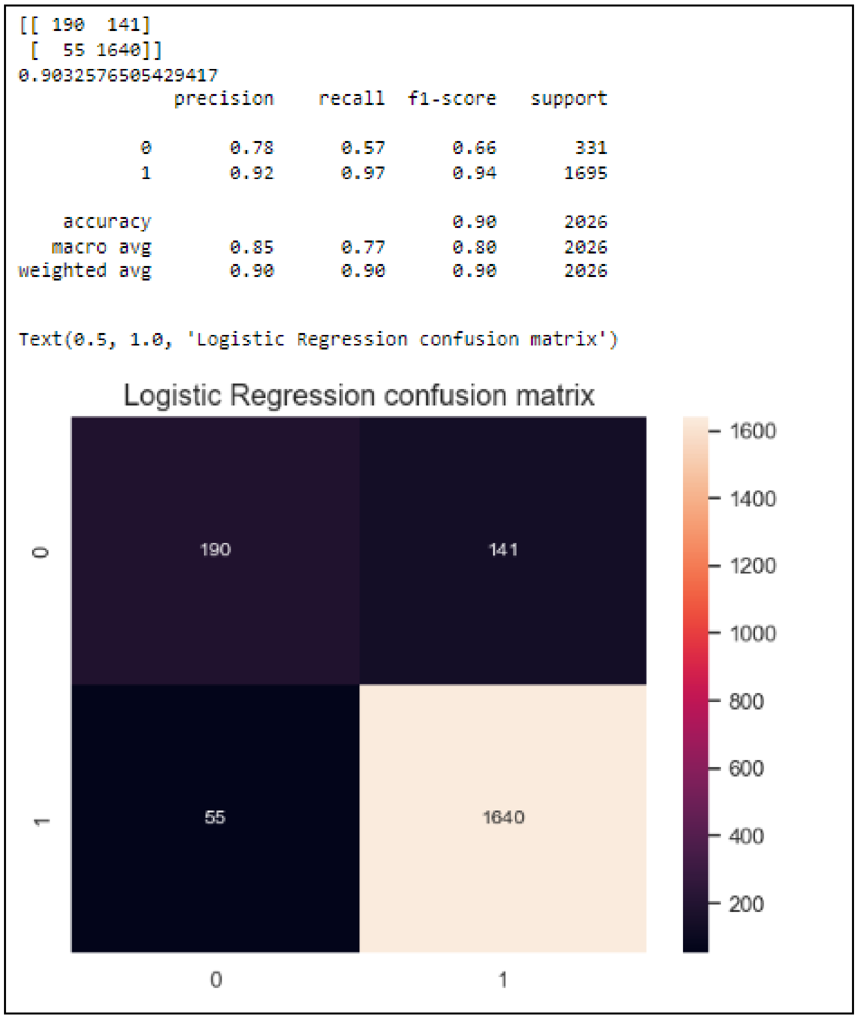
# CHAPTER THREE: Task 3

Additional data and analysis are needed in order to perform a thorough review, taking into account the unique requirements and limitations of the current business challenge.



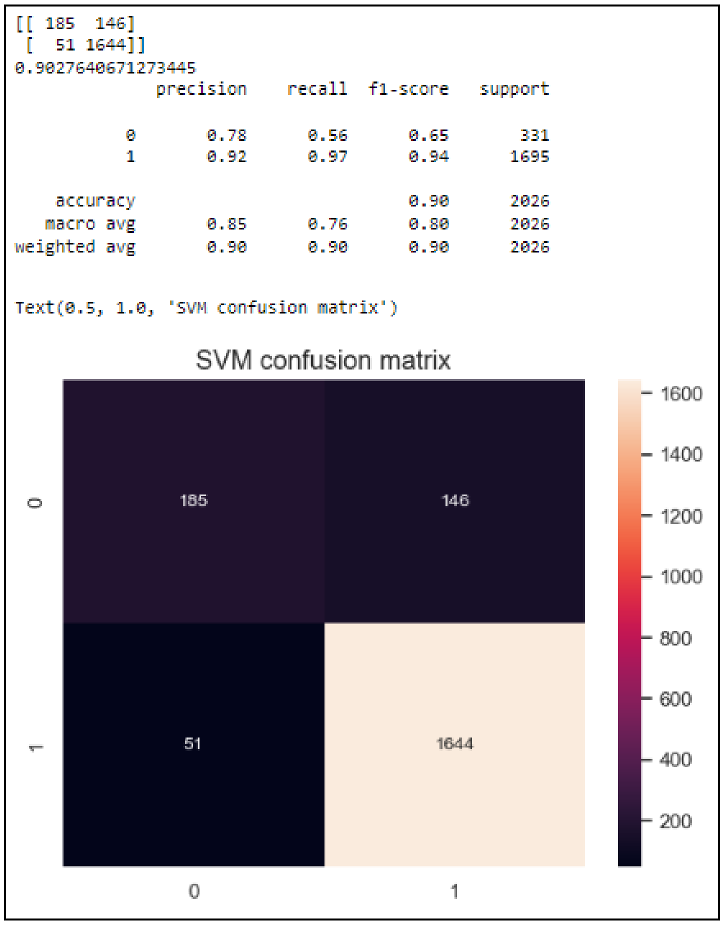
```
[[ 200  131]
 [  41 1654]]
0.9151036525172754
              precision    recall  f1-score   support

           0       0.83      0.60      0.70       331
           1       0.93      0.98      0.95      1695

    accuracy                           0.92      2026
   macro avg       0.88      0.79      0.82      2026
weighted avg       0.91      0.92      0.91      2026


: Text(0.5, 1.0, 'knn confusion matrix')
```

**Figure 17: KNN model evaluation**



```
[[ 190  141]
 [  55 1640]]
0.9032576505429417
              precision    recall  f1-score   support

           0       0.78      0.57      0.66       331
           1       0.92      0.97      0.94      1695

    accuracy                           0.90      2026
   macro avg       0.85      0.77      0.80      2026
weighted avg       0.90      0.90      0.90      2026


Text(0.5, 1.0, 'Logistic Regression confusion matrix')
```

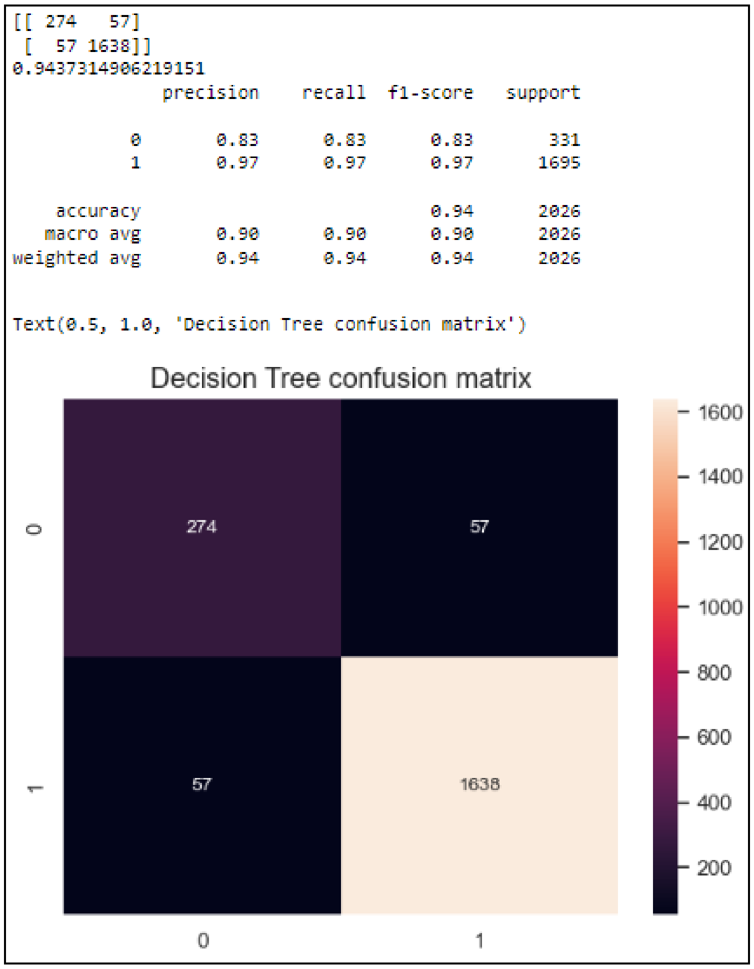
**Figure 18: Logistic regression model evaluation**



```
[[ 185  146]
 [  51 1644]]
0.9027640671273445
              precision    recall  f1-score   support

           0       0.78      0.56      0.65       331
           1       0.92      0.97      0.94      1695

    accuracy                           0.90      2026
   macro avg       0.85      0.76      0.80      2026
weighted avg       0.90      0.90      0.90      2026


Text(0.5, 1.0, 'SVM confusion matrix')
```

**Figure 19: SVM model evaluation**

In conclusion, assessing an algorithm entails taking into account a number of variables, including accuracy, scalability, interpretability, and other pertinent elements. Insights into the precision of several models are presented in the table, although scalability or interpretability are not specifically addressed (Sinha *et al*., 2019).



```
[[ 274   57]
 [  57 1638]]
0.9437314906219151
              precision    recall  f1-score   support

           0       0.83      0.83      0.83       331
           1       0.97      0.97      0.97      1695

    accuracy                           0.94      2026
   macro avg       0.90      0.90      0.90      2026
weighted avg       0.94      0.94      0.94      2026


Text(0.5, 1.0, 'Decision Tree confusion matrix')
```

**Figure 20: Decision Tree model evaluation**

Beyond accuracy, scalability, and interpretability, it is crucial to take other aspects into account while evaluating an algorithm (Cybulski and Scheepers, 2021). These might include things like computing effectiveness, resistance to noise or outliers, generalizability to unobserved data, and possibly ethical ramifications. To fully comprehend the algorithm's advantages and disadvantages, it is crucial to contrast the algorithm's performance with appropriate baselines or benchmarks.

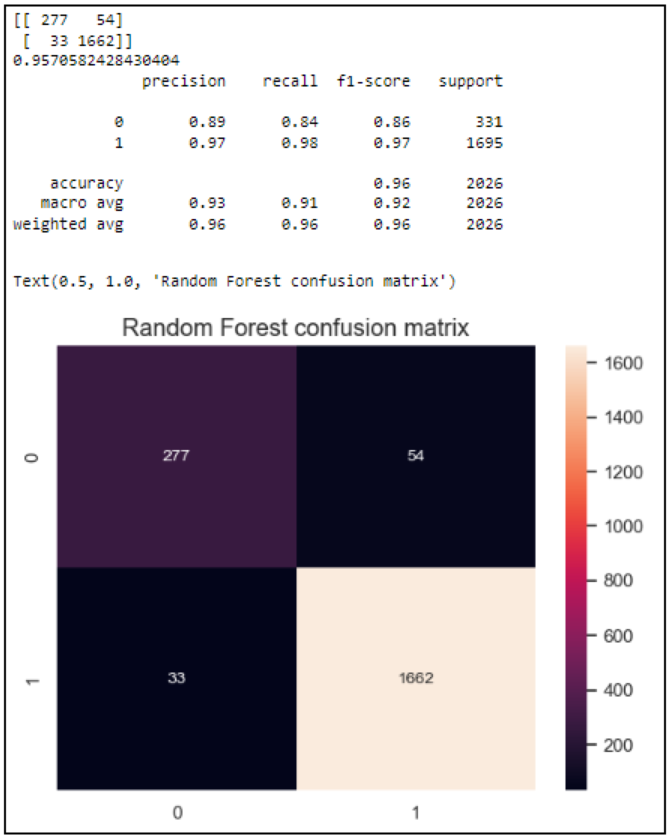
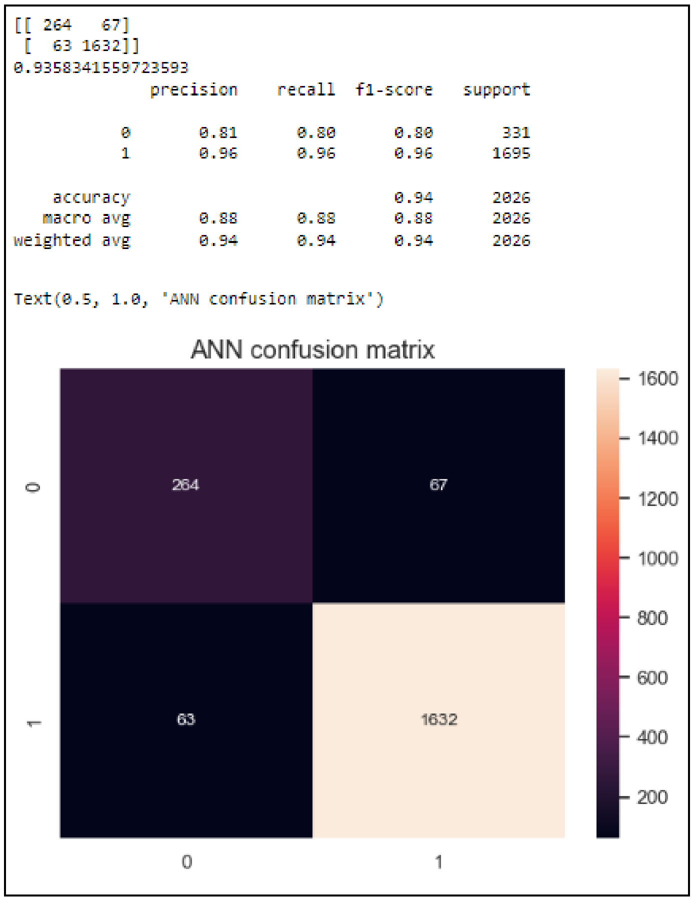

```
[[ 277   54]
 [  33 1662]]
0.9570582428430404
              precision    recall  f1-score   support

           0       0.89      0.84      0.86       331
           1       0.97      0.98      0.97      1695

    accuracy                           0.96      2026
   macro avg       0.93      0.91      0.92      2026
weighted avg       0.96      0.96      0.96      2026


Text(0.5, 1.0, 'Random Forest confusion matrix')
```

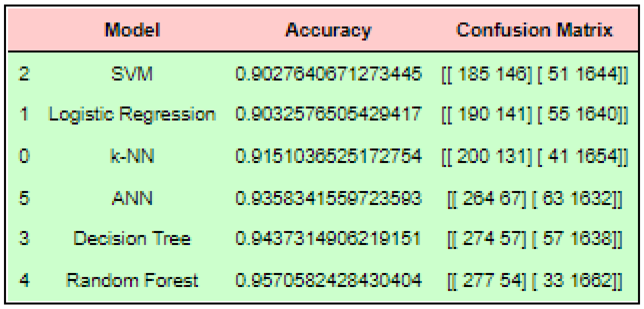
**Figure 21: Random Forest model evaluation**

The capacity of an algorithm to accommodate larger datasets or rising computing demands is referred to as scalability (Acciarini *et al*., 2023). When working with real-world business circumstances, where data quantities are frequently sizable, it is crucial to take scalability into consideration. Although scalability is not expressly mentioned in the chart, it is a crucial factor to take into account when selecting an algorithm for implementation in real-world settings. The capacity to comprehend and interpret the assumptions that underlie a model's predictions is referred to as interpretability. It is crucial in situations when decisions made using the model's output must be explicit and understandable. The interpretability of the models described in the above table varies (Park and Song, 2020).



```
[[ 264   67]
 [  63 1632]]
0.9358341559723593
              precision    recall  f1-score   support

           0       0.81      0.80      0.80       331
           1       0.96      0.96      0.96      1695

    accuracy                           0.94      2026
   macro avg       0.88      0.88      0.88      2026
weighted avg       0.94      0.94      0.94      2026

Text(0.5, 1.0, 'ANN confusion matrix')
```

**Figure 22: ANN model evaluation**



| | Model | Accuracy | Confusion Matrix |
|---|---|---|---|
| 2 | SVM | 0.9027640671273445 | [[ 185 146] [ 51 1644]] |
| 1 | Logistic Regression | 0.9032576505429417 | [[ 190 141] [ 55 1640]] |
| 0 | k-NN | 0.9151036525172754 | [[ 200 131] [ 41 1654]] |
| 5 | ANN | 0.9358341559723593 | [[ 264 67] [ 63 1632]] |
| 3 | Decision Tree | 0.9437314906219151 | [[ 274 57] [ 57 1638]] |
| 4 | Random Forest | 0.9570582428430404 | [[ 277 54] [ 33 1662]] |

**Figure 23: Model comparison**

Evaluation of an algorithm's performance is essential in the fields of predictive analytics and machine learning in order to determine its efficacy and applicability for solving actual business challenges. The review process must take into account a number of important variables, such as accuracy, scalability, and interpretability.

A crucial parameter for evaluating the effectiveness of a prediction model is accuracy. It stands for the model's capacity for accurate prediction (Ortigosa *et al*., 2019). In this instance, the table lists the accuracy ratings for several models. Random Forest, which has a rating of 0.957, is the model with the best accuracy. This shows that the Random Forest model is highly accurate in accurately classifying occurrences based on the dataset supplied.

## CONCLUSION

In conclusion, assessing the efficacy of prediction algorithms is essential for applying machine learning and artificial intelligence to real-world business challenges. The accuracy values for several models were displayed in the table that was supplied, demonstrating their propensity to produce accurate forecasts. However, in order to conduct a thorough analysis, extra aspects beyond accuracy must be taken into account. Although it isn't included in the chart, scalability is a key factor in handling enormous datasets and processing needs. For an algorithm to be used effectively in real-world circumstances, it must be able to scale well. It guarantees that the method can manage growing data quantities and computing needs without sacrificing performance (Brunk *et al*., 2021).

Understanding the logic behind a model's predictions is referred to as interpretability, another crucial consideration. In situations where stakeholders must have faith in and understand the model's output, transparent and explicable decision-making is crucial. It is important to note that some models, such logistic regression and decision trees, are widely thought to be more interpretable than sophisticated models like artificial neural networks, even if the table does not explicitly state what interpretability means. When evaluating an algorithm, it is important to take into account elements including computing efficiency, resilience, generalizability, and ethical implications in addition to accuracy, scalability, and interpretability. It is crucial to evaluate the algorithm's performance under different circumstances, its capacity for handling noisy or outlier data, its ability to generalise to new situations, and any potential ethical issues that may emerge.

## REFERENCES

Acciarini, C., Cappa, F., Boccardelli, P. and Oriani, R., 2023. How can organizations leverage big data to innovate their business models? A systematic literature review. Technovation, 123, p.102713.

Acciarini, C., Cappa, F., Boccardelli, P. and Oriani, R., 2023. How can organizations leverage big data to innovate their business models? A systematic literature review. *Technovation*, *123*, p.102713.

Akter, S., Michael, K., Uddin, M.R., McCarthy, G. and Rahman, M., 2022. Transforming business using digital innovations: The application of AI, blockchain, cloud and data analytics. Annals of Operations Research, pp.1-33.

Blank, J. and Deb, K., 2020. Pymoo: Multi-objective optimization in python. IEEE Access, 8, pp.89497-89509.

Brownlee, J., 2020. Data preparation for machine learning: data cleaning, feature selection, and data transforms in Python. Machine Learning Mastery.

Brunk, J., Stierle, M., Papke, L., Revoredo, K., Matzner, M. and Becker, J., 2021. Cause vs. effect in context-sensitive prediction of business process instances. *Information systems*, *95*, p.101635.

Choi, J., Yoon, J., Chung, J., Coh, B.Y. and Lee, J.M., 2020. Social media analytics and business intelligence research: A systematic review. Information Processing & Management, 57(6), p.102279.

Cybulski, J.L. and Scheepers, R., 2021. Data science in organizations: Conceptualizing its breakthroughs and blind spots. *Journal of Information Technology*, *36*(2), pp.154-175.

Doleck, T., Lemay, D.J., Basnet, R.B. and Bazelais, P., 2020. Predictive analytics in education: a comparison of deep learning frameworks. Education and Information Technologies, 25, pp.1951-1963.

Doran, M., 2022. A VIRTUAL EDUCATION INTERVENTION TO APPROXIMATE HANDS-ON-LEARNING: VIA TASK-CENTERED LEARNING PRAXIS.

Fathi, M., Haghi Kashani, M., Jameii, S.M. and Mahdipour, E., 2022. Big data analytics in weather forecasting: A systematic review. Archives of Computational Methods in Engineering, 29(2), pp.1247-1275.

Gambhir, E., Jain, R., Gupta, A. and Tomer, U., 2020, September. Regression analysis of COVID-19 using machine learning algorithms. In 2020 International conference on smart electronics and communication (ICOSEC) (pp. 65-71). IEEE.

Kavitha, M., Gnaneswar, G., Dinesh, R., Sai, Y.R. and Suraj, R.S., 2021, January. Heart disease prediction using hybrid machine learning model. In 2021 6th international conference on inventive computation technologies (ICICT) (pp. 1329-1333). IEEE.

Loy, J., 2019. Neural Network Projects with Python: The ultimate guide to using Python to explore the true power of neural networks through six projects. Packt Publishing Ltd.

Namkung, J., 2020. Machine learning methods for microbiome studies. Journal of Microbiology, 58, pp.206-216.

Nielsen, A., 2019. Practical time series analysis: Prediction with statistics and machine learning. O'Reilly Media.

Ortigosa, A., Carro, R.M., Bravo-Agapito, J., Lizcano, D., Alcolea, J.J. and Blanco, O., 2019. From lab to production: Lessons learnt and real-life challenges of an early student-dropout prevention system. *IEEE Transactions on Learning Technologies*, *12*(2), pp.264-277.

Park, G. and Song, M., 2020. Predicting performances in business processes using deep neural networks. *Decision Support Systems*, *129*, p.113191.

Prakash, D.C., Narayanan, R.C., Ganesh, N., Ramachandran, M., Chinnasami, S. and Rajeshwari, R., 2022, May. A study on image processing with data analysis. In AIP Conference Proceedings (Vol. 2393, No. 1, p. 020225). AIP Publishing LLC.

Shravya, C., Pravalika, K. and Subhani, S., 2019. Prediction of breast cancer using supervised machine learning techniques. International Journal of Innovative Technology and Exploring Engineering (IJITEE), 8(6), pp.1106-1110.

Singh, N., Chaturvedi, S. and Akhter, S., 2019, March. Weather forecasting using machine learning algorithm. In 2019 International Conference on Signal Processing and Communication (ICSC) (pp. 171-174). IEEE.

Sinha, T., Kapur, M., West, R., Catasta, M., Hauswirth, M. and Trninic, D., 2019. Impact of Explicit Failure and Success-driven Preparatory Activities on Learning. In *CogSci* (pp. 2804-2810).