

ANTICIPATING BUSINESS BANKRUPTCY

Date	6 November 2023
Team ID	PNT2022TMID592760
Project Name	Project – Anticipating Business Bankruptcy

Bankruptcy prediction has been a subject of interest for almost a century, and it still ranks high among hottest topics in economics. The aim of predicting financial distress is to develop a predictive model that combines various econometric measures and allows us to foresee the financial condition of a firm.

The purpose of the bankruptcy prediction is to assess the financial condition of a company and its future perspectives within the context of long-term operation on the market.

The dataset is about bankruptcy prediction of Polish companies. The data was collected from Emerging Markets Information Service (EMIS), which is a database containing information on emerging markets around the world. The bankrupt companies were analyzed in the period 2000-2012, while the still operating companies were evaluated for year 2007. Basing on the collected data five classification cases were distinguished, that depends on the forecasting period: The data contains financial rates from 1st year of the forecasting period and corresponding class label that indicates bankruptcy status.

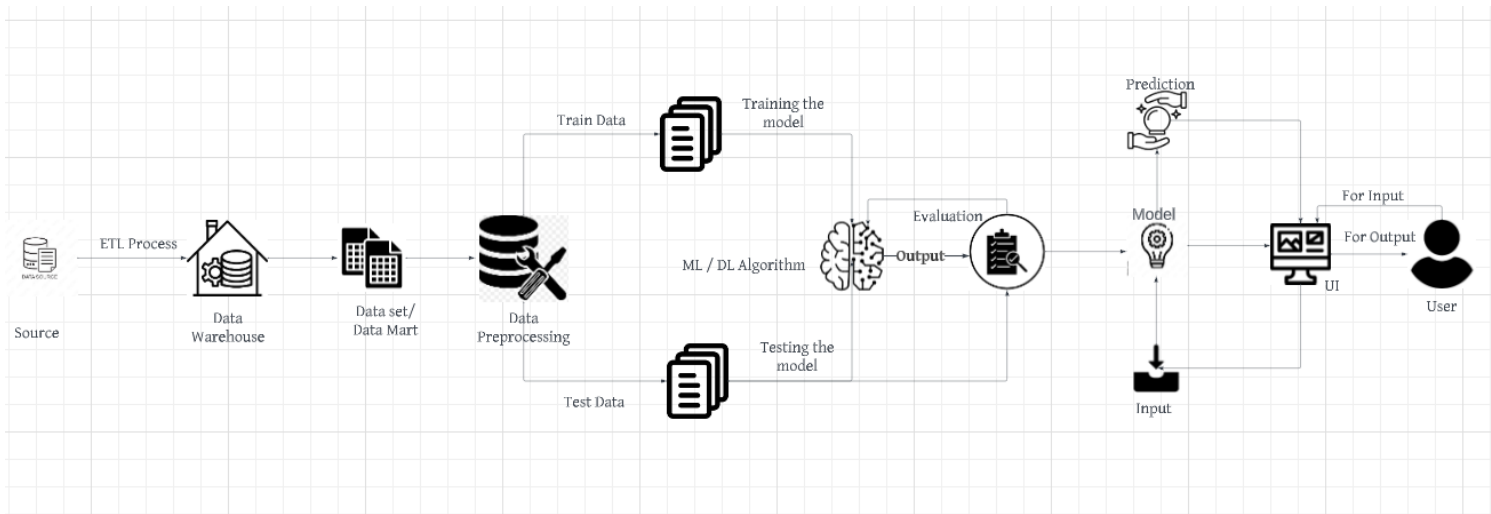
Aim:

The purpose of the bankruptcy prediction is to assess the financial condition of a company and its future perspectives within the context of long-term operation on the market.

Technical Architecture:

In this architecture, a user inputs the relevant engagement metrics such as ATTR1 to ATTR10 scores. The Flask application is responsible for handling the request and returning the prediction that if the company could go bankrupt or not in the forecasting period.

The Flask application then passes the input metrics to the trained algorithm model, which processes the input data and returns a prediction based on the learned relationship between the input metrics and repayment interval.



Project Flow:

- User interacts with the UI to enter the input.
- Entered input is analyzed by the model which is integrated.
- Once model analyses the input the prediction is showcased on the UI

To accomplish this, we must complete all the activities listed below:

- **Define / Problem understanding**

- o Specify the business problem.
- o Business Requirements
- o Social or Business Impact

- **Data Collection and Preparation**

- o Collect the dataset.
- o Data Preparation

- **Exploratory Data Analysis**

- o Descriptive statistical
- o Visual Analysis

- **Model Building**

- o Train-test split
- o Training and testing the Models using multiple algorithms.

- **Performance Testing**

- o Comparing model accuracy
- o Graphical representation of the model comparison.

- **Model Deployment**

- o Save the best model.
- o Test the model.
- o Integrate with Web Framework
- o GUI

- **Project Demonstration & Documentation**

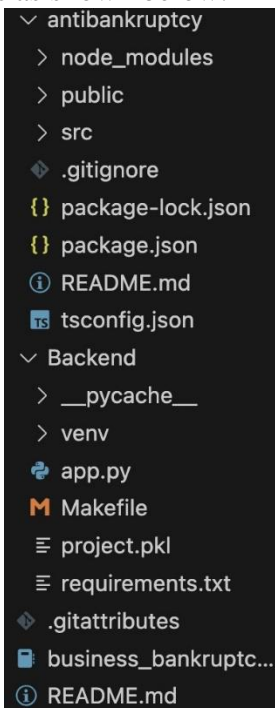
Prior Knowledge:

You must have prior knowledge of the following topics to complete this project.

- ML Concepts:
 - o Supervised learning: <https://www.javatpoint.com/supervised-machine-learning>
 - o Linear Regression: <https://www.javatpoint.com/linear-regression-in-machine-learning>
 - o SVM: <https://www.javatpoint.com/machine-learning-support-vector-machine algorithm>
 - o Decision tree: <https://www.javatpoint.com/machine-learning-decision-tree classification algorithm>
 - o Random forest: <https://www.javatpoint.com/machine-learning-random-forest algorithm>
- Evaluation metrics: <https://www.analyticsvidhya.com/blog/2019/08/11-important-modevaluation-error-metrics/>
- Regularisation: <https://www.javatpoint.com/regularization-in-machine-learning>
- Flask Basics: https://www.youtube.com/watch?v=lj4I_CvBnt0

Project Structure:

Create project folder which contains files as shown below:



- The data is obtained in csv files and is split for training and testing.
- We are building a Flask application which will require the html files to be stored in the templates folder.
- app.py file is used for routing purposes using scripting.
- project.pkl is the saved model. This will further be used in the Flask integration.

Milestone 1: Define Problem/ Problem Understanding

Activity 1: Specify the Business Problem

The aim of the bankruptcy prediction is to evaluate a company's financial situation and its prospects for the future in the context of its long-term activity on the market.

Activity 2: Business Requirements

Risk assessment: By predicting the likelihood of bankruptcy, businesses can evaluate the financial risks associated with engaging with a particular company. It allows them to make informed decisions about extending credit, entering into contracts, or investing in partnerships.

Financial planning: Accurate bankruptcy prediction helps companies in their financial planning activities. It allows them to anticipate potential cash flow problems, identify areas for improvement, and take proactive measures to mitigate risks.

Early warning system: Bankruptcy prediction models can serve as an early warning system, alerting companies to potential financial distress before it becomes critical. This allows them to take corrective actions promptly, such as restructuring debts, renegotiating contracts, or implementing cost-saving measures.

Activity 4: Social or Business Impact.

Social Impact:

- **Job losses:** When a company goes bankrupt, it often leads to layoffs and job losses for its employees. Predicting bankruptcy helps mitigate the impact by allowing employees to prepare for potential job transitions in advance. It also enables governments, labor organizations, and affected individuals to develop strategies for reemployment and support.
- **Investor protection:** Bankruptcy prediction plays a vital role in investor protection. When investors, shareholders, or creditors can anticipate a company's financial distress, they can take appropriate measures to protect their investments or minimize losses. It enables them to diversify their portfolios, sell stocks, or adjust their financial positions in a timely manner.

Business Impact:

- **Industry dynamics and competition:** Bankruptcies can reshape industry dynamics and alter competitive landscapes. Predicting bankruptcy enables businesses to anticipate market shifts, identify potential acquisition opportunities, or adjust their strategies to gain a competitive advantage. It fosters a more dynamic business environment by facilitating adaptive responses to market changes.
- **Financial system stability:** The stability of the financial system is crucial for economic growth and prosperity. Predicting company bankruptcies contributes to financial system stability by identifying potential risks and vulnerabilities. It helps financial institutions.

assess their exposure, manage risk, and make informed lending decisions. By avoiding excessive risk concentration, the likelihood of systemic financial crises can be reduced.

Milestone 2: Data Collection and Preparation:

Machine Learning depends heavily on data. It is the most crucial part aspect that makes algorithm training possible. So, this section guides on how to download dataset.

Activity 1: Collect the dataset.

There are many popular open sources for collecting data. Eg: kaggle.com, UCI repository, etc.

In this project we have used .csv data. This data is downloaded from kaggle.com. Please refer to the link given below to download the dataset.

Link: <https://www.kaggle.com/datasets/bhadaneeraj/bankruptcy-detection>

As the dataset is downloaded. Let us read and understand the data properly with the help of some visualization techniques and some analyzing techniques.

Note: There are several techniques for understanding the data. But here we have used some of it. In an additional way, you can use multiple techniques.

About this Data

Attribute Information:

Feature	Description
Attr1	net profit / total assets
Attr2	total liabilities / total assets
Attr3	working capital / total assets
Attr4	current assets / short-term liabilities
Attr5	$[(\text{cash} + \text{short-term securities} + \text{receivables} - \text{short-term liabilities}) / (\text{operating expenses} - \text{depreciation})] * 365$
Attr6	retained earnings / total assets
Attr7	EBIT / total assets
Attr8	book value of equity / total liabilities
Attr9	sales / total assets
Attr10	equity / total assets

Activity 1.1: Importing the Libraries

Please refer the google colab file for the entire process:

https://colab.research.google.com/drive/1hHy1UfZKC7h0DD6DXVMNtX_STeGkYJ5u#scrollTo=Htey9kyW9IEC

```
[ ] import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from scipy import stats
from sklearn import metrics
from sklearn.preprocessing import MinMaxScaler
from imblearn.over_sampling import SMOTE
from sklearn.model_selection import GridSearchCV
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, classification_report, roc_auc_score, roc_curve
from sklearn.preprocessing import MinMaxScaler
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from scipy.stats import uniform
from sklearn.neighbors import KNeighborsClassifier
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
import pickle
```

Activity 1.2: Read the Dataset

Our dataset format might be in .csv, excel files, .txt, .json, etc. We can read the dataset with the help of pandas.

In pandas we have a function called `read_csv()` to read the dataset. As a parameter we have to give the directory of the csv file.

```
[ ] data = pd.read_csv('/content/1year.csv')
df = data.iloc[:, :10]
df['class'] = data['class']
df.head(2)
```

	Attr1	Attr2	Attr3	Attr4	Attr5	Attr6	Attr7	Attr8	Attr9	Attr10	class
0	0.20055	0.37951	0.39641	2.0472	32.351	0.38825	0.24976	1.3305	1.1389	0.50494	0
1	0.20912	0.49988	0.47225	1.9447	14.786	0	0.25834	0.99601	1.6996	0.49788	0

Activity 2: Data Preparation

As we have understood how the data is, let's pre-process the collected data.

The download data set is not suitable for training the machine learning model as it might have so much randomness, so we need to clean the dataset properly in order to fetch good results. This activity includes the following steps.

- o Getting the Preliminary Information about the Dataset
- o Handling missing values
- o Dropping Unwanted column

Activity 2.1: Getting the Preliminary Information about the Dataset

i.e. Non-Null, Count, Dtype

Let's find the basic information about our dataset. To find the data type, `df.info()` function is used.


```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 7012 entries, 0 to 7011  
Data columns (total 11 columns):  
#   Column      Non-Null Count  Dtype  
---  ---  
0   Attr1       7009 non-null   float64  
1   Attr2       7009 non-null   float64  
2   Attr3       7009 non-null   float64  
3   Attr4       6982 non-null   float64  
4   Attr5       7004 non-null   float64  
5   Attr6       7009 non-null   float64  
6   Attr7       7009 non-null   float64  
7   Attr8       6987 non-null   float64  
8   Attr9       7011 non-null   float64  
9   Attr10      7009 non-null   float64  
10  class       7012 non-null   int64  
dtypes: float64(10), int64(1)  
memory usage: 602.7 KB
```

Activity 2.2: Handling missing values:

Check and number of missing values and its percentage for the all the columns and fill the missing values only for required columns (Input).

For checking the null values, `df.isnull()` function is used. To sum those null values, we use `.sum()` function.

```
[ ] df.isnull().sum()
```

```
Attr1      3
Attr2      3
Attr3      3
Attr4     30
Attr5      8
Attr6      3
Attr7      3
Attr8     25
Attr9      1
Attr10     3
class      0
dtype: int64
```

```
[ ] df.isnull().sum().sum()
```

```
82
```

When we checked for the “?” in the dataset we can easily find it. So, the meaning of the “?” was same as Null or NA, but we used a function `to_numeric()` defined as a function in the pandas library is to convert argument(s) to a numeric type, handling errors according to the specified error-handling arguments. In this case all the “?” is converted to NA while all the float values which are stored as strings will be directly converted to Numeric values.

```
[ ] df['Attr1'] = pd.to_numeric(df['Attr1'], errors='coerce')
df['Attr2'] = pd.to_numeric(df['Attr2'], errors='coerce')
df['Attr3'] = pd.to_numeric(df['Attr3'], errors='coerce')
df['Attr4'] = pd.to_numeric(df['Attr4'], errors='coerce')
df['Attr5'] = pd.to_numeric(df['Attr5'], errors='coerce')
df['Attr6'] = pd.to_numeric(df['Attr6'], errors='coerce')
df['Attr7'] = pd.to_numeric(df['Attr7'], errors='coerce')
df['Attr8'] = pd.to_numeric(df['Attr8'], errors='coerce')
df['Attr9'] = pd.to_numeric(df['Attr9'], errors='coerce')
df['Attr10'] = pd.to_numeric(df['Attr10'], errors='coerce')
```

Now we replaced the 82 null values from age column with the median of column as below.

```
[ ] df['Attr1'].fillna(df['Attr1'].median(),inplace =True)
    df['Attr2'].fillna(df['Attr2'].median(),inplace =True)
    df['Attr3'].fillna(df['Attr3'].median(),inplace =True)
    df['Attr4'].fillna(df['Attr4'].median(),inplace =True)
    df['Attr5'].fillna(df['Attr5'].median(),inplace =True)
    df['Attr6'].fillna(df['Attr6'].median(),inplace =True)
    df['Attr7'].fillna(df['Attr7'].median(),inplace =True)
    df['Attr8'].fillna(df['Attr8'].median(),inplace =True)
    df['Attr9'].fillna(df['Attr9'].median(),inplace =True)
    df['Attr10'].fillna(df['Attr10'].median(),inplace =True)
```

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

```
Attr1      0
Attr2      0
Attr3      0
Attr4      0
Attr5      0
Attr6      0
Attr7      0
Attr8      0
Attr9      0
Attr10     0
class      0
dtype: int64
```

Activity 2.3: Dropping Unwanted column

In the dataset, we need all the columns, as every column is important for prediction.

Milestone 3: Exploratory Data Analysis

Activity 1: Descriptive Statistical

Descriptive analysis is to study the basic features of data with the statistical process. Here pandas have a worthy function called describe. With this describe() function we can understand the unique, top, and frequent values of categorical features. And we can find mean, std, min, max and percentile values of continuous features.

```
df.describe()
```

	Attr1	Attr2	Attr3	Attr4	Attr5	Attr6	Attr7	Attr8	Attr9	Attr10	class
count	7012.000000	7012.000000	7012.000000	7012.000000	7012.000000	7012.000000	7012.000000	7012.000000	7012.000000	7012.000000	7012.000000
mean	0.087257	0.489040	0.201028	1.632909	-3.876968	0.043682	0.104382	1.199756	1.487737	0.489200	0.036509
std	0.094333	0.248671	0.240993	0.848026	53.979919	0.102125	0.108990	1.064593	0.700571	0.243055	0.187566
min	-0.186650	0.000000	-0.469670	0.000000	-166.800000	-0.220210	-0.212090	-2.003200	0.000005	-0.245680	0.000000
25%	0.024510	0.296760	0.037581	1.064775	-31.555500	0.000000	0.031250	0.448965	1.037450	0.311465	0.000000
50%	0.076162	0.482320	0.182170	1.503450	-5.234550	0.000000	0.090330	1.018100	1.206400	0.492930	0.000000
75%	0.136060	0.670735	0.360150	1.959500	23.078500	0.066151	0.161073	1.593325	1.808025	0.672330	0.000000
max	0.368510	1.252900	0.865020	4.534600	160.930000	0.367010	0.427830	4.958100	3.765700	1.000000	1.000000

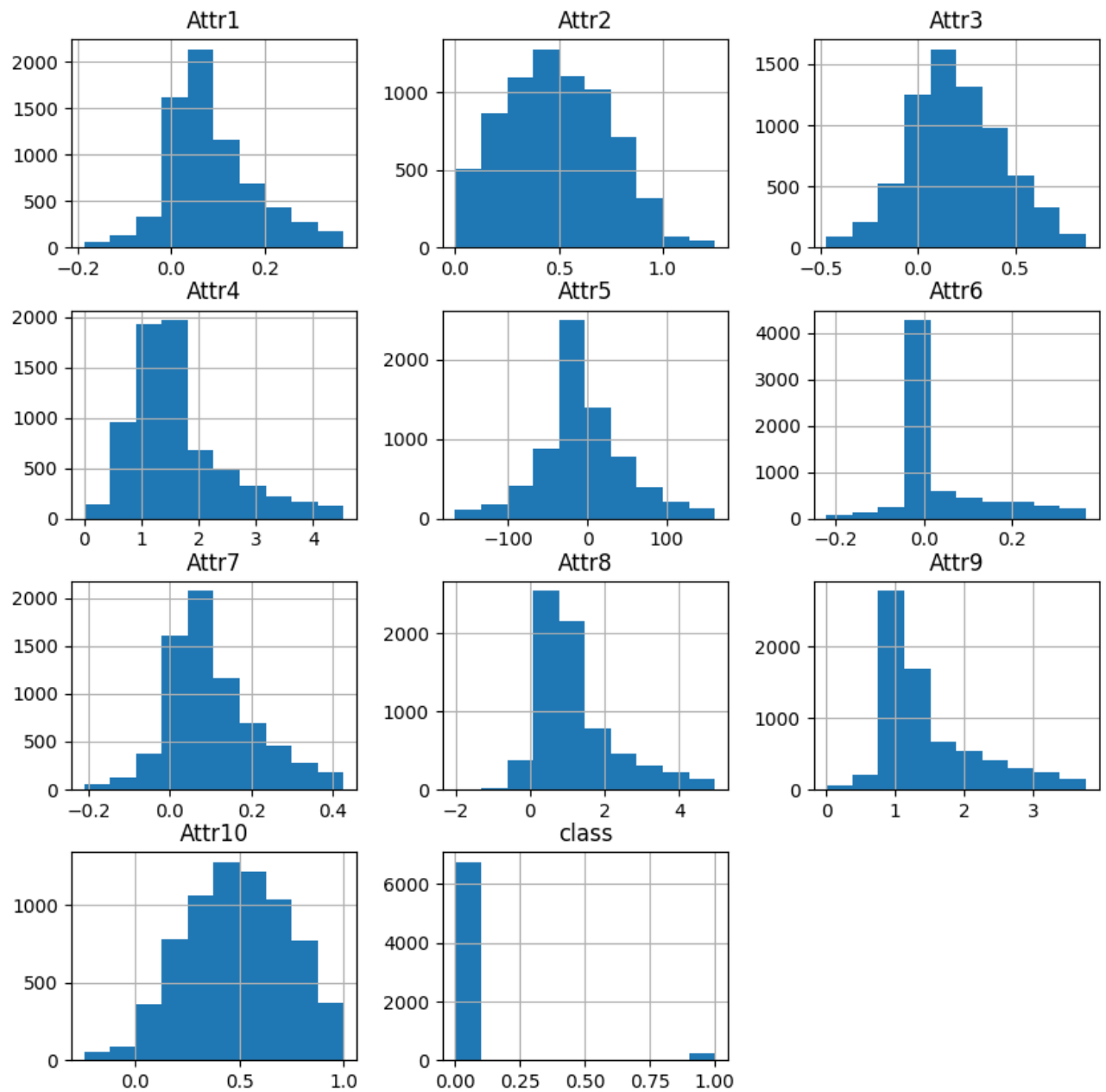
Activity 2: Visualization

Visual analysis is the process of using visual representations, such as charts, plots, and graphs, to explore and understand data. It is a way to quickly identify patterns, trends, and outliers in the data, which can help to gain insights and make informed decisions.

Histplot

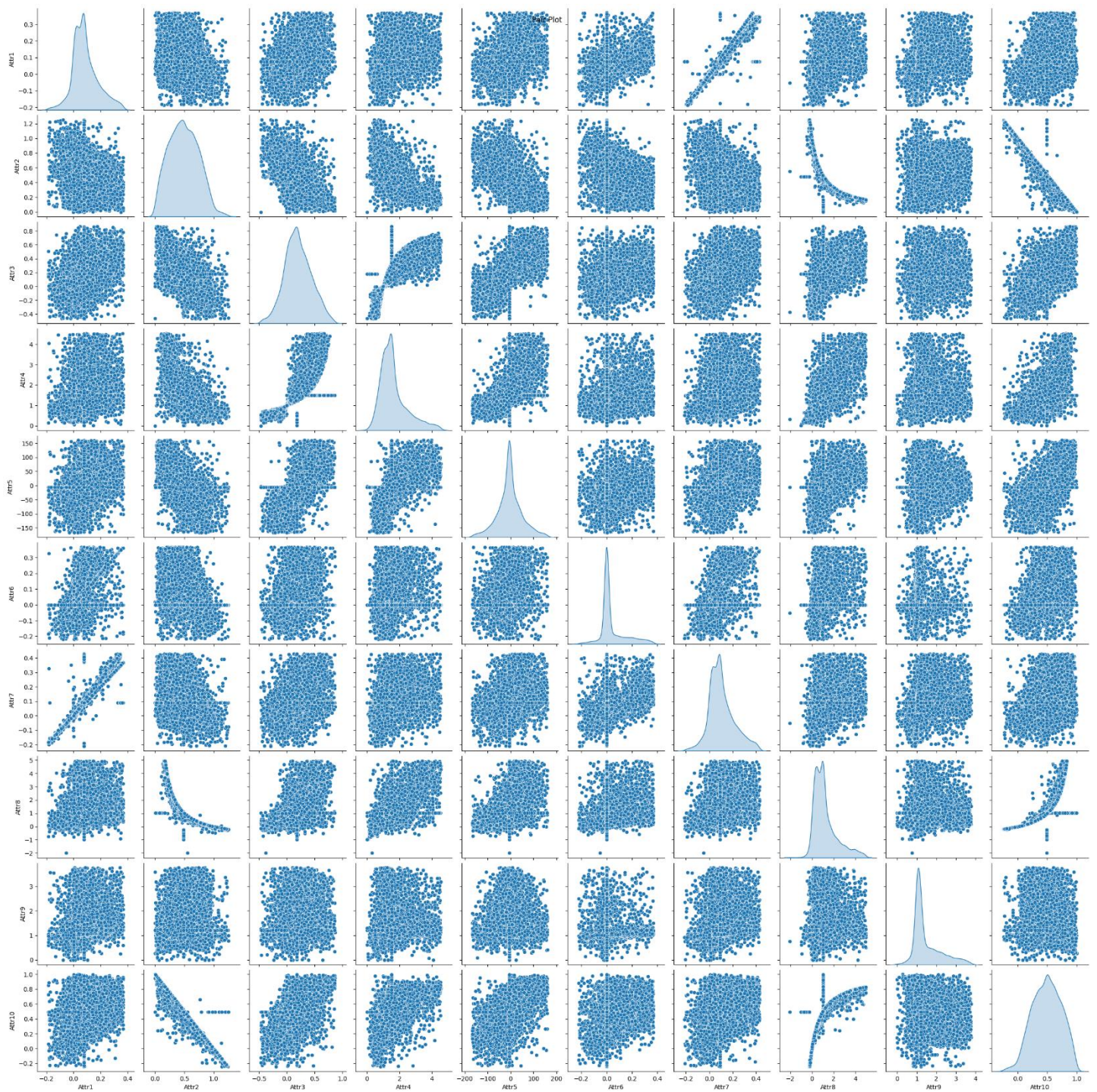
```
df.hist(figsize=(10,10))
```

```
array([[<Axes: title={'center': 'Attr1'}>,  
       <Axes: title={'center': 'Attr2'}>,  
       <Axes: title={'center': 'Attr3'}>],  
       [<Axes: title={'center': 'Attr4'}>,  
       <Axes: title={'center': 'Attr5'}>,  
       <Axes: title={'center': 'Attr6'}>],  
       [<Axes: title={'center': 'Attr7'}>,  
       <Axes: title={'center': 'Attr8'}>,  
       <Axes: title={'center': 'Attr9'}>],  
       [<Axes: title={'center': 'Attr10'}>,  
       <Axes: title={'center': 'class'}>, <Axes: >]], dtype=object)
```



Pairplot:

```
[ ] sns.pairplot(data=df, vars = ["Attr1","Attr2","Attr3","Attr4","Attr5","Attr6","Attr7","Attr8","Attr9","Attr10"], diag_kind='kde', kind='scatter')
plt.suptitle(f'Pair Plot')
plt.show()
```

Heatmap:

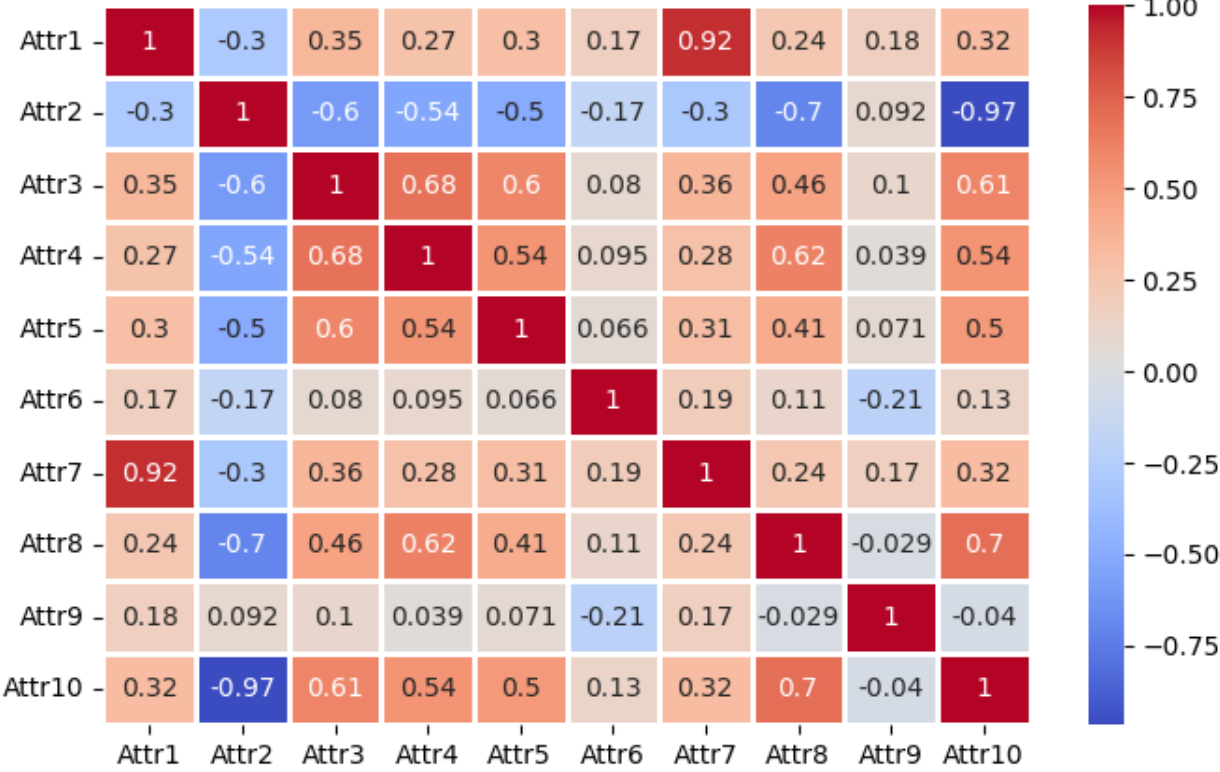
```

var_corr=["Attr1","Attr2","Attr3","Attr4","Attr5","Attr6","Attr7","Attr8","Attr9","Attr10"]
correlation_matrix = df[var_corr].corr()

plt.figure(figsize=(8, 5))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', linewidths=1)
plt.title(f'Correlation Heatmap between {", ".join(var_corr)}')
plt.show()

```

Correlation Heatmap between Attr1, Attr2, Attr3, Attr4, Attr5, Attr6, Attr7, Attr8, Attr9, Attr10



Activity 2.1 Class Analysis:

In the bar graph we can see that most of the companies are not bankrupt, so we need to balance the dataset.

```
[ ] smote = SMOTE()  
    x_train_smote,y_train_smote = smote.fit_resample(x_train,y_train)
```

```
▶ y_train.value_counts()
```

```
📄 0    5401  
   1     208  
   Name: class, dtype: int64
```

```
[ ] y_train_smote.value_counts()
```

```
0    5401  
1    5401  
Name: class, dtype: int64
```


Milestone 4: Model Building

Activity 1 Train-test split and Scaling.

Now let's split the Dataset into train and test sets. First split the dataset into X and y and then split the data set.

Here X and y variables are created. On X variable, data is passed with dropping the target variable. And on y target variable is passed. For splitting training and testing data we are using `train_test_split()` function from `sklearn`. As parameters, we are passing X, y, `test_size`, `random_state`.

In the current project we have below columns as X variable and y variables (Target variable): "Class."

```
[ ] y=df["class"]
     x=df.drop(columns=['class'])
```

```
x.head(2)
```

	Attr1	Attr2	Attr3	Attr4	Attr5	Attr6	Attr7	Attr8	Attr9	Attr10
0	0.20055	0.37951	0.39641	2.0472	32.351	0.0	0.24976	1.33050	1.1389	0.50494
1	0.20912	0.49988	0.47225	1.9447	14.786	0.0	0.25834	0.99601	1.6996	0.49788

```
[ ] scale=MinMaxScaler()
     X_scaled= pd.DataFrame(scale.fit_transform(x),columns =x.columns)
     X_scaled.head(2)
```

	Attr1	Attr2	Attr3	Attr4	Attr5	Attr6	Attr7	Attr8	Attr9	Attr10
0	0.697457	0.302905	0.648900	0.451462	0.607668	0.375004	0.721731	0.478890	0.302440	0.602579
1	0.712894	0.398978	0.705722	0.428858	0.554072	0.375004	0.735139	0.430841	0.451336	0.596911

Activity 2: Dealing with Imbalanced data.

```
[ ] smote = SMOTE()
     x_train_smote,y_train_smote = smote.fit_resample(x_train,y_train)
```

```
y_train.value_counts()
```

0	5401
1	208

Name: class, dtype: int64

```
[ ] y_train_smote.value_counts()
```

0	5401
1	5401

Name: class, dtype: int64

Activity 3: Training and testing the models using multiple algorithms.

Now our data is cleaned and it's time to build the model. We can train our data on different algorithms. For this project we are applying three classification algorithms. The best model is saved based on its performance.

Activity 3.1 Support Vector Classifier:

A function named “model” is created, then hyper parameter tuning is done using GridSearchCv and train and test data are passed as the parameters. Inside the function, “SVC” algorithm is initialized and training data is passed to the model with fit() function. Test data is predicted with predict() function and saved in a new variable. For evaluating the model accuracy score, confusion matrix and classification reports are used.

▼ SVM

```
[ ] model = SVC()
    parameters = {'kernel': ['linear', 'rbf', 'sigmoid'],
                  'C': [0.01, 0.1, 1, 10, 100],
                  'gamma': ['scale', 'auto']}
    clf = GridSearchCV(model, param_grid = parameters, verbose = 2)
```

```
▶ clf.fit(x_train_smote, y_train_smote)
```

```
[ ] clf.best_score_

0.9044618832159322
```

```
[ ] clf.best_params_

{'C': 100, 'gamma': 'scale', 'kernel': 'rbf'}
```

```
[ ] model1 = SVC(C=100, gamma='scale', kernel = 'rbf')
    model1.fit(x_train_smote, y_train_smote)
```

▼ SVC
SVC(C=100)

```
[ ] y_pred = model1.predict(x_test)
    accuracy_svc = accuracy_score(y_test, y_pred)
    print("Accuracy:", accuracy_svc)

Accuracy: 0.8153955808980755
```

```
[ ] pd.crosstab(y_test, y_pred)
```

col_0	0	1
class		
0	1128	227
1	32	16

```
[ ] print(classification_report(y_test,y_pred))
```

	precision	recall	f1-score	support
0	0.97	0.83	0.90	1355
1	0.07	0.33	0.11	48
accuracy			0.82	1403
macro avg	0.52	0.58	0.50	1403
weighted avg	0.94	0.82	0.87	1403

Activity 3.2: Logistic Regression Classifier:

A function named “model2” is created, then hyper parameter tuning is done using GridSearchCv and train and test data are passed as the parameters. Inside the function, “Logistic Regression” algorithm is initialized and training data is passed to the model with fit() function. Test data is predicted with predict() function and saved in a new variable. For evaluating the model accuracy score, confusion matrix and classification reports are used.

Logistic Regression

```
[ ] model2 = LogisticRegression()  
    parameters = {  
        'C': [0.001, 0.01, 0.1, 0.2, 0.3, 0.5, 1, 10, 15, 25, 100], # Inverse of regularization strength  
        'penalty': ['l2'], # Regularization type  
        'solver': ['liblinear', 'lbfgs', 'newton-cg'] # Optimization algorithm  
    }
```

```
[ ] clf = GridSearchCV(model2,param_grid = parameters,verbose =2)  
    clf.fit(x_train_smote,y_train_smote)
```

```
[ ] clf.best_score_  
  
0.6647832364988774
```

```
[ ] clf.best_params_  
  
{'C': 0.3, 'penalty': 'l2', 'solver': 'liblinear'}
```

```
[ ] model2 =LogisticRegression(C=25,penalty='l2',solver = 'lbfgs')  
    model2.fit(x_train_smote,y_train_smote)
```

```
LogisticRegression  
LogisticRegression(C=25)
```

```
▶ y_pred = model2.predict(x_test)  
    accuracy_lt = accuracy_score(y_test,y_pred)  
    print("Accuracy:", accuracy_lt)
```

```
➡ Accuracy: 0.6001425516749822
```

```
[ ] pd.crosstab(y_test,y_pred)
```

col_0	0	1
class		
0	811	544
1	17	31

```
[ ] print(classification_report(y_test,y_pred))
```

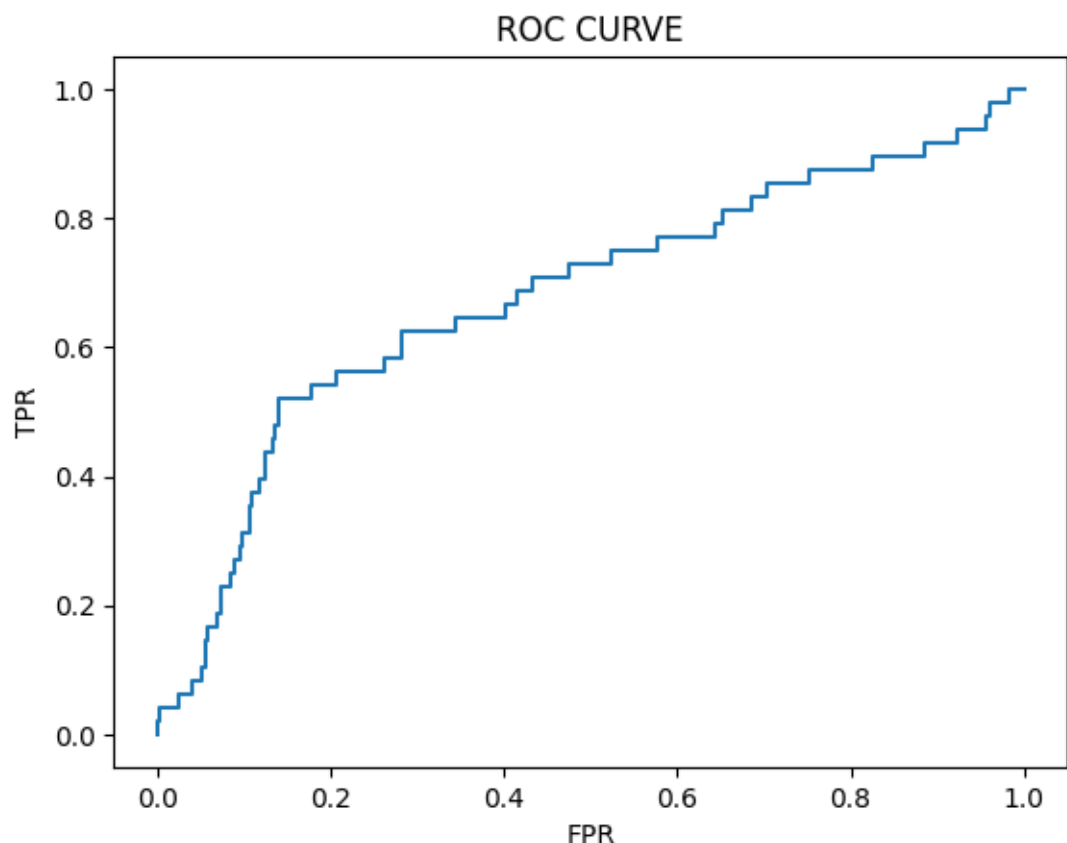
	precision	recall	f1-score	support
0	0.98	0.60	0.74	1355
1	0.05	0.65	0.10	48
accuracy			0.60	1403
macro avg	0.52	0.62	0.42	1403
weighted avg	0.95	0.60	0.72	1403

```
[ ] probability = model2.predict_proba(x_test)[:,1]
probability
```

```
array([0.21910562, 0.6633197 , 0.49001683, ..., 0.81085871, 0.33588813,
       0.24802934])
```

```
[ ] fpr, tpr, thresholds = roc_curve(y_test, probability)
```

```
[ ] plt.plot(fpr, tpr)
plt.xlabel('FPR')
plt.ylabel('TPR')
plt.title('ROC CURVE')
plt.show()
```



Activity 3.3 Random Forest Classifier:

A function named “model3” is created, then hyper parameter tuning is done using GridSearchCv and train and test data are passed as the parameters. Inside the function, “Randomforest” algorithm is initialized and training data is passed to the model with fit() function. Test data is predicted with predict() function and saved in a new variable. For evaluating the model accuracy score, confusion matrix and classification reports are used.

▼ RandomForest Classifier

```
[ ] model3 =RandomForestClassifier(criterion='entropy')
    parameters = {
        'n_estimators': [100, 300],
        'max_depth': [10, 20, 30],
        'min_samples_split': [5, 10],
        'min_samples_leaf': [1, 2],
        'max_features': ['sqrt', 'log2']
    }

[ ] clf = GridSearchCV(model3,param_grid = parameters,verbose =2)
    clf.fit(x_train_smote,y_train_smote)
```

```
[ ] clf.best_score_

0.9537129158311481
```

```
[ ] clf.best_params_

{'max_depth': 30,
 'max_features': 'log2',
 'min_samples_leaf': 1,
 'min_samples_split': 5,
 'n_estimators': 100}
```

```
[ ] model3 =RandomForestClassifier(max_depth= 30, max_features= 'sqrt', min_samples_leaf= 1, min_samples_split= 5, n_estimators=300)
    model3.fit(x_train_smote,y_train_smote)
```

```
▼ RandomForestClassifier
RandomForestClassifier(max_depth=30, min_samples_split=5, n_estimators=300)
```

```
[ ] y_pred= model3.predict(x_test)
    accuracy_rt = accuracy_score(y_test,y_pred)
    print("Accuracy:", accuracy_rt)
```

Accuracy: 0.9265858873841768

```
▶ pd.crosstab(y_test,y_pred)
```

col_0	0	1
class		
0	1294	61
1	42	6

```
[ ] print(classification_report(y_test,y_pred))
```

	precision	recall	f1-score	support
0	0.97	0.95	0.96	1355
1	0.09	0.12	0.10	48
accuracy			0.93	1403
macro avg	0.53	0.54	0.53	1403
weighted avg	0.94	0.93	0.93	1403


Activity 3.4 ANN:

A function named “classification” is created to create an ANN, train and test data are passed as the parameters. Inside the function, training data is passed to the model with fit() function. Test data is predicted with predict() function and saved in a new variable. For evaluating the model accuracy score, confusion matrix is used.

• ANN

```
[ ] classification = Sequential()  
classification.add(Dense(30,activation='relu'))  
classification.add(Dense(128,activation='relu'))  
classification.add(Dense(64,activation='relu'))  
classification.add(Dense(32,activation='relu'))  
classification.add(Dense(1,activation='sigmoid'))
```

```
[ ] classification.compile(optimizer='adam',loss = 'binary_crossentropy',metrics=['accuracy'] )
```

```
 classification.fit(x_train_smote, y_train_smote, epochs=100, batch_size=128)
```

```
[ ] y_pred = classification.predict(x_test)
```

```
[ ] y_pred =y_pred.flatten().astype('int')
```

```
[ ] pd.DataFrame({'Actual y_value': y_test.values.flatten(), 'Predicted y_value': y_pred.flatten().astype(int)}).head(10)
```

	Actual y_value	Predicted y_value
0	0	0
1	0	0
2	0	0
3	0	0
4	0	0
5	0	0
6	0	0
7	0	0
8	0	0
9	0	0

```
[ ] accuracy_ann = accuracy_score(y_test,y_pred)  
print("Accuracy:", accuracy_ann)
```

```
Accuracy: 0.9657875980042766
```

```
[ ] pd.crosstab(y_test,y_pred)
```

col_0	0
class	
0	1355
1	48

Activity 3.5 KNN:

A function named “model4” followed by hyper parameter tuning using Grid Search CV, and train and test data are passed as the parameters. Inside the function, training data is passed to the model with fit() function. Test data is predicted with predict() function and saved in a new variable. For evaluating the model accuracy score, confusion matrix is used.

▼ KNN

```
[ ] model5= KNeighborsClassifier()
    param_grid = {
        'n_neighbors': [3, 5, 7, 9],      # Number of neighbors to consider
        'weights': ['uniform', 'distance'], # Weighting scheme
        'p': [1, 2]                        # Minkowski distance power parameter
    }
```

```
[ ] grid_search = GridSearchCV(estimator=model5, param_grid=param_grid, cv=5)
    grid_search.fit(x_train_smote,y_train_smote)
```

```
GridSearchCV
└─ estimator: KNeighborsClassifier
    └─ KNeighborsClassifier
```

```
[ ] best_params = grid_search.best_params_
    best_estimator = grid_search.best_estimator_
    print("Best Parameters:", best_params)
```

Best Parameters: {'n_neighbors': 3, 'p': 2, 'weights': 'distance'}

```
[ ] test_accuracy = best_estimator.score(x_test, y_test)
    print("Test Set Accuracy:", test_accuracy)
```

Test Set Accuracy: 0.8239486813970064

```
[ ] y_pred = best_estimator.predict(x_test)
    accuracy_knn = accuracy_score(y_test, y_pred)
    print("Accuracy:", accuracy_knn)
```

Accuracy: 0.8239486813970064

```
[ ] report = classification_report(y_test, y_pred)
    print("Classification Report:\n", report)
```

```
Classification Report:
              precision    recall  f1-score   support

     0           0.97       0.85      0.90       1355
     1           0.03       0.15      0.05         48

 accuracy          0.82          0.82          0.82       1403
 macro avg         0.50          0.50          0.48       1403
 weighted avg      0.93          0.82          0.87       1403
```

Activity 3.6 Random Forest Classifier without GridSearchCV:

A function named “model3” is created, and train and test data are passed as the parameters. Inside the function, “Randomforest” algorithm is initialized and training data is passed to the model with fit() function. Test data is predicted with predict() function and saved in a new variable. For evaluating the model accuracy score, confusion matrix and classification reports are used.

Random Forest Without GridSearch CV

```
[ ] model6 = RandomForestClassifier(criterion='entropy')
```

```
[ ] model6.fit(x_train_smote,y_train_smote)
```

```
RandomForestClassifier
RandomForestClassifier(criterion='entropy')
```

```
▶ y_pred= model6.predict(x_test)
  accuracy_rtwgscv = accuracy_score(y_test,y_pred)
  print("Accuracy:", accuracy_rtwgscv)
```

Accuracy: 0.925160370634355

```
[ ] pd.crosstab(y_test,y_pred)
```

```
col_0    0    1
class
0      1290  65
1         40   8
```

```
[ ] report = classification_report(y_test, y_pred)
  print("Classification Report:\n", report)
```

```
Classification Report:
              precision    recall  f1-score   support

     0       0.97      0.95      0.96      1355
     1       0.11      0.17      0.13         48

 accuracy          0.93      1403
 macro avg         0.54      0.56      0.55      1403
 weighted avg      0.94      0.93      0.93      1403
```

Milestone 5 : Performance Testing

Activity 1: Comparing all the Models.

For comparing the above six models, the accuracy_df dataframe is used.

We choose the Random forest with GridSearchCV model as it has higher accuracy as compared to the rest excepting ANN but has lesser number of False Negative errors as compared to ANN which can be risky.

```
[ ] accuracy_df = pd.DataFrame({
    'Models': ['SVM', 'Randomforest', 'ANN', 'Logistic_reg', 'KNN', 'Randomforestwohpt'],
    'Accuracy': [accuracy_svc*100, accuracy_rt*100, accuracy_ann*100, accuracy_lt*100, accuracy_knn*100, accuracy_rtwgscv*100]
})

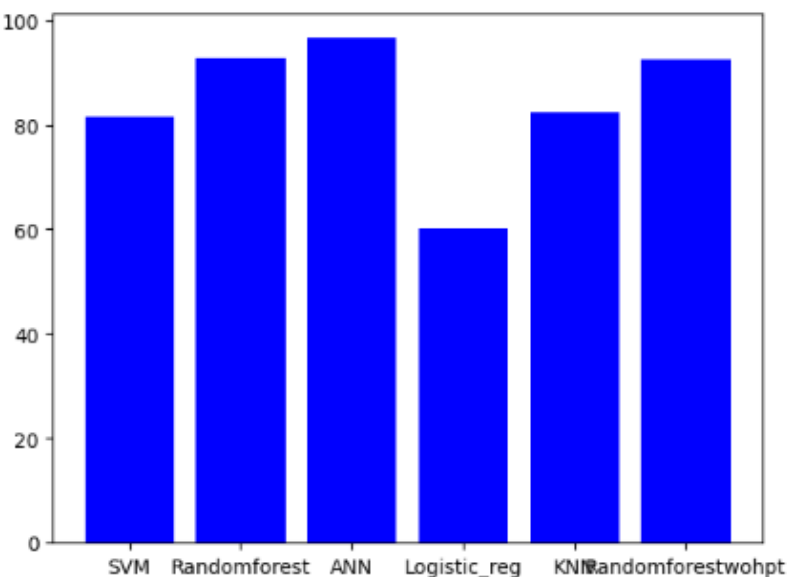
print(accuracy_df)
```

	Models	Accuracy
0	SVM	81.539558
1	Randomforest	92.658589
2	ANN	96.578760
3	Logistic_reg	60.014255
4	KNN	82.394868
5	Randomforestwohpt	92.516037

Activity 2: Graphical representation of the model comparison.

```
[ ] models = ['SVM', 'Randomforest', 'ANN', 'Logistic_reg', 'KNN', 'Randomforestwohpt']
accuracies = [accuracy_svc*100, accuracy_rt*100, accuracy_ann*100, accuracy_lt*100, accuracy_knn*100, accuracy_rtwgscv*100]
plt.bar(models, accuracies, color='blue')
```

<BarContainer object of 6 artists>



Milestone 6: Model Deployment

Activity 1: Save and load the best model

Saving the best model after comparing its performance using different evaluation metrics means selecting the model with the highest performance. This can be useful in avoiding the need to retrain the model every time it is needed and also to be able to use it in the future.

▾ Saving the Model

```
[ ] pickle.dump(model3,open('project.pkl','wb'))
```

We save the model using the pickle library into a file named model.pkl

Activity 2: Test the model:

Let's test the model first in python notebook itself.

As we have 10 features in this model, let's check the output by giving all the inputs.

▸ Testing the Saved Model

```
[ ] with open(r'/content/project.pkl', 'rb') as file:  
    loaded_model = pickle.load(file)
```

```
[ ] feature_names=['Attr1','Attr2','Attr3','Attr4','Attr5','Attr6','Attr7','Attr8','Attr9','Attr10']  
    for i in 1:  
        print(i)  
        input_data = pd.DataFrame([i], columns=feature_names)  
  
        # Use the loaded model to make predictions  
        predictions = loaded_model.predict(input_data)  
        print(predictions)  
        print()
```

Hence, we can conclude that our model is giving the accurate results.

Activity 3: Integrate with Web Framework

In this section, we will be building a web application that is integrated to the model we built. A UI is provided for the users where he must enter the values for predictions. The entered values are given to the saved model and prediction is showcased on the UI.

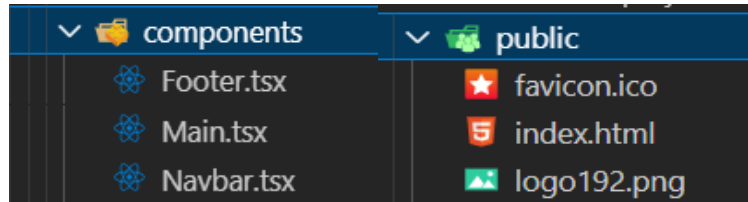
This section has the following tasks:

- Building HTML Pages
- Building server-side script
- Run the web application.

Activity 3.1: Building HTML and React pages:

For this project create two HTML files namely and save them in the templates folder.

- Index.html
- Footer.tsx
- Main.tsx
- Navbar.tsx



Activity 3.2: Build Python Code.

Create a new app.py file which will be stored in the Flask folder.

- Import the necessary Libraries.

Render HTML page:

```
TS Main.tsx x
antibankruptcy > src > components > TS Main.tsx > Main
1  import React, { useEffect, useState } from "react";
2
3  function Main() {
4    const [data, setData] = useState<number[]>();
5    const [showPopup, setPopup] = useState(false);
6    const [result, setResult] = useState(0);
7
8  > const CallModel = async () => { ...
41  };
42
43  > const fields = [ ...
54  ];
55
56  > useEffect(()=>{ ...
58  },[result])
59
60  return (
61    <
62      <div className="container m-5">
63        <div className="display-5">Bankruptcy Detection Model</div>
64        <hr />
65        <p>
66          Are you worried about the financial health of your business or
67          investments?
68          <br />
69          <br /> PredictRight is your trusted partner in making informed
70          decisions about bankruptcy risk. Our cutting-edge bankruptcy
71          prediction application leverages the power of data analytics and
72          machine learning to help you anticipate financial distress before it
73          becomes a crisis.
74        </p>
75        <div className="info"></div>
76        <div className="form border p-5">
77          <form className="w-100">
78            <div className="gap-5 d-flex justify-content-start flex-wrap">
```

```

3 import sklearn
4 from sklearn.preprocessing import StandardScaler
5 from flask import Flask, request, render_template
6 from flask_cors import CORS
7 import pandas as pd
8
9 app = Flask(__name__)
10 CORS(app)
11 # Dummy data to store posted values
12 data = []
13
14 @app.route('/')
15 def index():
16     return "Welcome to Anti Bankruptcy"
17
18 @app.route('/model/input', methods=['POST'])
19 def model_input():
20     # Handle the form submission
21     data = request.get_json()
22
23     try:
24         with open('project.pkl', 'rb') as file:
25             column_names= ["Attr1","Attr2","Attr3","Attr4","Attr5","Attr6","Attr7","Attr8","Attr9","Attr10"]
26             loaded_model = pickle.load(file)
27             input_data = pd.DataFrame([data["Input"]], columns=column_names)
28
29             # Use the loaded model to make predictions
30             predictions = loaded_model.predict(input_data)
31             return f'{predictions}'
32     except Exception as e:
33         print(f"Error: {str(e)}")
34         return "Error"
35
36 if __name__ == '__main__':
37     app.run(debug=True)

```

The code loads a pre-trained machine learning model which is a Random Forest model, from the pickle file and provides two routes.

The home route ("/") displays a welcome message, while the "/model/input" route handles POST requests, receiving user input in JSON format.

The input data is then used to make bankruptcy predictions using the model. The results are returned as a response. The code also includes error handling to address exceptions during model loading or prediction. When executed, the Flask application runs in debug mode, allowing for testing and development.

Activity 4: GUI:

The GUI (Graphical User Interface) created in this Flask application is designed to predict the bankruptcy of the company bases on below features:

- Net Profit
- Total Liabilities/ Total Assets
- Working Capital
- Current Assets
- Cashflow Coverage Ratio
- Retained Earnings
- EBIT
- Book Value of Equity
- Sales
- Equity

The user can input this feature in a form provided in the home page of the web application. After clicking on the "Test Bankruptcy" button, the application will predict the level of freedom of that country based on the rules and the random forest model defined in the Python script.

Bankruptcy Detection Model

Are you worried about the financial health of your business or investments?

PredictRight is your trusted partner in making informed decisions about bankruptcy risk. Our cutting-edge bankruptcy prediction application leverages the power of data analytics and machine learning to help you anticipate financial distress before it becomes a crisis.

Net Profit

Total Liabilities/ Total Assets

Working Capital

Current Assets

Cashflow Coverage Ratio

Retained Earnings

EBIT

Book Value of Equity

Sales

Equity

PredictRight is your trusted partner in making informed decisions about bankruptcy risk. Our cutting-edge bankruptcy prediction application leverages the power of data analytics and machine learning to help you anticipate financial distress before it becomes a crisis.

Net Profit

Total Liabilities/ Total Assets

Working Capital

Current Assets

Cashflow Coverage Ratio

Retained Earnings

EBIT

Book Value of Equity

Sales

Equity

Test Bankruptcy

Class values for binary classification:

- 0 - company that did not bankrupt in the forecasting period
- 1 - bankrupted company

Milestone 7: Project Demonstration & Documentation

Below mentioned deliverables to be submitted along with other deliverables.

Activity 1: Record explanation Video for project end to end solution.

Activity 2: Project Documentation-Step by step project development procedure