National College of Ireland

Configuration Manual

MSc Research Project
MSc Data Analytics

Adebola Abdullahi-Attah
Student ID: X19119283

School of Computing
National College of Ireland

Supervisors: Dr Paul Stynes & Dr Pramod Pathak

National College of Ireland
MSc Project Submission Sheet
School of Computing

Student Name:	Adebola Abdullahi-Attah
Student ID:	X19119283
Programme:	MSc Data Analytics **Year:** 2020
Module:	MSc Research Project
Lecturer:	Dr Paul Stynes & Dr Pramod Pathak
Submission Due Date:	17th August 2020
Project Title:	A Novel Feature Based Ensemble Approach to Bankruptcy Detection
Word Count:	1215 **Page Count:** 14

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: AAA

Date: 15th August 2020

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

Adebola Abdullahi-Attah
Student ID: X19119283

1 Introduction

This manual shows the hardware and software requirements, Also the steps taken to accomplish the implementation of the research thesis on "A novel feature selection based ensemble approach to bankruptcy detection".

2 Hardware and Software Requirements

2.1 Hardware Description

The research implementation was carried out on a Hewlett-Packard (HP) laptop having this description.

- Operating system: Window 10 Home (2019)
- Processor: Intel ® Core ™ i7-8565U CPU @ 1.80ghz
- System Type: 64-bit operating System, x64-based processor
- Installed Memory (RAM): 8.00 GB

2.2 Software Description

The following software enabled the implementation

- Microsoft office: Excel, Word
- Web browser: Google Chrome, Microsoft Edge
- Programming Tool: Python version 3, Google Colaboratory (Cloud based Jupyter notebook environment)
- Drawing tool: Lucidchart

3 Methodology & Implementation

3.1 Data Collection and Preparation

Step 1

The first step is getting the Polish company bankruptcy dataset from the university of California Irvine (UCI) data repository as shown in figure 1.

UCI Machine Learning Repository
Center for Machine Learning and Intelligent Systems
About Citation Policy Donate a Data Set Contact
Search
Repository Web
View ALL Data Sets

Polish companies bankruptcy data Data Set
Download: Data Folder, Data Set Description

Abstract: The dataset is about bankruptcy prediction of Polish companies.The bankrupt companies were analyzed in the period 2000-2012, while the still operating companies were evaluated from 2007 to 2013.

Data Set Characteristics:	Multivariate	Number of Instances:	10503	Area:	Business
Attribute Characteristics:	Real	Number of Attributes:	64	Date Donated	2016-04-11
Associated Tasks:	Classification	Missing Values?	Yes	Number of Web Hits:	106064

Figure 1: Dataset from UCI

Step 2
The dataset contains five files which is downloaded in a folder named data as shown in figure 2.

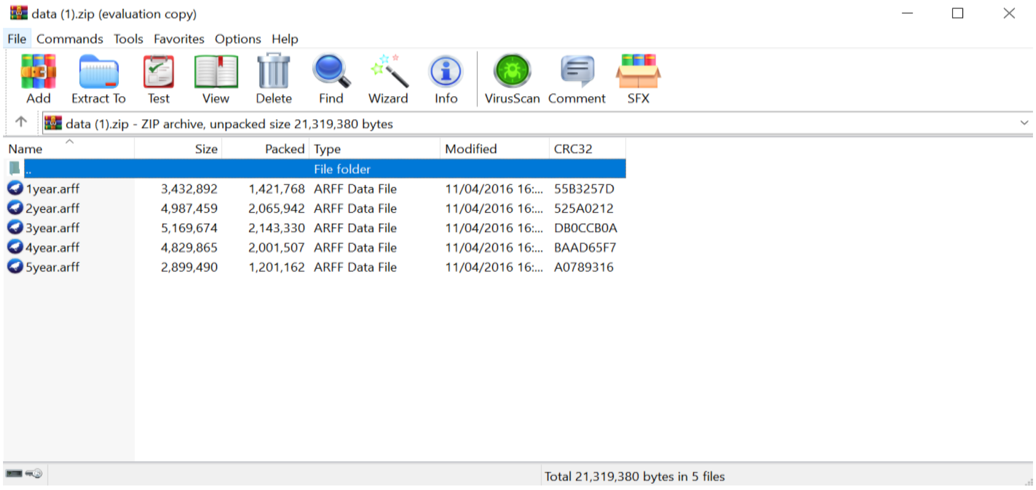

Figure 2: Dataset Files

Step 3

The folder is uploaded into an existing G-mail account drive (adebolaaattah@gmail.com) as shown in figure 3.

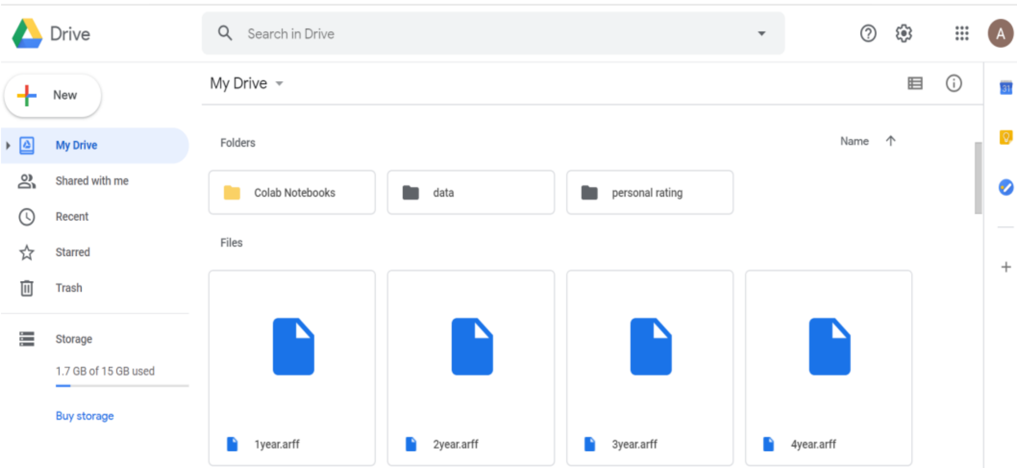

Figure 3: Upload data on drive

3.2 Google Colaboratory Environment Setup

The google colaboratory (colab) is set up for the smooth running of the python codes. An existing Google e-mail address is used as adebolaaattah@gmail.com. Figure 4 shows the setup environment.

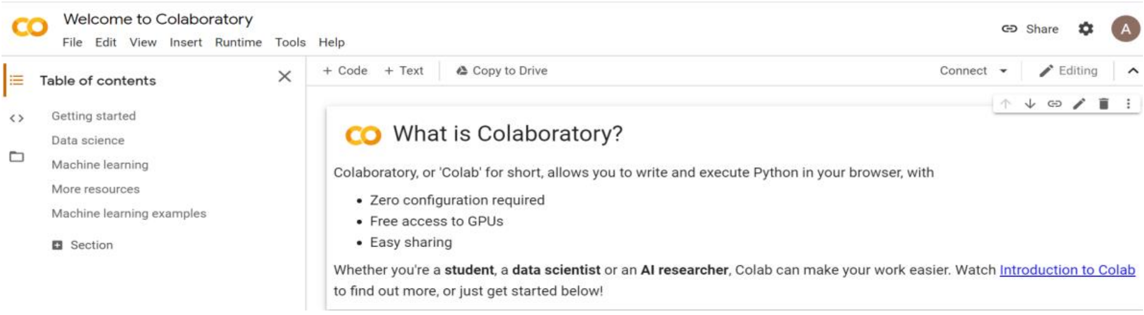

Figure 4: Signing into Google Colaboratory.

Step 1

Go to chrome browser and type in the address https://colab.research.google.com

Step 2

Go to file and open a new notebook to get started

Step 3

Mount google drive in colab Jupyter notebook and execute as shown in figure 5.

```
from google.colab import drive
drive.mount('/content/drive')
```

Figure 5: Mount google drive

Step 4

Click on the url as shown in figure 6

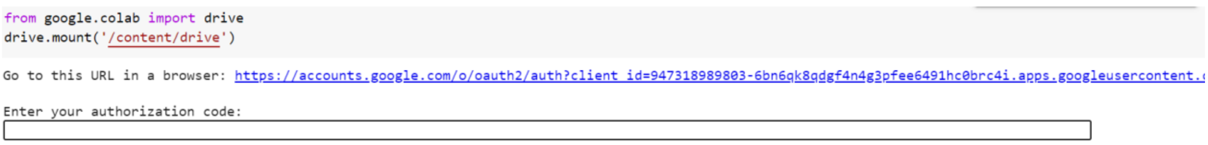

Figure 6: Acessing authorization code

Step 5

Select the Google account to be used for the colab set up

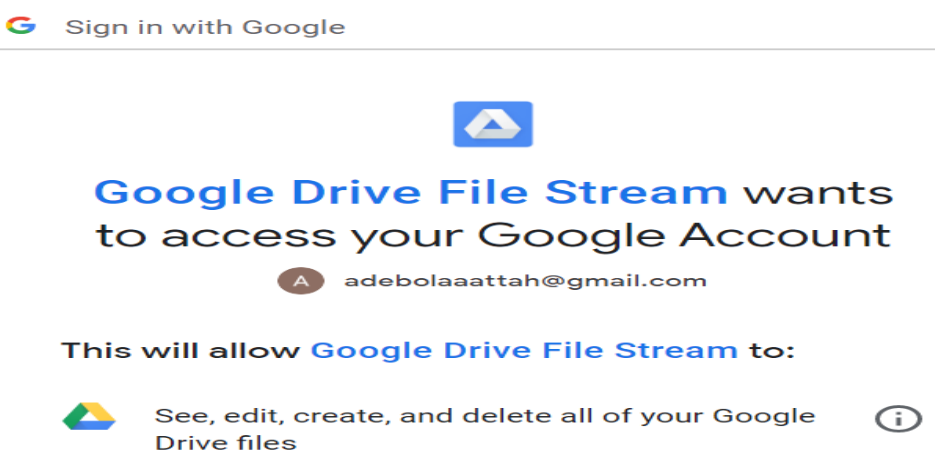

Figure 7: Selecting google account

Step 6
copy the authorization code and paste into the space provided on the notebook to enable the jupyter notebook access to the file location.

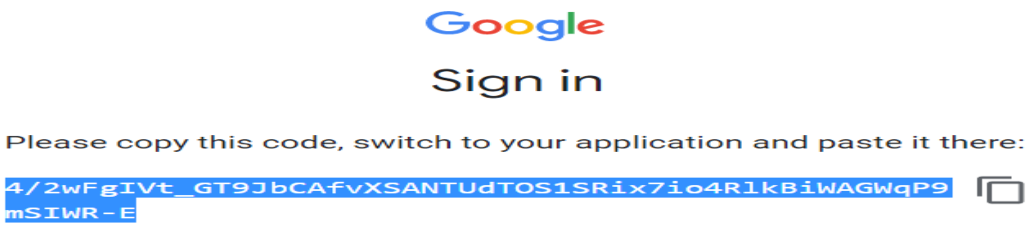

Figure 8: Copy authorization code

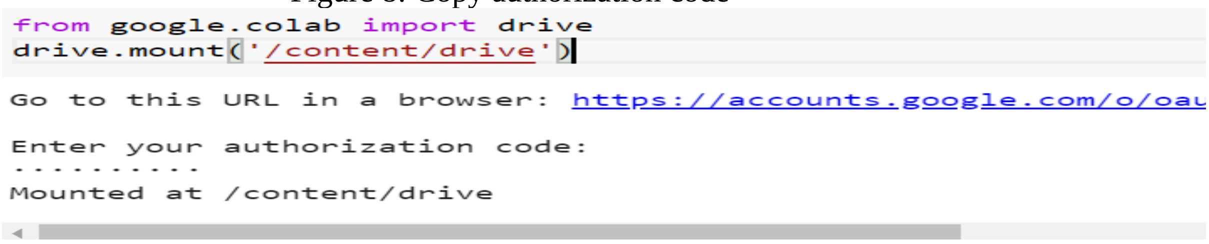

Figure 9: Drive mounted in Jupyter notebook

3.3 Importing Libraries

The libraries needed for the exploratory data analysis, imputation of missing values, graph plotting and statistical analysis, oversampling of the minority class, implementation of the particle swarm optimization which had to be installed as shown in the figure below.

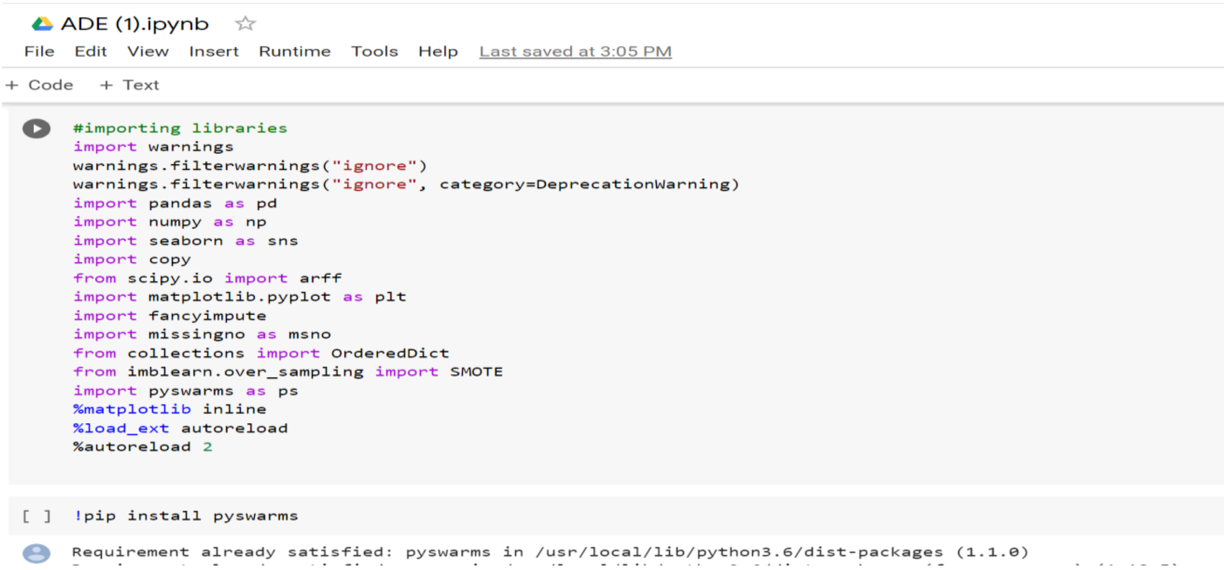

Figure 10: Importing Libraries

3.4 Accessing Data and Storing into list

ADE (1).ipynb ☆
File Edit View Insert Runtime Tools Help Last saved at 3:05 PM
+ Code + Text

Store arff files into a list

```
[ ] # Loads the 5 raw .arff files into a list
    def load_data():
        N=5
        return [arff.loadarff("/content/drive/My Drive/" + str(i+1) + 'year.arff') for i in range(N)]

    # store raw .arff files into dataframes
    def real_data():
        return [pd.DataFrame(data_i_year[0]) for data_i_year in load_data()]

    # BankDF is a list of 5 dataframes
    BankDf = real_data()

    # VIew the first 5 rows of the dataset 'year1'
    BankDf[0].head()
```

	Attr1	Attr2	Attr3	Attr4	Attr5	Attr6	Attr7	Attr8	Attr9	Attr10	Attr11	Attr12
0	0.200550	0.37951	0.39641	2.0472	32.3510	0.38825	0.249760	1.33050	1.1389	0.50494	0.249760	0.65980
1	0.209120	0.49988	0.47225	1.9447	14.7860	0.00000	0.258340	0.99601	1.6996	0.49788	0.261140	0.51680

Figure 11: Loading data

3.5 Imputing missing values using MICE

The missing values were imputed using the IterativeImputer which was imported from fancyimpute package across the five years annual data.

ADE (1).ipynb ☆
File Edit View Insert Runtime Tools Help Last saved at 3:05 PM
- Code + Text

```
[ ] from fancyimpute import IterativeImputer
    MICE_imputer = IterativeImputer()
    First_MICE = normalized_Firstdf.copy(deep=True)
    First_MICE.iloc[:, :] = MICE_imputer.fit_transform(First_MICE)
    FirstYr_imp = First_MICE.iloc[:, :]
    FirstYr_imp.head()

    Second_MICE = normalized_Seconddf.copy(deep=True)
    Second_MICE.iloc[:, :] = MICE_imputer.fit_transform(Second_MICE)
    SecondYr_imp = Second_MICE.iloc[:, :]
    SecondYr_imp.head()

    Third_MICE = normalized_Thirddf.copy(deep=True)
    Third_MICE.iloc[:, :] = MICE_imputer.fit_transform(Third_MICE)
    ThirdYr_imp = Third_MICE.iloc[:, :]
    ThirdYr_imp.head()

    Fourth_MICE = normalized_Fourthdf.copy(deep=True)
    Fourth_MICE.iloc[:, :] = MICE_imputer.fit_transform(Fourth_MICE)
    FourthYr_imp = Fourth_MICE.iloc[:, :]
    FourthYr_imp.head()
```

Figure 14: Filling missing values using Multiple Imputation by Chained Equations

3.6 Executing Feature Selection Techniques

This section shows the execution of the six feature selection techniques for ease of replication of experiments.

The sklearn package enabled the implementation of the classifier based feature selection techniques after splitting the data in figure 16, libraries such as mutual_info_classif, mutual_info_regression, SelectKBest, SelectPercentile were imported for the mutual gain as shown in figure 17, Also, from the sklearn.ensemble the GradientBoostingClassifier and the RandomForestClassifier is imported as shown in figure 19 and 20. The mlxtend.feature_selection enabled the importation of the ExhaustiveFeatureSelector as shown in figure 21. The features gotten from each technique is then ensemble through a voting technique of unanimous, minority, hard voting, and any vote. The features selected are shown in the appendix.

ADE (1).ipynb

File Edit View Insert Runtime Tools Help Last saved at 3:05 PM

Code + Text

```
#Pearson Correlation
def correlation(dataset, threshold):
    col_corr = set()
    corr_matrix = dataset.corr()
    for i in range(len(corr_matrix.columns)):
        for j in range(i):
            if abs(corr_matrix.iloc[i, j]) > threshold:
                colname = corr_matrix.columns[i]
                col_corr.add(colname)
    return col_corr

corr_features = correlation(FullMice_Imputed, 0.6)
print('correlated features: ', len(set(corr_features)) ) #33 features
```

```
correlated features:  33
```

```
FullMice_ImputedCorr = FullMice_Imputed
FullMice_ImputedCorr.drop(labels=corr_features, axis=1, inplace=True)
FullMice_ImputedCorr.head()
```

	X1	X2	X4	X5	X7	X8	X9	X12	X13	X15	X
0	0.036336	-0.033776	-0.043896	0.007971	-0.007676	-0.069140	-0.043276	-0.026338	-0.012065	-0.015553	-0.012C

Figure 15: Execution of Pearson Feature Selection

ADE (1).ipynb

File Edit View Insert Runtime Tools Help Last saved at 3:05 PM

+ Code + Text

```
#split data
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(FullMice_Imputed, ax,
                                                    test_size=0.3,
                                                    random_state=101)
```

Figure 16: Data split for classifier-based feature selection.

ADE (1).ipynb

File Edit View Insert Runtime Tools Help Last saved at 3:05 PM

+ Code + Text

```python
#Mutual Information Gain
from sklearn.feature_selection import mutual_info_classif, mutual_info_regression
from sklearn.feature_selection import SelectKBest, SelectPercentile
mutual_info = mutual_info_classif(X_train.fillna(0), y_train)
#mutual_info
mi_series = pd.Series(mutual_info)
mi_series.index = X_train.columns
mi_series.sort_values(ascending=False)
#K best feature
k_best_features = SelectKBest(mutual_info_classif, k=31).fit(X_train.fillna(0), y_train)
print('Selected top 31 features: {}'.format(X_train.columns[k_best_features.get_support()]))
```

```
Selected top 31 features: Index(['X1', 'X2', 'X4', 'X5', 'X7', 'X8', 'X9', 'X12', 'X13', 'X15', 'X20',
       'X21', 'X27', 'X28', 'X29', 'X30', 'X32', 'X33', 'X37', 'X39', 'X41',
       'X42', 'X43', 'X47', 'X53', 'X55', 'X56', 'X57', 'X58', 'X59', 'X61'],
      dtype='object')
```

```python
#remove columns by information gain
FullMice_ImputedInfoGain = FullMice_Imputed
FullMice_ImputedInfoGain = FullMice_ImputedInfoGain.filter(X_train.columns[k_best_features.get_support()])
```

Figure 17: Features selected by Information gain.

ADE (1).ipynb

File Edit View Insert Runtime Tools Help Last saved at 3:05 PM

+ Code + Text

```python
from sklearn.ensemble import GradientBoostingClassifier
def gbt_importance(X_train,y_train,max_depth=10,top_n=31,n_estimators=50,
                   random_state=0):

    model = GradientBoostingClassifier(n_estimators=n_estimators,
                                       max_depth=max_depth,
                                       random_state=random_state)
    model.fit(X_train, y_train)
    importances = model.feature_importances_
    indices = np.argsort(importances)[::-1]
    feat_labels = X_train.columns
    std = np.std([tree[0].feature_importances_ for tree in model.estimators_],
                 axis=0) #  inter-trees variability.
    for f in range(X_train.shape[1]):
        print(feat_labels[indices[f]])
    return model

gbt_importance(X_train,y_train)
```

```
X27
X29
X9
X55
X56
```

Figure 18: Execution of Gradient Boosting Classifier for feature selection and importance.

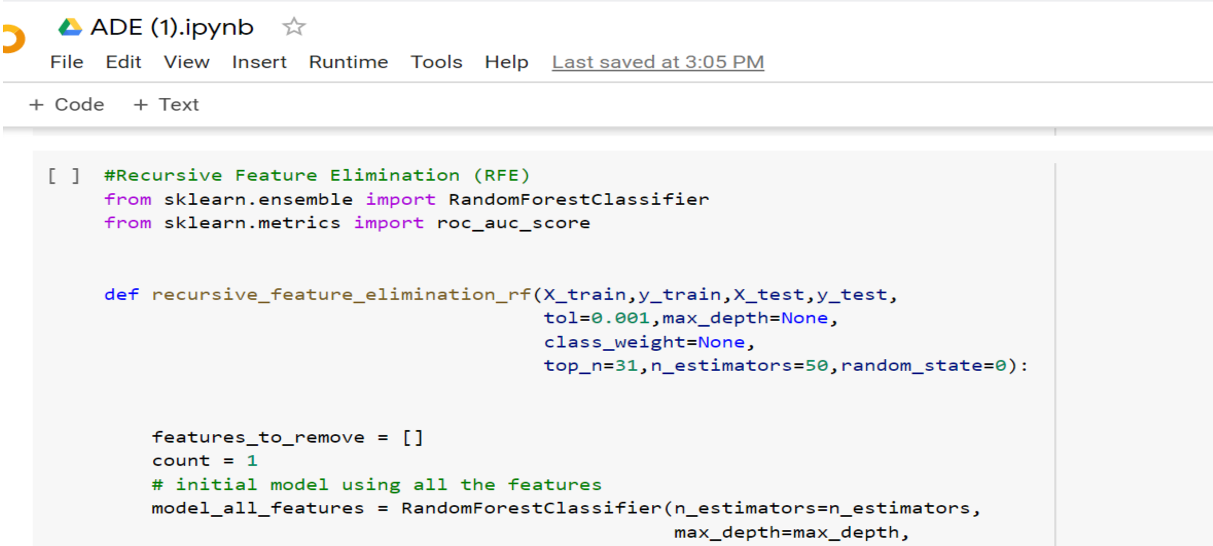

Figure 19: Execution of Recursive elimination feature selection technique

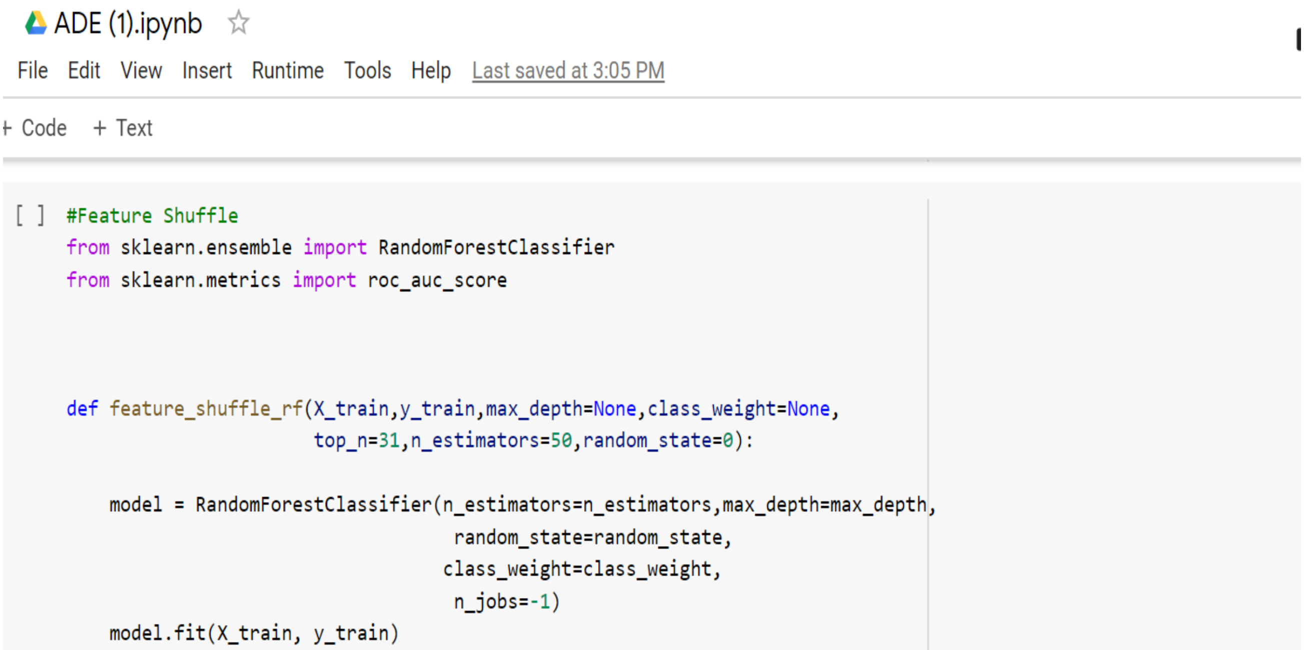

Figure 20: Execution of feature shuffle method

ADE Thesis (1).ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
[ ] #Exhaustive search
    from sklearn.ensemble import RandomForestClassifier
    from sklearn.metrics import roc_auc_score
    from mlxtend.feature_selection import ExhaustiveFeatureSelector as EFS

    efs1 = EFS(RandomForestClassifier(n_jobs=-1,n_estimators=5, random_state=0),
               min_features=1,
               max_features=31,
               scoring='roc_auc',
               print_progress=True,
               cv=2)

    efs1 = efs1.fit(np.array(X_train[X_train.columns[0:64]].fillna(0)), y_train)
```

```
[ ] EFSLabels= np.array(['X27',  'X2',   'X15',  'X55',  'X29',  'X1',   'X39',  'X9',
                         'X58',  'X21',  'X61',  'X57',  'X7',   'X41',  'X13',  'X8',
                         'X42',  'X12',  'X3',   'X45',  'X43',  'X47',  'X5',   'X53',
                         'X33',  'X32',  'X4',   'X28',  'X11',  'X30'])
    FullMice_ImputedEFS = FullMice_Imputed
    FullMice_ImputedEFS =  FullMice_ImputedEFS.filter(EFSLabels)
```

Figure 21: Execution of Exhaustive search method

3.7 Modelling

This section shows the steps taken to build the models that would enable the prediction of bankruptcy.

Step 1

The voting table is loaded unto the jupyter notebook shown in figure 22

```
#Load voting table from Excel
VotingTable = pd.read_csv("/content/drive/My Drive/VotingTableA.csv")
VotingTable.head()
```

Figure 22: Loading Voting Table

Step 2

SMOTE is implemented on the data using imblearn library as shown in figure 23.

ADE (1).ipynb

File Edit View Insert Runtime Tools Help Last saved at 3:05 PM

+ Code + Text

```
#SMOTE split data
from imblearn.over_sampling import SMOTE
sm = SMOTE(random_state = 101)
X_smoteALL, y_smoteALL = sm.fit_sample(X_trainALL, y_trainALL.ravel())

X_smote5, y_smote5 = sm.fit_sample(X_train5, y_train5.ravel())

X_smote4, y_smote4 = sm.fit_sample(X_train4, y_train4.ravel())

X_smote1, y_smote1 = sm.fit_sample(X_train1, y_train1.ravel())


X_train5_sm, X_test5_sm, y_train5_sm, y_test5_sm = train_test_split(X_smote5, y_smote5,
                                                                    test_size=0.3, random_st
X_trainALL_sm, X_testALL_sm, y_trainALL_sm, y_testALL_sm = train_test_split(X_smoteALL, y_sm
                                                                    test_size=0.3, random_st
X_train4_sm, X_test4_sm, y_train4_sm, y_test4_sm = train_test_split(X_smote4, y_smote4,
                                                                    test_size=0.3, random_st
X_train1_sm, X_test1_sm, y_train1_sm, y_test1_sm = train_test_split(X_smote1, y_smote1,
                                                                     test_size=0.3, random_s
```

Figure 23: Execution of balanced data

Step 3

ANN Execution

For the execution of the ANN execution the Pyswarm library is used from python. Figure 24 & 25 shows the execution of the PSO-ANN model. The number of hidden layers is built using the 2/3 number of inputs with the addition of the number of outputs rule system.

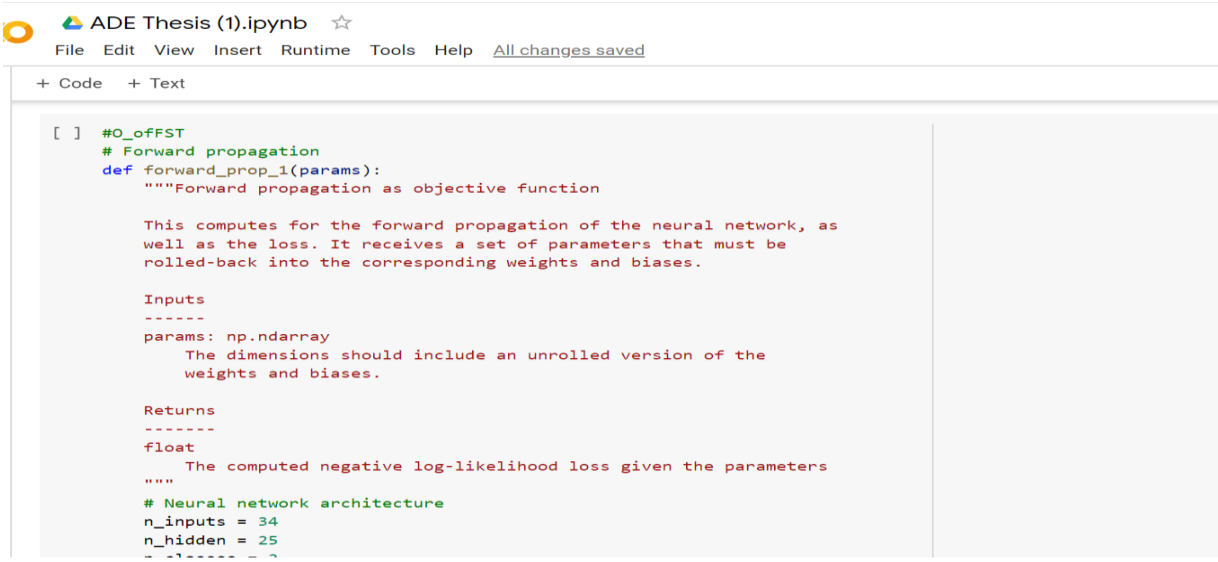

Figure 24: Execution of PSO-ANN model

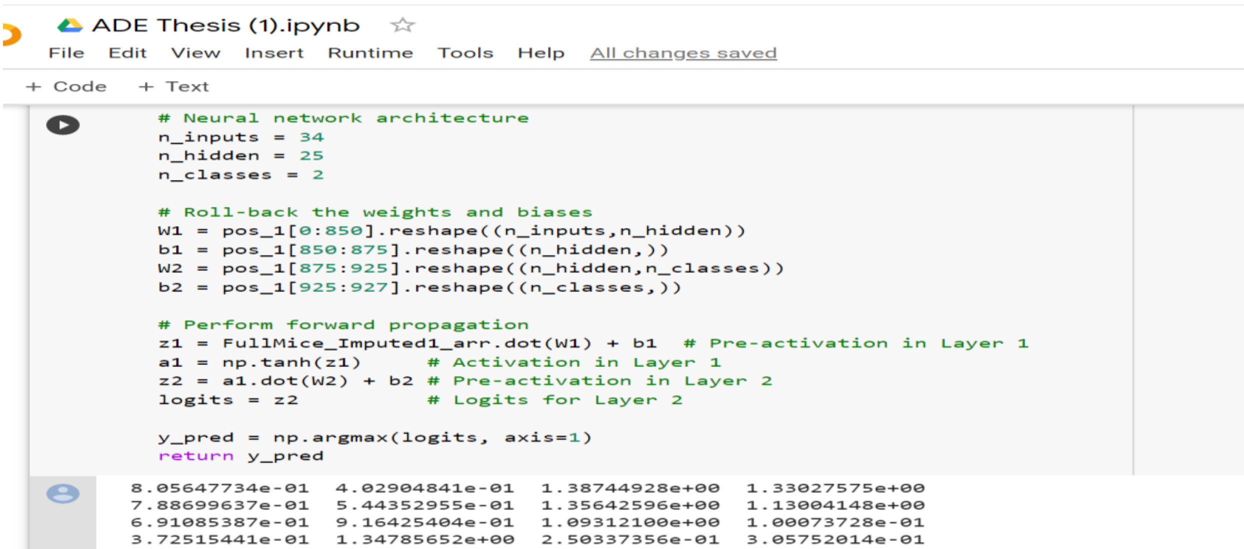

Figure 25: Execution of ANN Architecture

ADE Thesis (1).ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
#accuracy of ANN with PSO
(predict1_sm(X_train1_sm, pos_1_sm) == y_train1_sm).mean()#0.969
```

```
0.969938195302843
```

```
#ROC of ANN with PSO
from sklearn import metrics
fpr, tpr, thresholds = metrics.roc_curve(y_train1_sm, predict1_sm(X_train1_sm, pos_1_sm))
metrics.auc(fpr, tpr) #0.969
```

```
0.9699367296499444
```

Figure 26: Results of PSO-ANN on balanced data.

Step 4

Random Forest model Execution

The libraries needed for the implementation is imported as shown below in figure 27. The random.seed is also set to enable different outputs for each iteration of the experiment.

ADE Thesis (1).ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
#Import Random Forest Model
from sklearn.ensemble import RandomForestClassifier
from sklearn import metrics

#Create a RF classifier
rf_classifier = RandomForestClassifier(n_estimators = 200,
                                       bootstrap = True,
                                       random_state = 101)
#5
```

Figure 27: Libraries imported for random forest model

Step 5

XGboost model Execution

The libraries needed for the implementation is imported as shown below in figure 28. The random.seed is also set to enable different outputs for each experiment.

ADE (1).ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
[ ] #Import XGBoost Model
    from sklearn.ensemble import GradientBoostingClassifier
    from sklearn import metrics

    #Create a RF classifier
    xgb_classifier = GradientBoostingClassifier(n_estimators = 200,
                                                max_depth = 10,
                                                 random_state = 101)
```

Figure 28: Libraries imported XGboost model

Step 6

The library needed for the execution of SVC is imported from sklearn library shown in figure 26.

ADE (1).ipynb

File Edit View Insert Runtime Tools Help All changes saved

Code + Text

```
from sklearn.svm import SVC
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn import metrics

#Create a XGB classifier
xgb_classifier = GradientBoostingClassifier(n_estimators = 200,
                                            max_depth = 10,
                                             random_state = 101)

#Create a RF classifier
rf_classifier = RandomForestClassifier(n_estimators = 200,
                                        bootstrap = True,
                                        random_state = 101)

models = [xgb_classifier, rf_classifier]
```

Figure 26: Libraries imported for the ensemble model of SVC-RF-XGboost

4 Appendix

4.1 Feature selection table

Features	Exhaustive feature	Random shuffling	Recursive feature elimination	Gradient boosting	Information gain	Correlation	Count
X1	*	*		*	*	*	5
X2	*	*	*	*	*	*	6
X3	*						1
X4	*	*		*	*	*	5
X5	*	*		*	*	*	5
X6							0
X7	*	*		*	*	*	5
X8	*	*	*	*	*	*	6
X9	*	*	*	*	*	*	6
X10							0
X11	*						1
X12	*	*	*	*	*	*	6
X13	*	*		*	*	*	5
X14							0
X15	*	*	*	*	*	*	6
X16							0
X17							0
X18							0
X19							0
X20	*	*		*	*	*	5
X21	*	*	*	*	*	*	6
X22							0
X23							0
X24							0
X25							0
X26							0
X27	*	*	*	*	*	*	6
X28	*	*		*	*	*	5
X29	*	*	*	*	*	*	6
X30	*		*	*	*	*	5
X31							0
X32	*	*	*	*	*	*	6
X33	*	*	*	*	*	*	6
X34							0
X35							0
X36							0
X37		*	*	*	*	*	5
X38							0
X39	*	*	*	*	*	*	6
X40							0
X41	*	*	*	*	*	*	6
X42	*	*	*	*	*	*	6
X43	*	*	*	*	*	*	6
X44							0
X45	*						1
X46							0
X47	*	*	*	*	*	*	6
X48							0
X49							0
X50							0
X51							0
X52							0
X53	*	*	*	*	*	*	6
X54							0
X55	*	*	*	*	*	*	6
X56		*	*	*	*	*	5
X57	*	*	*	*	*	*	6
X58	*	*	*	*	*	*	6
X59			*	*	*	*	4
X60							0
X61	*	*	*	*	*	*	6
X62							0
X63							0
X64							0

References

Son, H., Hyun, C., Phan, D. and Hwang, H.J. (2019) 'Data analytic approach for bankruptcy prediction', *Expert Systems with Applications*, 138, p. 112816, ScienceDirect. doi: 10.1016/j.eswa.2019.07.033

Zięba, M., Tomczak, S.K. and Tomczak, J.M. (2016) 'Ensemble boosted trees with synthetic features generation in application to bankruptcy prediction', *Expert Systems with Applications*, 58, pp. 93-101, ScienceDirect. doi: 10.1016/j.eswa.2016.04.001