Financial distress prediction for US hospitals with machine learning following KDD Methodology

MSc Research Project
MSc in Financial Technology

Manel Hmida
Student ID: 21213445

School of Computing
National College of Ireland

Supervisor: Noel Cosgrave

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Manel Hmida		
Student ID:	x21213445		
Programme:	MSc in Financial Technology	**Year:**	……2022/2023.
Module:	Research project		
Lecturer:	Noel Cosgrave		
Submission Due Date:	14 August 2023		
Project Title:	…… Financial distress prediction for US hospitals with machine learning following KDD Methodology		
Word Count:	……1152………	**Page Count:**	9.

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: ……Manel Hmida

Date: ……14 August 2023

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Manel Hmida
Student ID: x21213445

1 Introduction

The configuration manual is prepared to describe the technical steps conducted to answer the research question of the author's project which is about "Financial distress prediction for US hospitals using machine learning following KDD methodology".
The manual starts by specifying the hardware configuration used to accomplish the project's objectives. Next, the author displays the software that has been used to process and the train the machine learning models on the data. The manual walks through all the steps the author have done to reach the findings.

2 Hardware configuration

The author used a laptop with the following characteristics that contributed to the successful implementation of the methodology of the project:

Table 1 Hardware configuration

Processor	11th Gen Intel(R) Core (TM) i7-1165G7 @ 2.80GHz 2.80 GHz
Ram	12.0 GB
System Type	64-bit operating system, x64-based processor
Operating system	Windows 11 Home

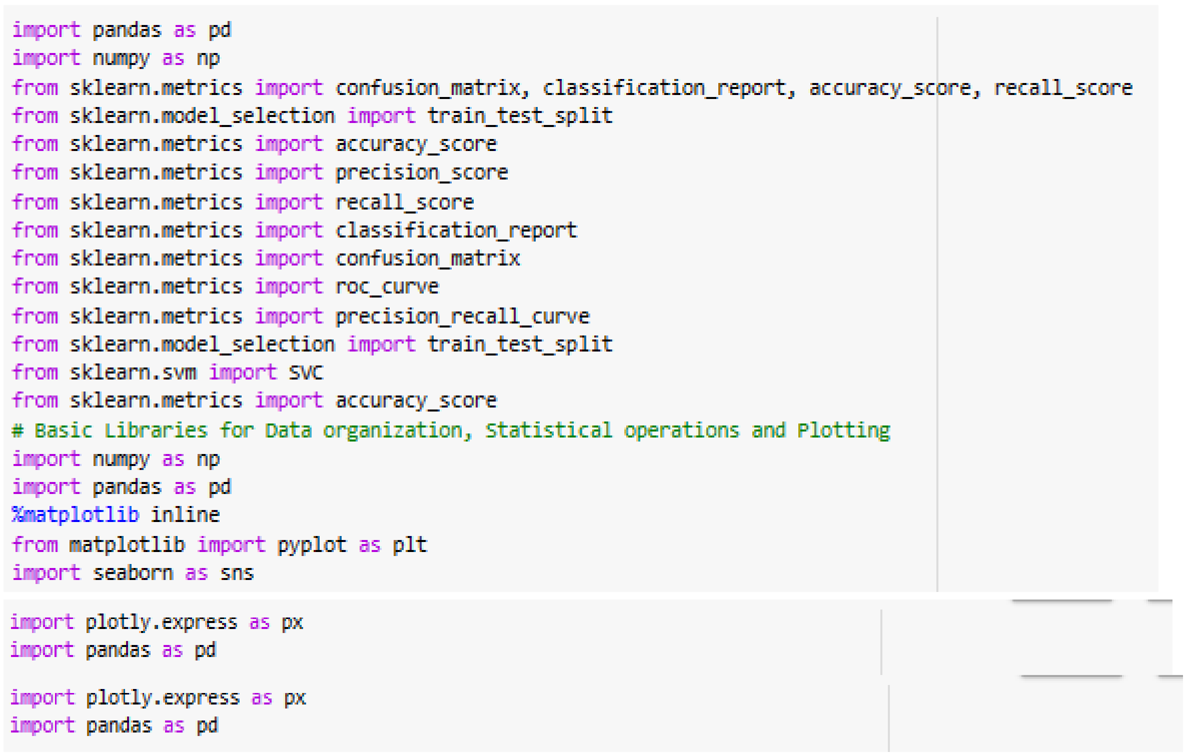
3 Software configuration

This project utilized Python programming language and ran on Colab. A notebook of the codes is attached for testing.

4 Walk through project's codes.

This section is dedicated to showcase and explain the steps initiated by the author to achieve the projects objectives.

4.1 Libraries



```
import pandas as pd
import numpy as np
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score, recall_score
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import roc_curve
from sklearn.metrics import precision_recall_curve
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score
# Basic Libraries for Data organization, Statistical operations and Plotting
import numpy as np
import pandas as pd
%matplotlib inline
from matplotlib import pyplot as plt
import seaborn as sns
```

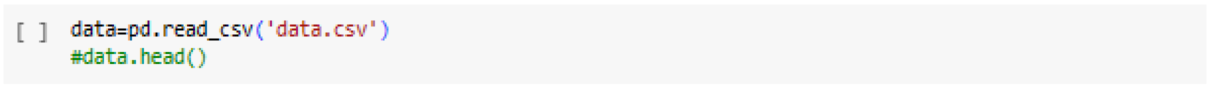
```
import plotly.express as px
import pandas as pd

import plotly.express as px
import pandas as pd
```

Figure 1 Imported libraries.

For reading, cleaning, processing and modelling the data several libararies need to be imported. The above screenshot showcases the liabraries that have been imported from Python that helped visualize and model the data.

4.2 Data loading

The data set csv file was uploaded into Colab and the following lines of codes enable the reading and display of the loaded file:



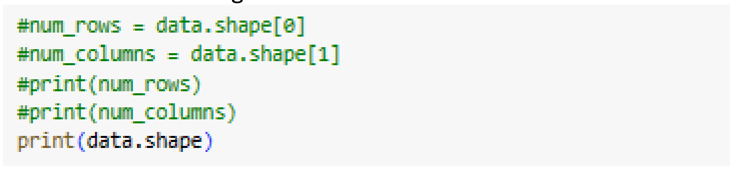
```
[ ]  data=pd.read_csv('data.csv')
     #data.head()
```

Figure 2 Data loading

4.3 Data preprocessing and transformation

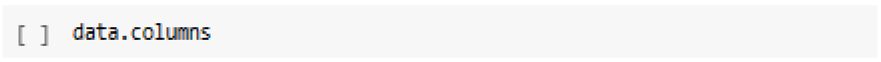
This section is dedicated to display all the steps taken to first comprehend the data (shape, structure, data distribution) and then its transformation.

- The author started by understanding the structure of the data, the following code enable the count of rows and columns besides knowing the title of the columns:



```
#num_rows = data.shape[0]
#num_columns = data.shape[1]
#print(num_rows)
#print(num_columns)
print(data.shape)
```

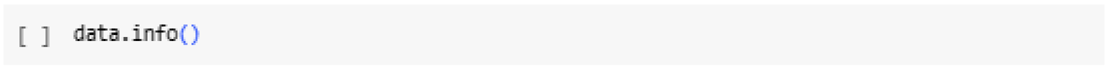
Figure 3 Data shape



```
[ ]  data.columns
```

Figure 4 Name of the columns

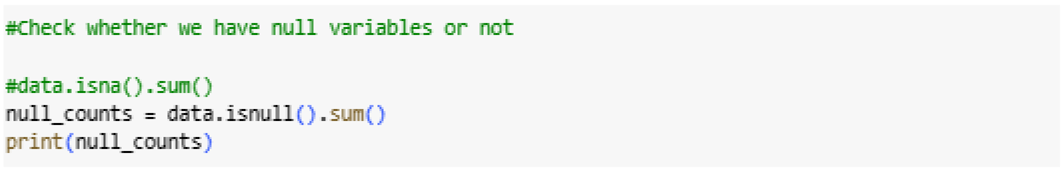
- It is essential to underscore the type of the data the author is dealing with in order to anticipate transformation actions afterwards. Data info displays the type of data for each column:



```
[ ]  data.info()
```

Figure 5 Data type

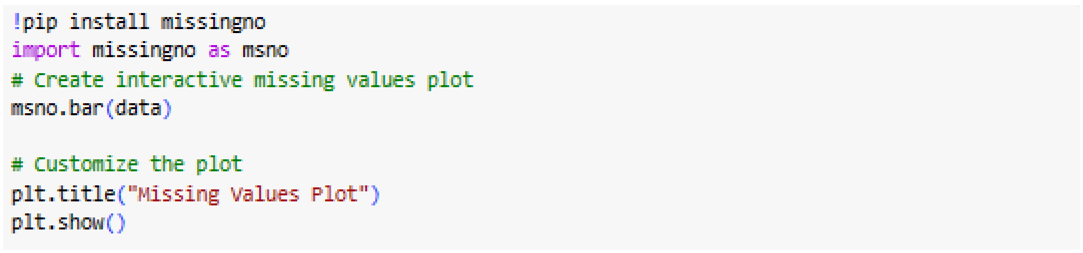
- Checking for missing values is a crucial step that clarify the state of the data:



```
#Check whether we have null variables or not

#data.isna().sum()
null_counts = data.isnull().sum()
print(null_counts)
```

Figure 6 Check up for missing values.

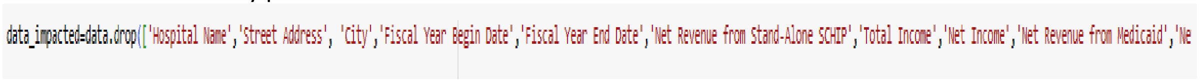
- Visualization is a strong tool for data analytics; therefore, the author used the following codes to graphically visualize the missing values in the data:



```
!pip install missingno
import missingno as msno
# Create interactive missing values plot
msno.bar(data)

# Customize the plot
plt.title("Missing Values Plot")
plt.show()
```

Figure 7 Missing values plot.

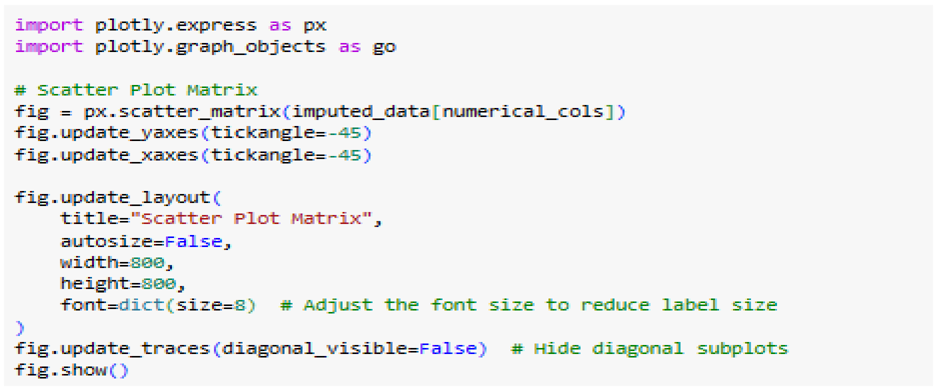
- Based on the above codes, the decision of removing irrelevant columns where fortunately the missing values were mostly present:



```
data_impacted=data.drop(['Hospital Name','Street Address', 'City','Fiscal Year Begin Date','Fiscal Year End Date','Net Revenue from Stand-Alone SCHIP','Total Income','Net Income','Net Revenue from Medicaid','Ne
```

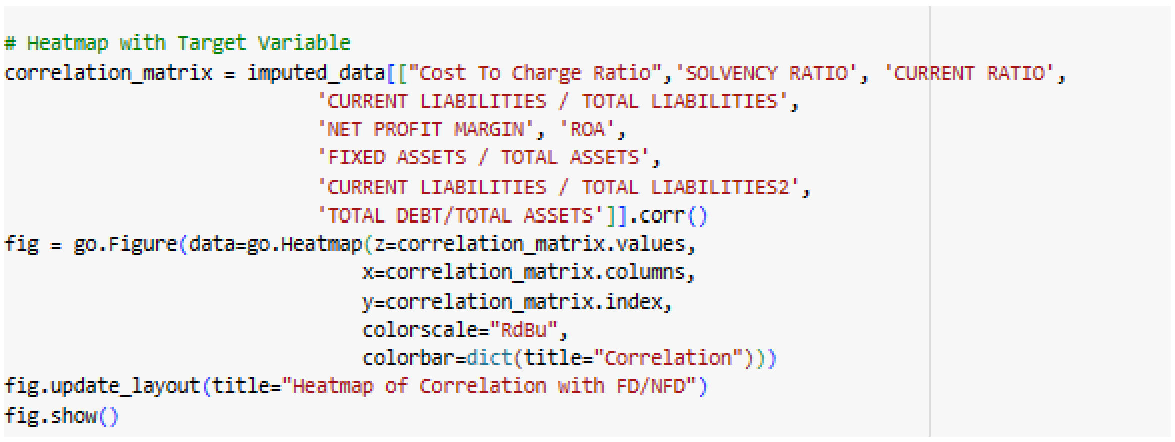
‘hospital Name’, ‘Street Address', 'City', 'Fiscal Year Begin Date’, ‘Fiscal Year End Date', 'Net Revenue from Stand-Alone SCHIP’, ‘Total Income', 'Net Income', 'Net Revenue from Medicaid’, ‘Net Revenue from Stand-Alone SCHIP’, ‘net revenue', 'TOTAL DEBT','OPERATING INCOME': are the columns that are not relevant for modeling as the author needs the nine financial ratios and type of control feature.

- Other plots were generated to visualize the data distribution and degree of correlation between the features:



```
import plotly.express as px
import plotly.graph_objects as go

# Scatter Plot Matrix
fig = px.scatter_matrix(imputed_data[numerical_cols])
fig.update_yaxes(tickangle=-45)
fig.update_xaxes(tickangle=-45)

fig.update_layout(
    title="Scatter Plot Matrix",
    autosize=False,
    width=800,
    height=800,
    font=dict(size=8)  # Adjust the font size to reduce label size
)
fig.update_traces(diagonal_visible=False)  # Hide diagonal subplots
fig.show()
```

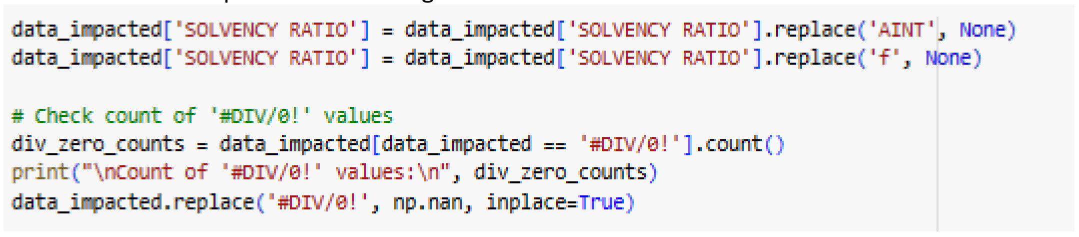
Figure 8 Data distribution



```
# Heatmap with Target Variable
correlation_matrix = imputed_data[["Cost To Charge Ratio",'SOLVENCY RATIO', 'CURRENT RATIO',
                                  'CURRENT LIABILITIES / TOTAL LIABILITIES',
                                  'NET PROFIT MARGIN', 'ROA',
                                  'FIXED ASSETS / TOTAL ASSETS',
                                  'CURRENT LIABILITIES / TOTAL LIABILITIES2',
                                  'TOTAL DEBT/TOTAL ASSETS']].corr()
fig = go.Figure(data=go.Heatmap(z=correlation_matrix.values,
                               x=correlation_matrix.columns,
                               y=correlation_matrix.index,
                               colorscale="RdBu",
                               colorbar=dict(title="Correlation")))
fig.update_layout(title="Heatmap of Correlation with FD/NFD")
fig.show()
```

Figure 9 Correlation map

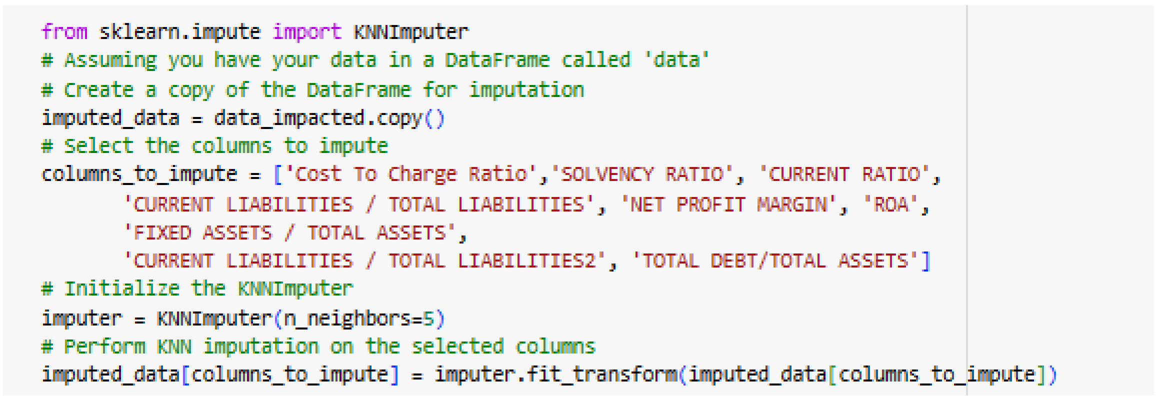
- The financial ratios showed different types of missing values like ‘#DIV/0’, ‘AINT’ therefore the author used this code to replace those missing values with nan in order to deal with them later.



```
data_impacted['SOLVENCY RATIO'] = data_impacted['SOLVENCY RATIO'].replace('AINT', None)
data_impacted['SOLVENCY RATIO'] = data_impacted['SOLVENCY RATIO'].replace('f', None)

# Check count of '#DIV/0!' values
div_zero_counts = data_impacted[data_impacted == '#DIV/0!'].count()
print("\nCount of '#DIV/0!' values:\n", div_zero_counts)
data_impacted.replace('#DIV/0!', np.nan, inplace=True)
```

Figure 10 Transformation of missing values

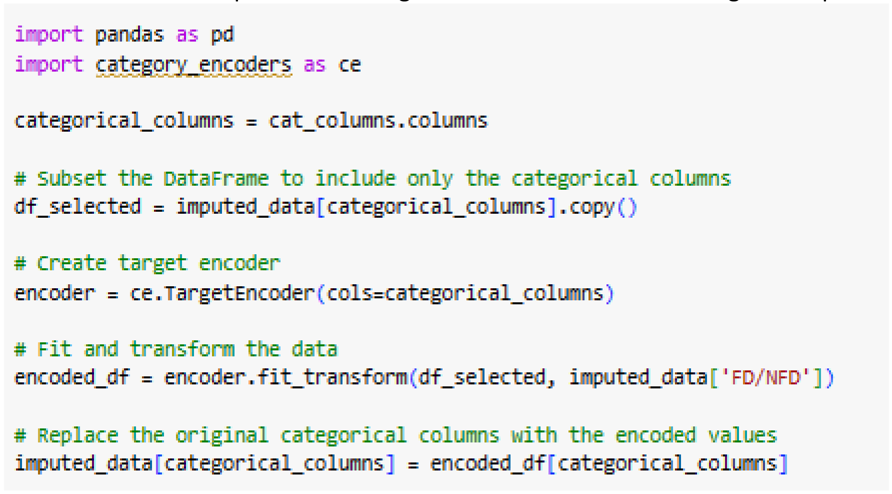
- The missing values for the ratio were then replaced using KNN imputation as the author believes it is the suitable method to keep more observations for modeling and KNN can achieve that:



```
from sklearn.impute import KNNImputer
# Assuming you have your data in a DataFrame called 'data'
# Create a copy of the DataFrame for imputation
imputed_data = data_impacted.copy()
# Select the columns to impute
columns_to_impute = ['Cost To Charge Ratio','SOLVENCY RATIO', 'CURRENT RATIO',
       'CURRENT LIABILITIES / TOTAL LIABILITIES', 'NET PROFIT MARGIN', 'ROA',
       'FIXED ASSETS / TOTAL ASSETS',
       'CURRENT LIABILITIES / TOTAL LIABILITIES2', 'TOTAL DEBT/TOTAL ASSETS']
# Initialize the KNNImputer
imputer = KNNImputer(n_neighbors=5)
# Perform KNN imputation on the selected columns
imputed_data[columns_to_impute] = imputer.fit_transform(imputed_data[columns_to_impute])
```

Figure 11 KNN imputation

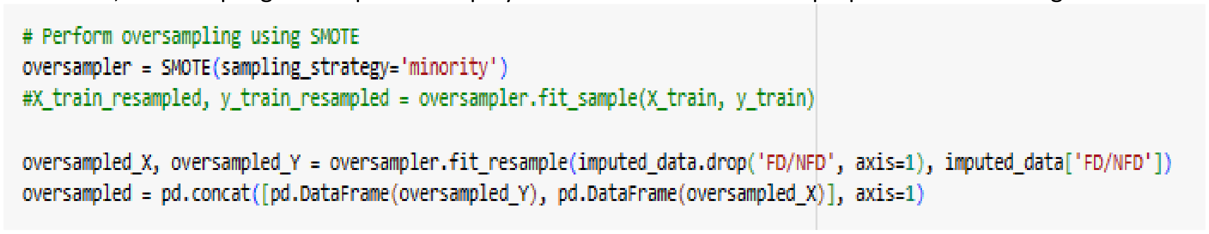
- 'Type of control' feature represents a categorical data therefore encoding was implemented:



```
import pandas as pd
import category_encoders as ce

categorical_columns = cat_columns.columns

# Subset the DataFrame to include only the categorical columns
df_selected = imputed_data[categorical_columns].copy()

# Create target encoder
encoder = ce.TargetEncoder(cols=categorical_columns)

# Fit and transform the data
encoded_df = encoder.fit_transform(df_selected, imputed_data['FD/NFD'])

# Replace the original categorical columns with the encoded values
imputed_data[categorical_columns] = encoded_df[categorical_columns]
```

Figure 12 Categorical data encoding

- The author is dealing with imbalanced data which can represent a problem of bias in models. As a result, oversampling technique was deployed to balance the data and prepare it for modeling:

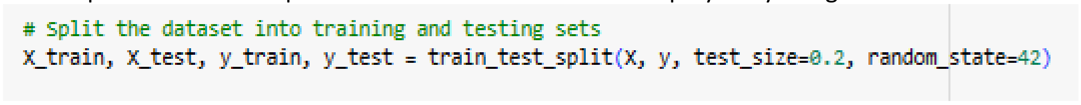


```
# Perform oversampling using SMOTE
oversampler = SMOTE(sampling_strategy='minority')
#X_train_resampled, y_train_resampled = oversampler.fit_sample(X_train, y_train)

oversampled_X, oversampled_Y = oversampler.fit_resample(imputed_data.drop('FD/NFD', axis=1), imputed_data['FD/NFD'])
oversampled = pd.concat([pd.DataFrame(oversampled_Y), pd.DataFrame(oversampled_X)], axis=1)
```

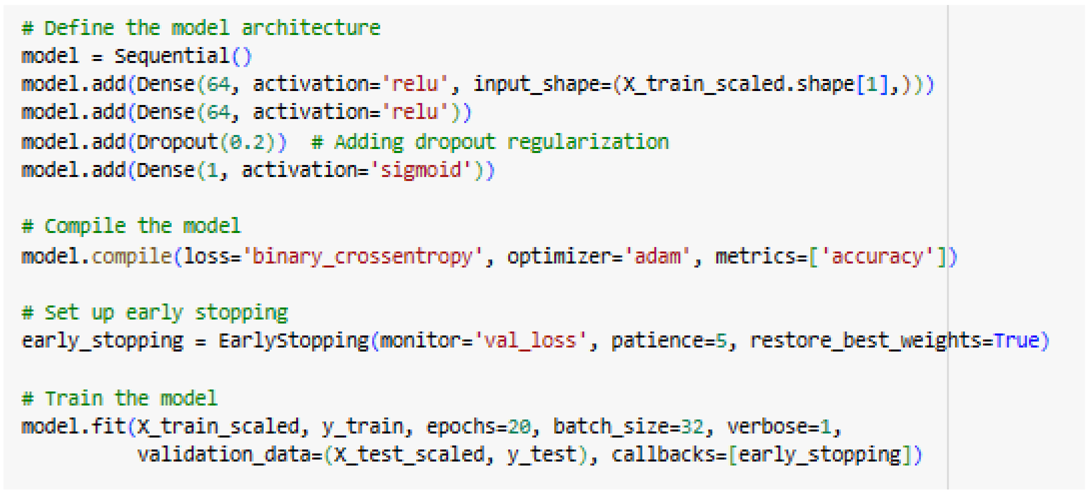
Figure 13 Oversampling using SMOTE.

4.4 Implementation of the chosen models

As thoroughly explained in the author's report, three models are chosen to be pursued for the financial distress prediction. The implementation of the models was deployed by using cross validation:

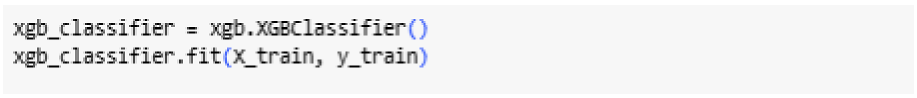


```
# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```



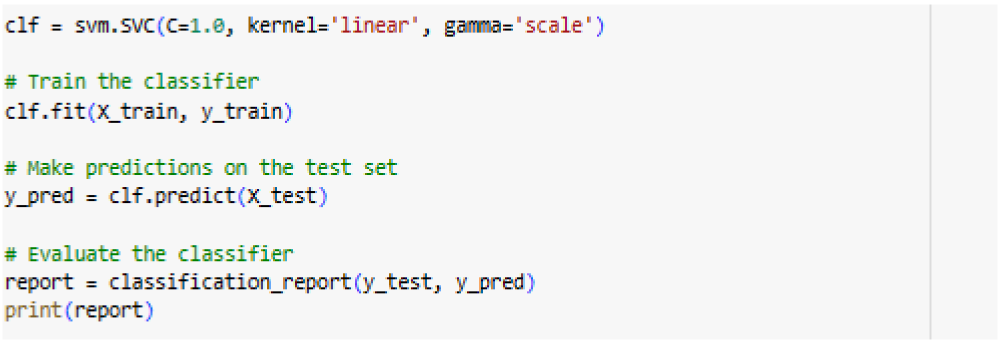
```python
# Define the model architecture
model = Sequential()
model.add(Dense(64, activation='relu', input_shape=(X_train_scaled.shape[1],)))
model.add(Dense(64, activation='relu'))
model.add(Dropout(0.2))  # Adding dropout regularization
model.add(Dense(1, activation='sigmoid'))

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

# Set up early stopping
early_stopping = EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=True)

# Train the model
model.fit(X_train_scaled, y_train, epochs=20, batch_size=32, verbose=1,
          validation_data=(X_test_scaled, y_test), callbacks=[early_stopping])
```

Figure 14 DNN implementation



```python
xgb_classifier = xgb.XGBClassifier()
xgb_classifier.fit(X_train, y_train)
```

Figure 15 Boost implementation

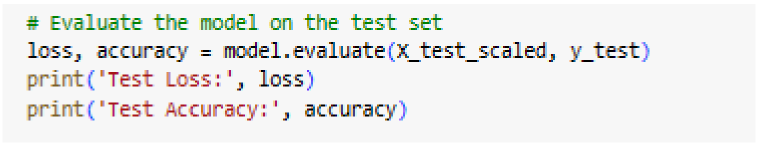


```python
clf = svm.SVC(C=1.0, kernel='linear', gamma='scale')

# Train the classifier
clf.fit(X_train, y_train)

# Make predictions on the test set
y_pred = clf.predict(X_test)

# Evaluate the classifier
report = classification_report(y_test, y_pred)
print(report)
```

Figure 16 SVM implementation

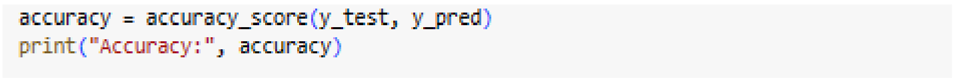
4.5 Evaluation

For performance evaluation, the author used different performance metrics:



```python
# Evaluate the model on the test set
loss, accuracy = model.evaluate(X_test_scaled, y_test)
print('Test Loss:', loss)
print('Test Accuracy:', accuracy)
```

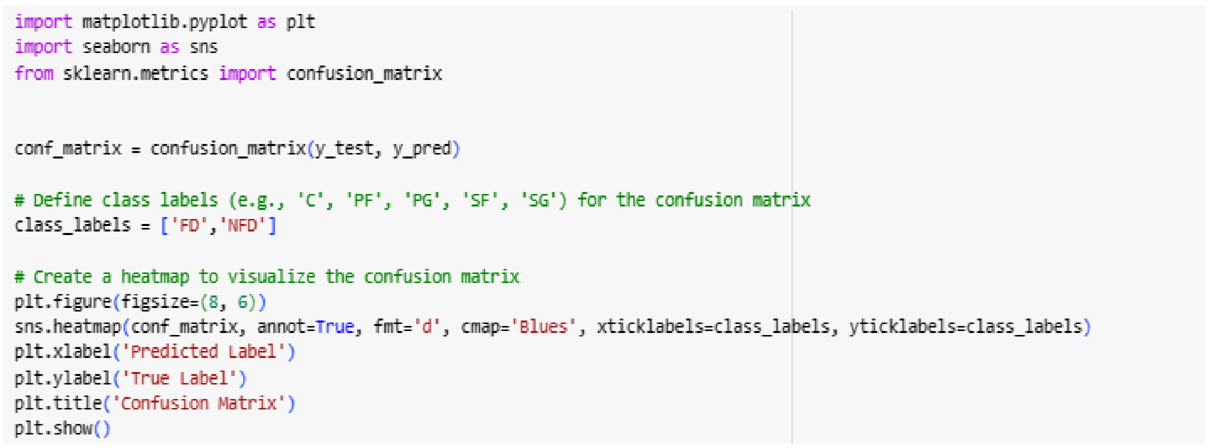
Figure 17 DNN evaluation metrics



```python
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

Figure 18 Boost results

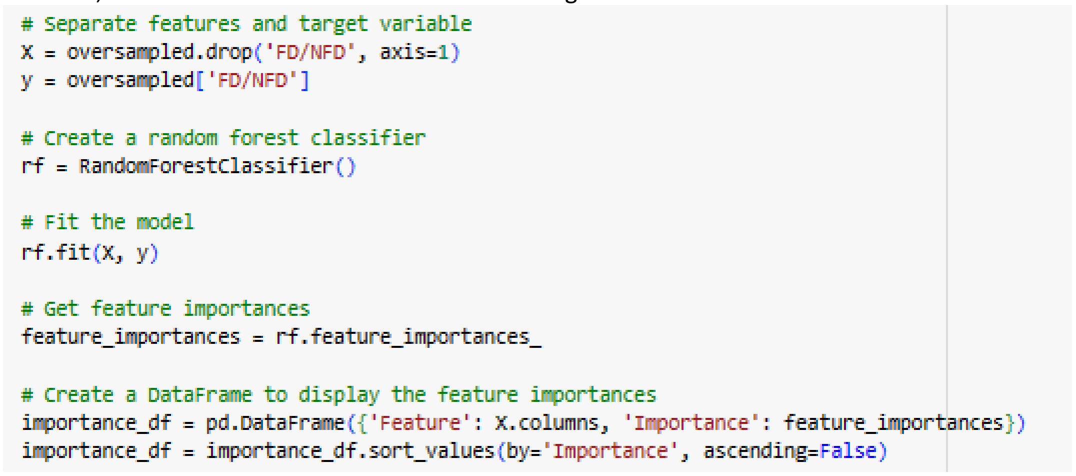
- For further understanding of the results, the author generated a plot for confusion matrix of XGboost using the following codes:



```python
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import confusion_matrix


conf_matrix = confusion_matrix(y_test, y_pred)

# Define class labels (e.g., 'C', 'PF', 'PG', 'SF', 'SG') for the confusion matrix
class_labels = ['FD','NFD']

# Create a heatmap to visualize the confusion matrix
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues', xticklabels=class_labels, yticklabels=class_labels)
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.title('Confusion Matrix')
plt.show()
```

Figure 19 Confusion Matrix for XGBoost

- Second objective of the project is to identify which of the features are the most significant for prediction, random forest classifier was used to distinguish that:



```python
# Separate features and target variable
X = oversampled.drop('FD/NFD', axis=1)
y = oversampled['FD/NFD']

# Create a random forest classifier
rf = RandomForestClassifier()

# Fit the model
rf.fit(X, y)

# Get feature importances
feature_importances = rf.feature_importances_

# Create a DataFrame to display the feature importances
importance_df = pd.DataFrame({'Feature': X.columns, 'Importance': feature_importances})
importance_df = importance_df.sort_values(by='Importance', ascending=False)
```

Figure 20 Random Forest classifier for features importance

The results and the generated plots helped the author drawing conclusions on to what extent can machine learning models predict financial distress in the healthcare sector using the data set of US hospitals. The author ended up by comparing the performance of the three deployed models and identifying which one was the most suitable for the data. Besides, he succeeded in identifying the most important feature for the prediction. Overall, these codes helped building significant conclusions and answered to the research question.