Anticipating Business Bankruptcy

Project Report Documentation

Team: Hrishikesh G Kulkarni, Anirudh Soma,Ayush

Table of Contents

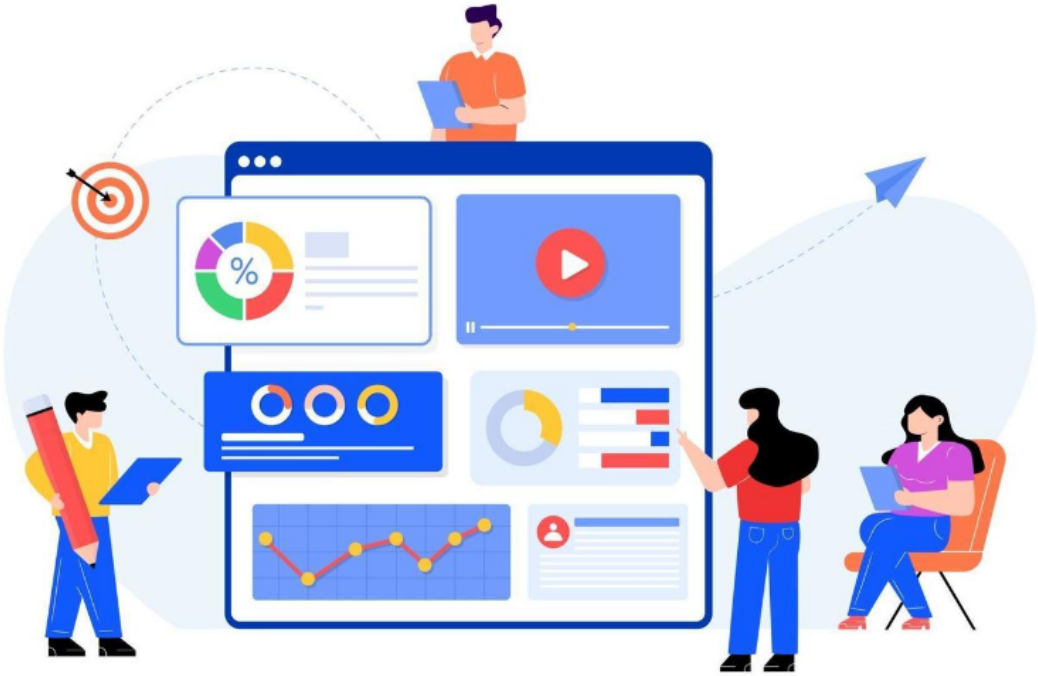
1.INTRODUCTION

1.1 Project Overview

Bankruptcy forecasting has been around for nearly a century and remains one of the most important topics in economics. The aim of this study is to create a predictive model that combines various economic indicators to predict the financial health of a company. By assessing the company's financial health and future prospects, we can gain valuable insight into the company's long-term market performance.

1.2 Purpose

The aim of this project is to analyze the predictors of corporate bankruptcy. By examining financial ratios and other factors that indicate a risk of bankruptcy , we aim to understand the causes of financial risks and develop a predictive model that can predict the financial status of the company. The insights gained from this study will help stakeholders make decisions and implement strategies to reduce financial risks via the predictions based on provided datasets.



2.LITERATURE SURVEY

2.1 Existing Problem

Bankruptcy prediction has long been a topic of interest in economics because it has implications for investors, borrowers, and other stakeholders . Despite the abundance of research in this field, the task of predicting financial crises remains a challenge. Current approaches are based on a combination of financial ratios, statistical models and machine learning techniques. However, the accuracy and reliability of bankruptcy prediction models must continue to be improved.

2.2 References

1.Altman, E. I. (1968). Financial ratios, discriminant analysis and the prediction of corporate bankruptcy. The Journal of Finance, 23(4), 589-609.

2.Ohlson, J. A. (1980). Financial ratios and the probabilistic prediction of bankruptcy. Journal of Accounting Research, 18(1), 109-131.

3.Zmijewski, M. E. (1984). Methodological issues related to the estimation of financial distress prediction models. Journal of Accounting Research, 22(1), 59-82.

4.Taffler, R. J. (1984). Empirical models for the monitoring of UK corporations. Journal of Banking & Finance, 8(2), 199-227.

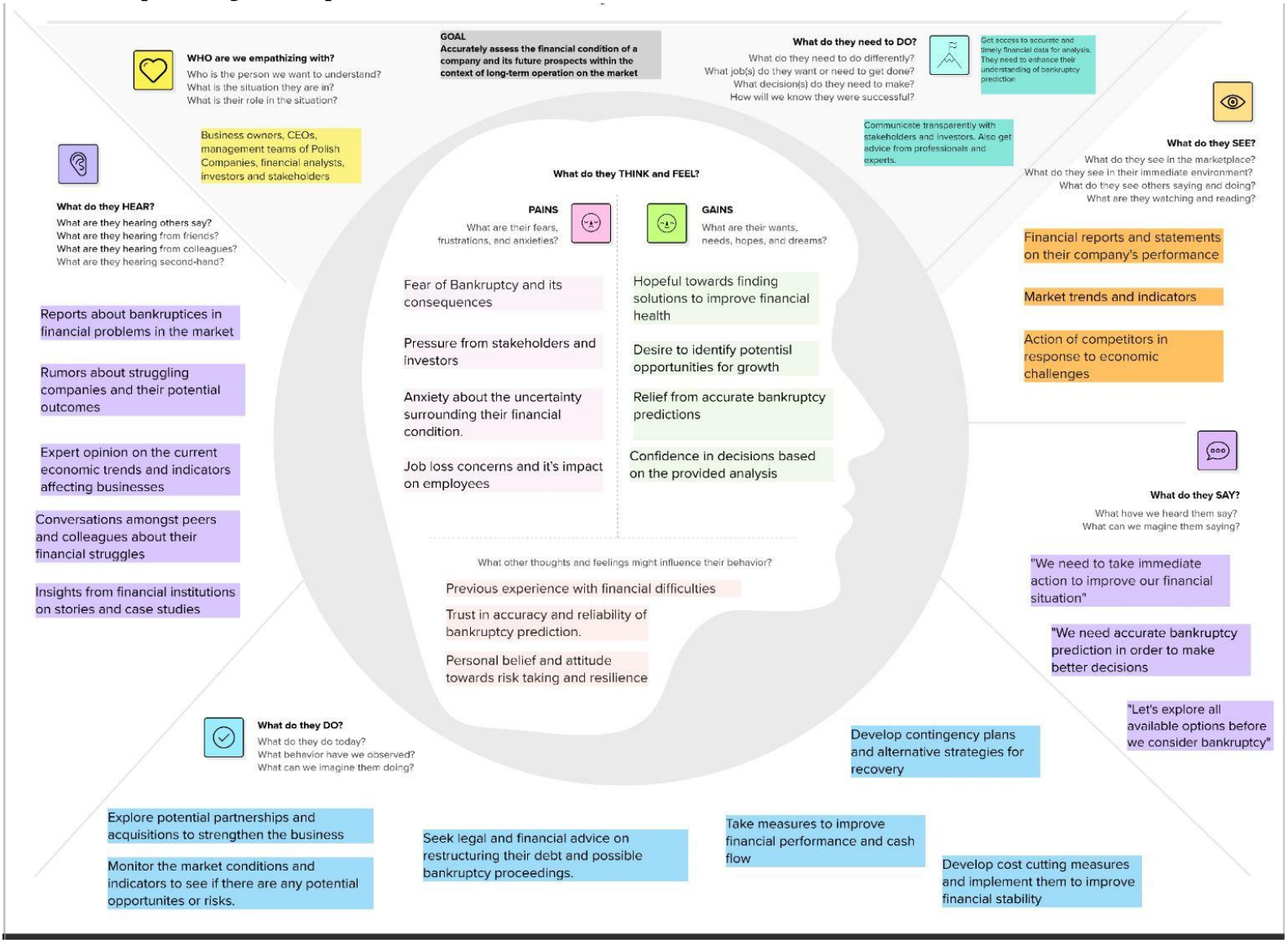
5.Shumway, T. (2001). Forecasting bankruptcy more accurately: A simple hazard model. The Journal of Business, 74(1), 101-124

2.3 Problem Statement Definition

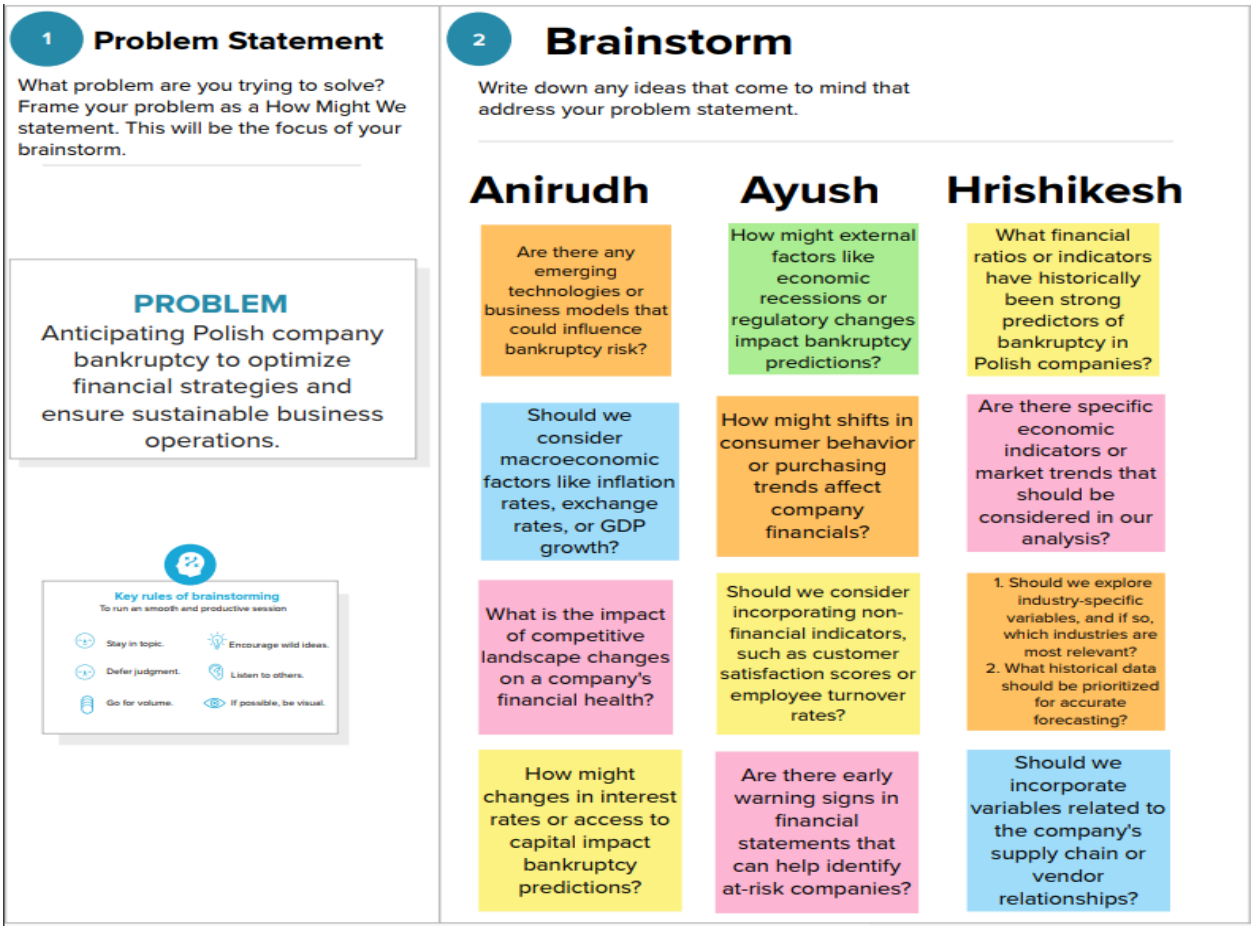
The problem addressed in this project is the accurate prediction of bankruptcy for companies. We aim to develop a predictive model that combines various factors and measures to effectively assess the financial condition of firms. The challenge lies in identifying the key factors and patterns that distinguish bankrupt companies from those that continue to operate successfully. The ultimate goal is to provide stakeholders with reliable insights into the financial health and long-term prospects of companies, enabling them to make informed decisions in the market.

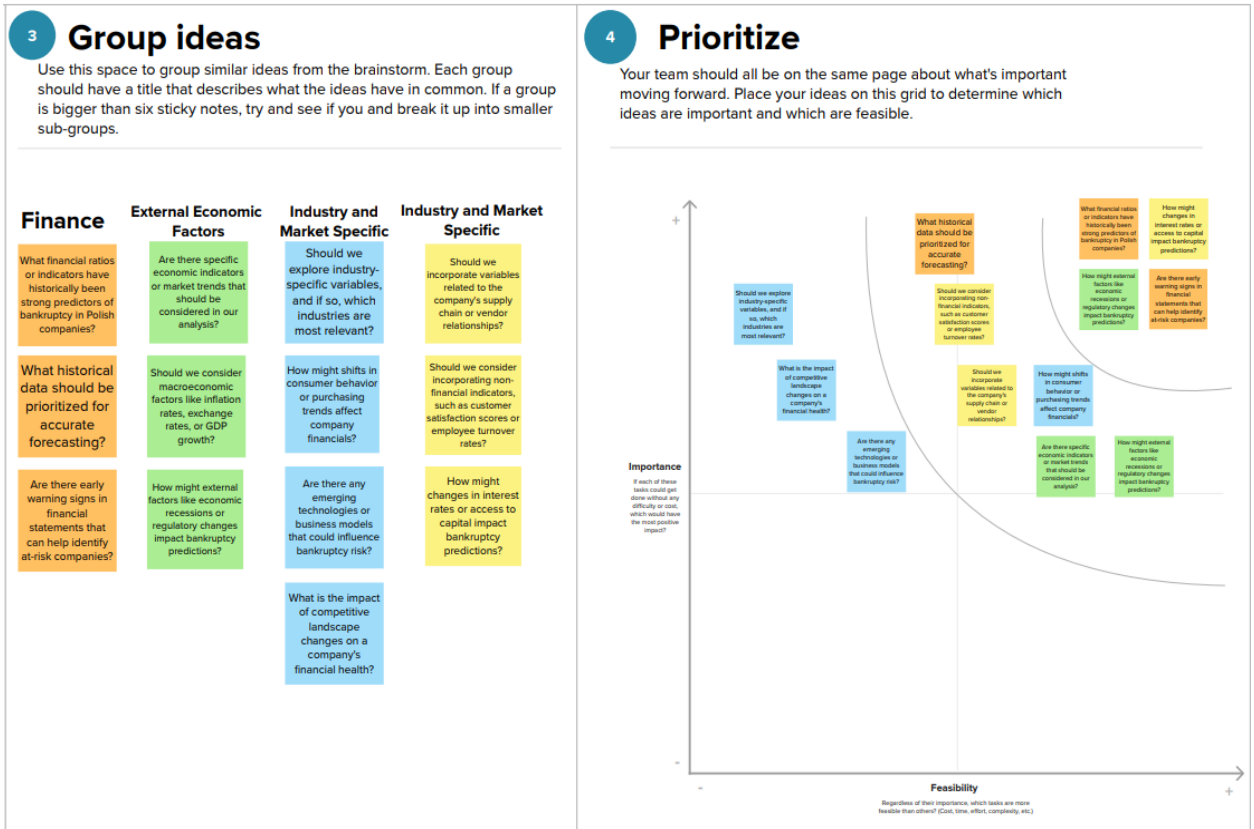
3.IDEATION AND PROPOSED SOLUTION

3.1 Empathy Map Canvas



3.2 Ideation and Brainstorming





4.REQUIREMENT ANALYSIS

4.1 Functional Requirements

> The system should provide the capability to select relevant features for the model
> The system should support choosing the appropriate machine learning model.
> The system should facilitate the training of the selected model.
> The system should allow the evaluation of the performance of the trained model.
> The system should generate reports and visualize the results of the model.
> The system should develop a user interface for displaying the results.
> The system should gather and prepare data for analysis.
> The system should clean the data to remove inconsistencies.
> The system should create new features from existing data.
> The system should save the trained model for later use.
> The system should integrate the model with an application.
> The system should design the user interface for the application.

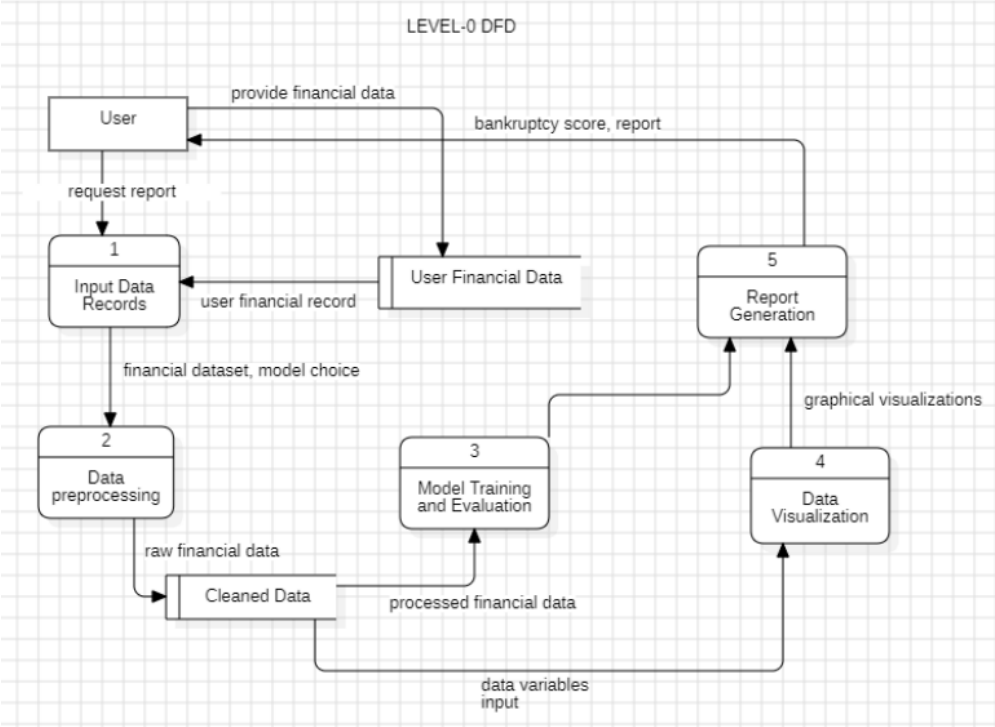
4.2 Non Functional Requirements

> The model should have the ability to make correct predictions.
> The application should have a quick response time.
> The system should efficiently utilize system resources.
> The application should be able to handle different scenarios and inputs.
> The system should be able to handle increasing workload or user base.
> The application should have consistency and stability.

5.PROJECT DESIGN

5.1 Data Flow Diagrams and User Stories

Data Flow Diagram



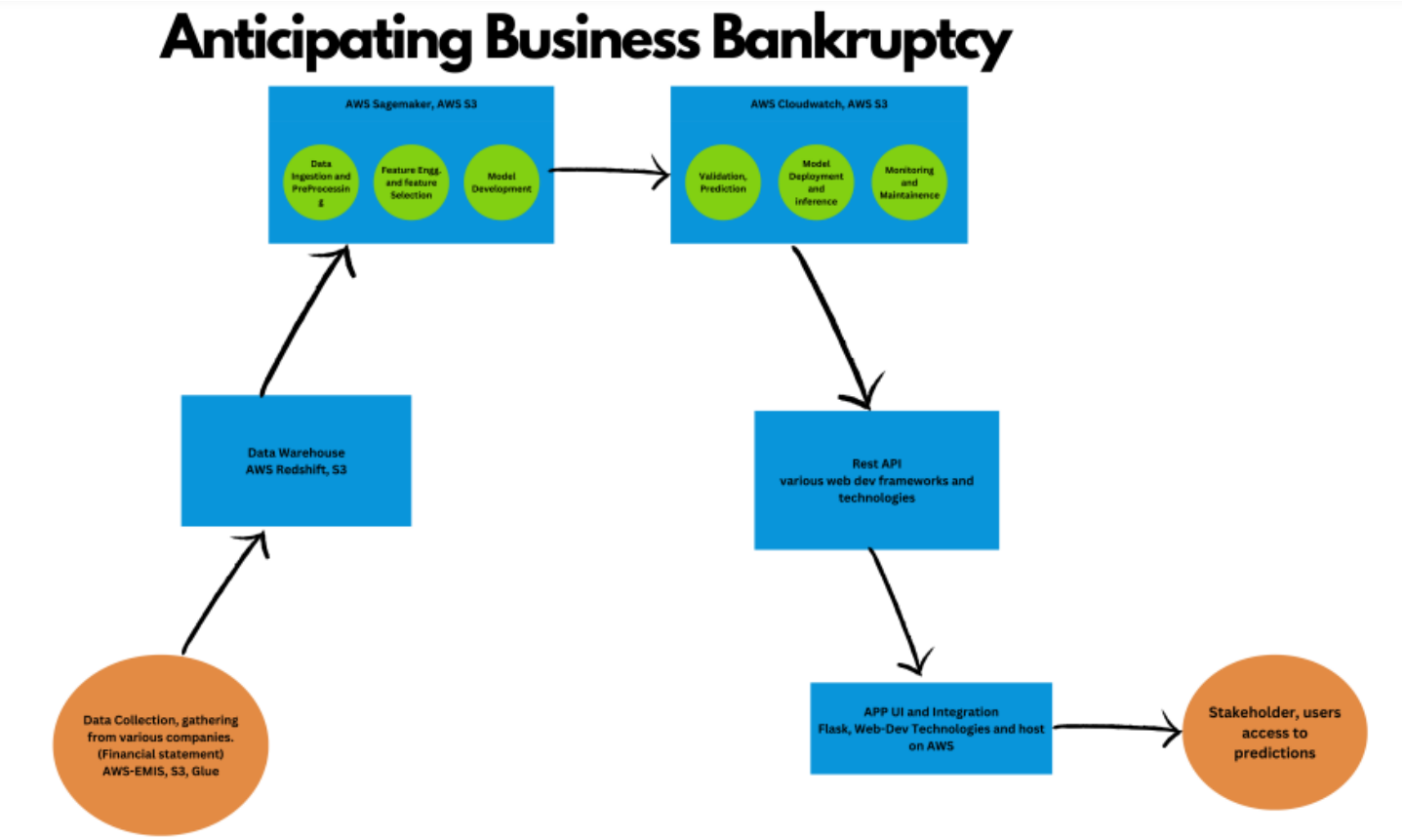
User Case Stories

USER CASE STORIES

User Type	Functional Requirement	User Story Number	User Story/Task	Acceptance Criteria	Priority	Release
Credit Analyst	Generate Bankruptcy Risk Report	USN-1	As a credit analyst, I want to generate a bankruptcy risk report for a company to assess its legitimacy	Users can input the company's financial data and get the risk score	High	Sprint 1
Individual Investor	Portfolio Rebalancing based on Bankruptcy Risk	USN-2	As an individual investor, I want a risk score to balance my portfolio based on the bankruptcy risk of constituent companies.	The system retrieves the bankruptcy risk scores of the companies in the portfolio.	Medium	Sprint 2
Mergers and Acquisitions Analyst	Perform Due Diligence on Target Company	USN-3	As an MA analyst, I want an insight on a target company to assess its financial health and bankruptcy risk.	The report provides insights into the target company's financial health and assists in decision-making for the acquisition.	High	Sprint 1
Risk Manager	Set Bankruptcy Risk Thresholds	USN-4	As a risk manager, I want to set bankruptcy risk thresholds for different types of businesses to monitor their financial stability.	The system allows the risk manager to define bankruptcy risk thresholds based on	Medium	Sprint 2

6.PROJECT PLANNING AND SCHEDULING

6.1 Technical Architecture



6.2 Sprint Planning and Estimation

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	5	High	Hrishikesh, Anirudh, Ayush
Sprint-1	Registration	USN-2	As a user, I will receive a confirmation email once I have registered for the application.	3	High	Hrishikesh, Anirudh,
Sprint-1	Login	USN-3	As a user, I can log into the application by entering my email and password.	5	High	Hrishikesh, Ayush
Sprint-2	Dashboard	USN-4	As a user, I can access a dashboard that provides an overview of my financial data and bankruptcy predictions.	8	Medium	Anirudh, Ayush
Sprint-2	Data Collection	USN-5	As a user, I can input financial data for my company and request a bankruptcy prediction.	7	High	Hrishikesh, Anirudh, Ayush
Sprint-2	Dashboard	USN-6	As a user, I can view the historical performance of my company's financial health.	5	Medium	Anirudh, Ayush
Sprint-3	Dashboard	USN-7	As a user, I can receive alerts and recommendations based on the bankruptcy prediction results.	8	High	Hrishikesh, Ayush
Sprint-3	Documentation	USN-8	As a user, I can access a user guide and documentation to understand how the bankruptcy prediction system works.	3	Low	Anirudh

6.3 Sprint Delivery Schedule

Project Tracker, Velocity & Burndown Chart: (4 Marks)

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	13	6 Days	24 Oct 2022	29 Oct 2022	13	29 Oct 2022
Sprint-2	20	6 Days	31 Oct 2022	05 Nov 2022	20	05 Nov 2022
Sprint-3	11	6 Days	07 Nov 2022	12 Nov 2022	11	12 Nov 2022

To calculate the team's average velocity (AV) per iteration unit (story points per day:
AV = (Total Story Points) / (Duration in Days)

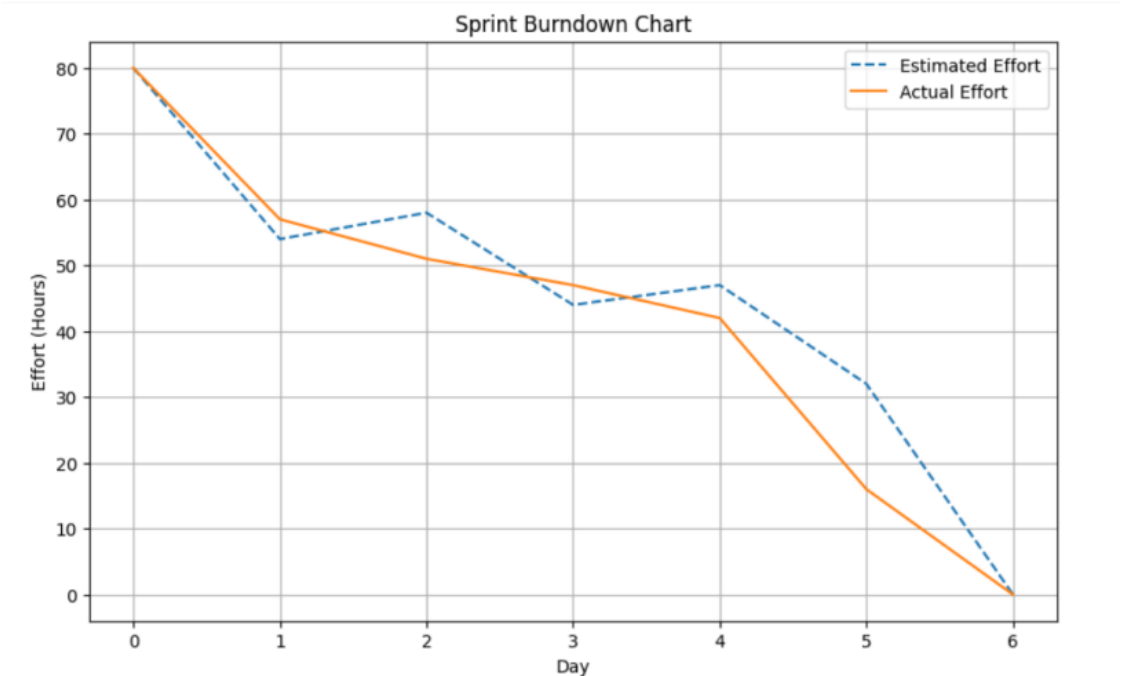
Let's calculate the average velocity for the given sprints:

1. **Sprint-1:**
   - Total Story Points: 13
   - Duration: 6 days
   - AV = 13 / 6 = 2.17 (rounded to two decimal places)
2. **Sprint-2:**
   - Total Story Points: 20
   - Duration: 6 days
   - AV = 20 / 6 = 3.33 (rounded to two decimal places)
3. **Sprint-3:**
   - Total Story Points: 11
   - Duration: 6 days
   - AV = 11 / 6 = 1.83 (rounded to two decimal places)

The average velocity for each of the given sprints is as follows:
- Sprint-1: 2.17 story points per day
- Sprint-2: 3.33 story points per day
- Sprint-3: 1.83 story points per day

These values represent our team's performance in terms of story points completed per day for each sprint.

Burndown Chart:



7.CODING AND SOLUTIONS

7.1 Data Cleaning and Processing

1.Importing necessary modules

```
[ ] import pandas as pd
    import numpy as np
    import matplotlib.pyplot as plt
    import seaborn as sbn
    from scipy.stats import zscore
```

2.Loading the dataset

```
[ ] df = pd.read_csv('1year.csv')
    df.head()
```

	Attr1	Attr2	Attr3	Attr4	Attr5	Attr6	Attr7	Attr8	Attr9	Attr10	...	Attr56	Attr57	Attr58	Attr59	Attr60	Attr61	Attr62	Attr63	Attr64	class
0	0.20055	0.37951	0.39641	2.0472	32.351	0.38825	0.24976	1.3305	1.1389	0.50494	...	0.121960	0.39718	0.87804	0.001924	8.416	5.1372	82.658	4.4158	7.4277	0
1	0.20912	0.49988	0.47225	1.9447	14.786	0	0.25834	0.99601	1.6996	0.49788	...	0.121300	0.42002	0.85300	0	4.1486	3.2732	107.350	3.4	60.987	0
2	0.24866	0.69592	0.26713	1.5548	-1.1523	0	0.30906	0.43695	1.309	0.30408	...	0.241140	0.81774	0.76599	0.69484	4.9909	3.951	134.270	2.7185	5.2078	0
3	0.081483	0.30734	0.45879	2.4928	51.952	0.14988	0.092704	1.8661	1.0571	0.57353	...	0.054015	0.14207	0.94598	0	4.5746	3.6147	86.435	4.2228	5.5497	0
4	0.18732	0.61323	0.2296	1.4063	-7.3128	0.18732	0.18732	0.6307	1.1559	0.38677	...	0.134850	0.48431	0.86515	0.12444	6.3985	4.3158	127.210	2.8692	7.898	0

5 rows x 65 columns

```
[ ] df.shape
```

```
(7012, 65)
```

```
[ ] df.info()
```

```
 9   Attr10  7012 non-null   object
 10  Attr11  7012 non-null   object
```

```
[ ] df.describe()
```

	Attr13	Attr19	Attr20	Attr23	Attr30	Attr31	Attr39	Attr42	Attr43	Attr44	Attr49	Attr55	Attr56	Attr58	Attr62	class
count	7012.000000	7012.000000	7.012000e+03	7012.000000	7012.000000	7012.000000	7012.000000	7012.000000	7.012000e+03	7.012000e+03	7012.000000	7.012000e+03	7.012000e+03	7.012000e+03	7.012000e+03	7012.000000
mean	2.097737	0.463352	1.164518e+03	0.424494	23.618569	0.474681	0.269890	0.264140	4.598113e+03	3.433623e+03	-1.370293	8.871752e+03	-1.580742e+02	1.590784e+02	4.773179e+03	0.036509
std	159.870167	30.679634	9.325781e+04	29.773895	1828.800876	30.680472	27.093165	27.100065	3.631187e+05	2.699093e+05	107.833017	7.255154e+04	1.323538e+04	1.323537e+04	3.111157e+05	0.187566
min	-607.420000	-622.060000	0.000000e+00	-634.590000	-149.070000	-622.060000	-701.630000	-701.630000	0.000000e+00	0.000000e+00	-9001.000000	-8.004700e+05	-1.108300e+06	-4.194000e-03	0.000000e+00	0.000000
25%	0.038130	0.014867	1.708150e+01	0.011360	0.094599	0.019336	0.012792	0.011542	6.527350e+01	3.389925e+01	-0.009971	1.017100e+02	2.035400e-02	8.646675e-01	4.320550e+01	0.000000
50%	0.080909	0.050059	3.563500e+01	0.042140	0.206880	0.056767	0.047011	0.047891	9.461200e+01	5.180600e+01	0.022800	1.607050e+03	6.343050e-02	9.387550e-01	6.850900e+01	0.000000
75%	0.145000	0.104640	6.070000e+01	0.088390	0.365590	0.115022	0.103553	0.102215	1.304425e+02	7.549225e+01	0.075505	5.964150e+03	1.377800e-01	9.818700e-01	1.062575e+02	0.000000
max	13315.000000	2156.800000	7.809200e+06	2156.800000	152860.000000	2156.800000	2156.500000	2156.800000	3.039300e+07	2.258400e+07	31.639000	4.398400e+06	1.000000e+00	1.108300e+06	2.501600e+07	1.000000

3.Dropping to keep only the first 10 attributes as per document

```
[ ] df.drop(df.iloc[:, 10:64], inplace=True, axis=1)
```

```
[ ] df.head()
```

	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
2	0.24866	0.69592	0.26713	1.5548	-1.1523	0	0.30906	0.43695	1.309	0.30408	0
3	0.081483	0.30734	0.45879	2.4928	51.952	0.14988	0.092704	1.8661	1.0571	0.57353	0
4	0.18732	0.61323	0.2296	1.4063	-7.3128	0.18732	0.18732	0.6307	1.1559	0.38677	0

4.Missing values

```
[ ] # Checking for missing values
    missing_values = df.isnull().sum()
    print(missing_values)
```

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

No missing values!

```
[ ] (df.eq('?')).any()
```

```
Attr1     True
Attr2     True
Attr3     True
Attr4     True
Attr5     True
Attr6     True
Attr7     True
Attr8     True
Attr9     True
Attr10    True
```

```
[ ] (df.eq('?')).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
```

Filling '?' with NaN and then doing the graphical analysis / EDA

```
[ ] # Checking for ? values and fill them
    df.replace('?',np.NAN,inplace=True)

[ ] (df.eq('?')).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
```

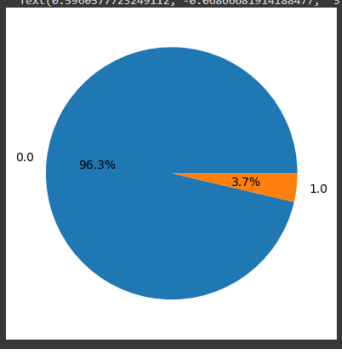
Filling '?' with NaN and then doing the graphical analysis / EDA

```
[ ] # Checking for ? values and fill them
    df.replace('?',np.NAN,inplace=True)

[ ] (df.eq('?')).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

[ ] 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
```

7.2 Analyzing Data: Univariate, Bivariate and Multivariate Data

Univariate Data

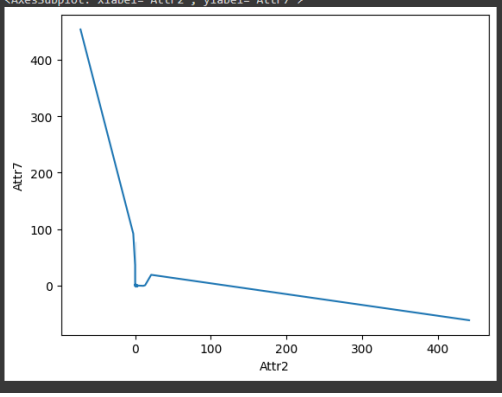
```
[ ] plt.pie(df["class"].value_counts(), labels = df["class"].unique(),autopct ='%1.1f%%')

([<matplotlib.patches.Wedge at 0x1c1e5671060>,
  <matplotlib.patches.Wedge at 0x1c1e5671510>],
 [Text(-1.0927725884889667, 0.12588911727041197, '0.0'),
  Text(1.0927725825956707, -0.12588916842678877, '1.0')],
 [Text(-0.5960577755394363, 0.06866679123840651, '96.3%'),
  Text(0.5960577723249112, -0.06866681914188477, '3.7%')])
```



Bivariate Data

```
sbn.lineplot(x=df['Attr2'],y=df['Attr7'])
```

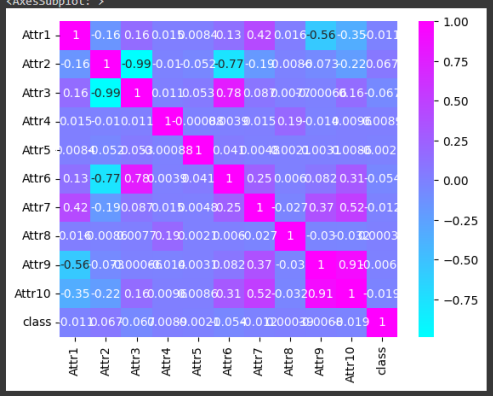
```
<AxesSubplot: xlabel='Attr2', ylabel='Attr7'>
```



Multivariate Data

```
[ ] sbn.heatmap(df.corr(),cmap="cool",annot=True)
```

```
<AxesSubplot: >
```

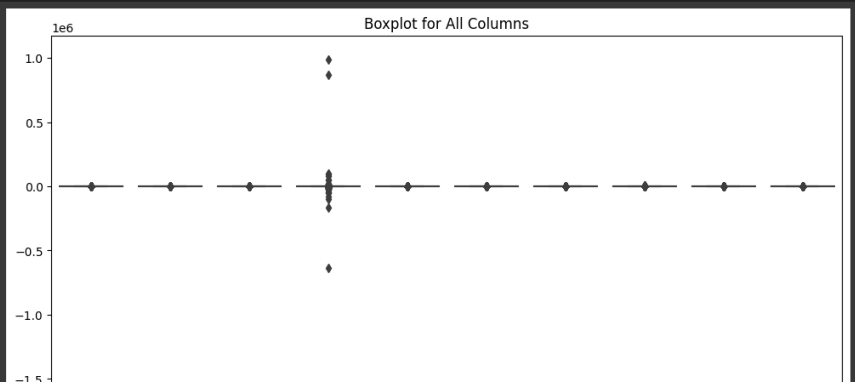


7.2 Removing Outliers in Data

```
7.Outlier Removal
```

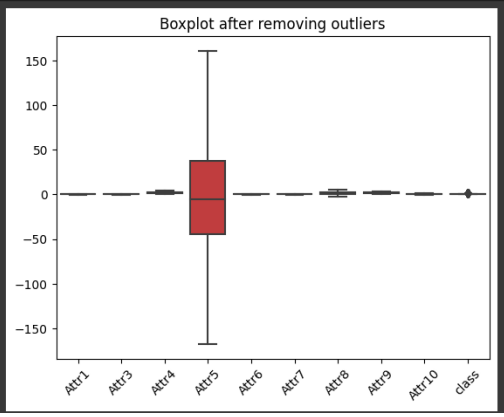
```
+ Code  + Text
```

```
[ ] # Making a boxplot to see if there are any outliers
    plt.figure(figsize=(12, 8))
    sbn.boxplot(data=df)
    plt.xticks(rotation=45)
    plt.title('Boxplot for All Columns')
    plt.show()
```



Checking if outliers are handled

```
cols = ["Attr1","Attr3","Attr4","Attr5","Attr6","Attr7","Attr8","Attr9","Attr10"]
for i in cols:
  q1 = df[i].quantile(0.25)
  q3 = df[i].quantile(0.75)
  iqr = q3 - q1
  uL = q3 + 1.5*iqr
  lL = q1 - 1.5*iqr
  df[i] = np.where(df[i]>uL,uL,np.where(df[i]<lL,lL,df[i]))
sbn.boxplot(data=df)
plt.xticks(rotation=45)
plt.title('Boxplot after removing outliers')
plt.show()
```



7.3 Scaling and Training

8.Scaling,Testing and Training

```
from sklearn.preprocessing import MinMaxScaler
from imblearn.over_sampling import SMOTE
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

X = df.drop("class", axis=1)
y = df["class"]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

smote = SMOTE(random_state=42)
X_train_resampled, y_train_resampled = smote.fit_resample(X_train, y_train)

#scale
scaler = MinMaxScaler()
X_train_scaled = scaler.fit_transform(X_train_resampled)
X_test_scaled = scaler.transform(X_test)
```

```
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.metrics import classification_report
```

```
from sklearn.ensemble import RandomForestClassifier
rfc=RandomForestClassifier(criterion='entropy')
rfc.fit(X_train_scaled, y_train_resampled)
y_pred1_train = rfc.predict(X_train_scaled)
y_pred1_test = rfc.predict(X_test_scaled)
```

7.4 Testing and Accuracy of Models

Testing and Accuracy for all models are tested, see BankruptcyDetection.ipynb

9.Testing and Accuracy

```
print('Training Accuracy for RandomForest = ', accuracy_score(y_train_resampled,y_pred1_train))
```

```
Training Accuracy for RandomForest =  0.9999074759437454
```

```
print('Testing Accuracy for RandomForest = ', accuracy_score(y_test,y_pred1_test))
```

```
Testing Accuracy for RandomForest =  0.9151817533856023
```

```
print("====================Random Forest Classifier====================")
print(classification_report(y_test, y_pred1_test))
from sklearn.metrics import confusion_matrix
import matplotlib.pyplot as plt


cm=confusion_matrix(y_test, y_pred1_test)
# Create a heatmap using Seaborn
sbn.heatmap(cm, annot=True)
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()
```

```
====================Random Forest Classifier====================
              precision    recall  f1-score   support

         0.0       0.97      0.94      0.96      1352
         1.0       0.12      0.22      0.16        51

    accuracy                           0.92      1403
   macro avg       0.55      0.58      0.56      1403
weighted avg       0.94      0.92      0.93      1403
```

Accuracy comparisons

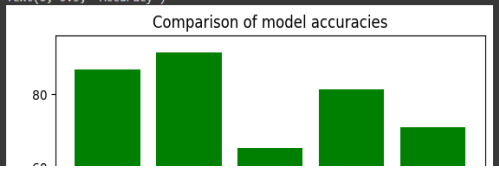
```
acc_DT = accuracy_score(y_test, y_pred5_test)
acc_RF = accuracy_score(y_test, y_pred1_test)
acc_KNN = accuracy_score(y_test, y_pred2_test)
acc_SVM = accuracy_score(y_test, y_pred4_test)
acc_LR = accuracy_score(y_test, y_pred3_test)
```

```
accuracy_df = pd.DataFrame({
    'Model': ['DecisionTree', 'RandomForest', 'LogisticRegression', 'KNN', 'SVM'],
    'Accuracy': [acc_DT*100, acc_RF*100, acc_LR*100, acc_KNN*100, acc_SVM*100]
})
print(accuracy_df)
```

```
                Model   Accuracy
0        DecisionTree  86.671418
1        RandomForest  91.518175
2  LogisticRegression  65.146115
3                 KNN  81.254455
4                 SVM  70.990734
```

```
models = ['SVM', 'DecisionTree', 'RandomForest', 'LogisticRegression', 'KNN']
accuracies = [acc_DT*100, acc_RF*100, acc_LR*100, acc_KNN*100, acc_SVM*100]
plt.bar(models, accuracies, color='green')
#add title and axis labels
plt.title('Comparison of model accuracies')
plt.xlabel('Models')
plt.ylabel('Accuracy')
```

```
Text(0, 0.5, 'Accuracy')
```



7.5 Saving Models

10.Saving models

```
[ ] #saving models with pickle
    import pickle
    with open('decision_tree.pkl', 'wb') as file:
        pickle.dump(dt, file)

    with open('random_forest_model.pkl', 'wb') as file:
        pickle.dump(rfc, file)

    with open('logistic_reg.pkl', 'wb') as file:
        pickle.dump(logReg, file)


    with open('svm.pkl', 'wb') as file:
        pickle.dump(svm, file)

    with open('knn.pkl', 'wb') as file:
        pickle.dump(knn, file)

    import joblib
    joblib.dump(scaler, 'scaler_model.joblib')
```

```
['scaler_model.joblib']
```

7.6 Deployment (main = app.py)

```
from flask import Flask, render_template, request, jsonify

import joblib

import numpy as np

import pandas as pd

from sklearn.preprocessing import MinMaxScaler



app = Flask(__name__)



# Load the model

model1 =
joblib.load(r"c:\Users\hrish\DataScience\SmartBridge_DS\Bankruptcy\knn.pkl")

model2 =
joblib.load(r"c:\Users\hrish\DataScience\SmartBridge_DS\Bankruptcy\svm.pkl")

model3 =
joblib.load(r"c:\Users\hrish\DataScience\SmartBridge_DS\Bankruptcy\decision_tree.pkl
")

model4 =
joblib.load(r"c:\Users\hrish\DataScience\SmartBridge_DS\Bankruptcy\logistic_reg.pkl"
)

model5 =
joblib.load(r"c:\Users\hrish\DataScience\SmartBridge_DS\Bankruptcy\random_forest_mod
el.pkl")
```

```python
@app.route('/')
def index():
    return render_template('indexDsn.html')



@app.route('/predict', methods=['POST'])
def predict():
    try:
        data = request.get_json(force=True)
        selected_model = data['selectedModel']
        attribute_values =data['attributeValues']
        attribute_values = np.array(attribute_values).reshape(1, -1)
        scaler =
joblib.load(r"c:\Users\hrish\DataScience\SmartBridge_DS\Bankruptcy\scaler_model.jobl
ib")
        attribute_values = scaler.transform(attribute_values)
        app.logger.info(f"Received prediction request for model: {selected_model}")


        if selected_model == 'model1':
            used_model = "Model 1 (KNN)"
            prediction=(model1.predict(attribute_values))
            prediction_label = "will be" if prediction[0] == 1 else "will not be"
            app.logger.info(f"Prediction using {used_model}: {prediction}")


        elif selected_model == 'model2':
            used_model = "Model 2 (SVM)"
            prediction=(model2.predict(attribute_values))
```

```
            prediction_label = "will be" if prediction[0] == 1 else "will not be"
            app.logger.info(f"Prediction using {used_model}: {prediction}")

        elif selected_model == 'model3':
            used_model = "Model 3 (Decision Tree)"
            prediction=(model3.predict(attribute_values))
            prediction_label = "will be" if prediction[0] == 1 else "will not be"
            app.logger.info(f"Prediction using {used_model}: {prediction}")

        elif selected_model == 'model4':
            used_model = "Model 4 (Logistic Regression)"
            prediction=(model4.predict(attribute_values))
            prediction_label = "will be" if prediction[0] == 1 else "will not be"
            app.logger.info(f"Prediction using {used_model}: {prediction}")

        elif selected_model == 'model5':
            used_model = "Model 5 (Random Forest)"
            prediction=(model5.predict(attribute_values))
            prediction_label = "will be" if prediction[0] == 1 else "will not be"
            app.logger.info(f"Prediction using {used_model}: {prediction}")

        else:
            return jsonify(error="Invalid model selected")

        return
jsonify(prediction=int(prediction[0]),usedModel=used_model,predictionLabel=predictio
n_label)
    except Exception as e:
        app.logger.error(f"Error in prediction: {str(e)}")
```

```
        return jsonify(error=str(e))


if __name__ == '__main__':
    app.run(debug=True)
```

7.8 Front End

indexDsn.html

```
<!DOCTYPE html>
<html lang="en">


<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Bankruptcy</title>
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css"
rel="stylesheet"

integrity="sha384-T3c6CoIi6uLrA9TneNEoa7RxnatzjcDSCmG1MXxSR1GAsXEV/Dwwykc2MPK8M2HN"
crossorigin="anonymous">
    <link rel="stylesheet" href="{{ url_for('static', filename='style.css') }}">
</head>


<body>
    <div class="container">
        <nav id="mainNavbar" class="navbar fixed-top" style="background-color:
#023047;">
            <div class="container-fluid">
```

```
                <h1 class="navbar-brand">Anticipating Business Bankruptcy</h1>
            </div>
        </nav>
    </div>
    <section class="container-fluid px-5 spacing">
        <div class="row align-items-center text-center">
            <div class="col-lg-5 col-md-6 px-lg-5">
                <div id="headingGrp" class="text-white text-center mt-5">
                    <span class="intro">
                        <h1 class="display-4">Bankruptcy Calculator</h1>
                    </span>
                    <span class="desc">
                        <h2 class="display-6">Helping you forecast the possibility
of Bankruptcy for your Business</h2>
                    </span>
                </div>
            </div>
            <div class="col-lg-7 col-md-6 px-lg-5">
                <img class="img-fluid"

src="https://www.iicle.com/images/thumbs/0002356_business-bankruptcy-practice-2022-e
dition_600.png"
                    alt="">
            </div>
        </div>
    </section>


    <section class="container-fluid p-0 mt-5">
        <div class="container">
```

```
            <div class="background-image" style="height: 100px;"></div>
        </div>
    </section>


    <section class="container mt-5">
        <div class="container">
            <span class="intro">
                <h1 class="display-4">Calculator for prediction</h1>
            </span>
            <span class="desc">
                <h1 class="display-6">Enter the inputs as asked and get your
required prediction</h1>
            </span>
        </div>
    </section>


    <section>
        <form id="predictionForm">
            <hr>
            <label for="selectedModel" class="form-label">Select Model:</label>
            <select id="selectedModel" name="selectedModel" class="form-control">
                <option value="model1">Model 1 (KNN)</option>
                <option value="model2">Model 2 (SVM)</option>
                <option value="model3">Model 3 (Decision Tree)</option>
                <option value="model4">Model 4 (Logistic Regression)</option>
                <option value="model5">Model 5 (Random Forest)</option>
            </select>
            <hr>
            <div class="mb-3">
```

```
                <label for="attribute1" class="form-label">Profitability
Ratio:</label>
                <input type="text" class="form-control" id="attribute1"
name="attribute1" required>
            </div>
            <div class="mb-3">
                <label for="attribute2" class="form-label">Leverage Ratio:</label>
                <input type="text" class="form-control" id="attribute2"
name="attribute2" required>
            </div>
            <div class="mb-3">
                <label for="attribute3" class="form-label">Efficiency Ratio:</label>
                <input type="text" class="form-control" id="attribute3"
name="attribute3" required>
            </div>
            <div class="mb-3">
                <label for="attribute4" class="form-label">Current Ratio:</label>
                <input type="text" class="form-control" id="attribute4"
name="attribute4" required>
            </div>
            <div class="mb-3">
                <label for="attribute5" class="form-label">Cash Conversion
Cycle:</label>
                <input type="text" class="form-control" id="attribute5"
name="attribute5" required>
            </div>
            <div class="mb-3">
                <label for="attribute6" class="form-label">Retention Ratio:</label>
                <input type="text" class="form-control" id="attribute6"
name="attribute6" required>
            </div>
```

```
            <div class="mb-3">
                <label for="attribute7" class="form-label">EBIT Margin or Return on
Assets (ROA):</label>
                <input type="text" class="form-control" id="attribute7"
name="attribute7" required>
            </div>
            <div class="mb-3">
                <label for="attribute8" class="form-label">Equity
Multiplier:</label>
                <input type="text" class="form-control" id="attribute8"
name="attribute8" required>
            </div>
            <div class="mb-3">
                <label for="attribute9" class="form-label">Asset Turnover
Ratio:</label>
                <input type="text" class="form-control" id="attribute9"
name="attribute9" required>
            </div>
            <div class="mb-3">
                <label for="attribute10" class="form-label">Equity Ratio:</label>
                <input type="text" class="form-control" id="attribute10"
name="attribute10" required>
            </div>
            <hr>

            <div class="mb-3">
                <button type="button" class="btn btn-success btn-lg"
onclick="predict()">Predict</button>
            </div>
            <hr>
        </form>
```

```
        <div id="result"></div>

        <script>
            function predict() {
                // Get input values for all 10 attributes
                var attribute1 = document.getElementById('attribute1').value;
                var attribute3 = document.getElementById('attribute3').value;
                var attribute4 = document.getElementById('attribute4').value;
                var attribute5 = document.getElementById('attribute5').value;
                var attribute6 = document.getElementById('attribute6').value;
                var attribute7 = document.getElementById('attribute7').value;
                var attribute8 = document.getElementById('attribute8').value;
                var attribute9 = document.getElementById('attribute9').value;
                var attribute10 = document.getElementById('attribute10').value;
                var selectedModel = document.getElementById('selectedModel').value;

                // Create a JSON object with attribute values
                var data = {
                    "selectedModel": selectedModel,
                    "attributeValues": [parseFloat(attribute1),
parseFloat(attribute3), parseFloat(attribute4),
                    parseFloat(attribute5), parseFloat(attribute6),
parseFloat(attribute7), parseFloat(attribute8),
                    parseFloat(attribute9), parseFloat(attribute10)],
                };

                // Make a POST request to the Flask server
                fetch('/predict', {
```

```
                    method: 'POST',
                    headers: {
                        'Content-Type': 'application/json',
                    },
                    body: JSON.stringify(data),
                })
                    .then(response => response.json())
                    .then(data => {
                        if (data && data.prediction !== undefined) {
                            // Display the prediction result
                            var predictionLabel = data.predictionLabel === "will" ?
"will be" : "will not be";
                            document.getElementById('result').innerHTML = `The
${data.usedModel} model predicts that the company ${data.predictionLabel} bankrupted
in the given forecasting period`;
                        } else {
                            // Display an error message if prediction is undefined
                            document.getElementById('result').innerHTML = "Error:
Unable to retrieve prediction.";
                        }
                    })
                    .catch((error) => {
                        console.error('Error:', error);
                        // Display an error message if there is an error in the
fetch request
                        document.getElementById('result').innerHTML = "Error: Unable
to make prediction. Please try again.";
                    });
            }
        </script>
```

```
    </section>
    <script src="https://code.jquery.com/jquery-3.6.0.min.js"

integrity="sha384-Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm"
        crossorigin="anonymous"></script>
    <script
src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.11.8/dist/umd/popper.min.js"

integrity="sha384-I7E8VVD/ismYTF4hNIPjVp/Zjvgyol6VFvRkX/vR+Vc4jQkC+hVqc2pM8ODewa9r"
        crossorigin="anonymous"></script>
    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/js/bootstrap.min.js"

integrity="sha384-BBtl+eGJRgqQAUMxJ7pMwbEyER4l1g+O15P+16Ep7Q9Q+zqX6gSbd85u4mG4QzX+"
        crossorigin="anonymous"></script>
</body>


</html>
```

style.css

```
body{
    background-color: #8ecae6;
    padding-top: 70px;
}

#mainNavbar{
    font-size: 3rem;
    font-weight: 400;
}
```

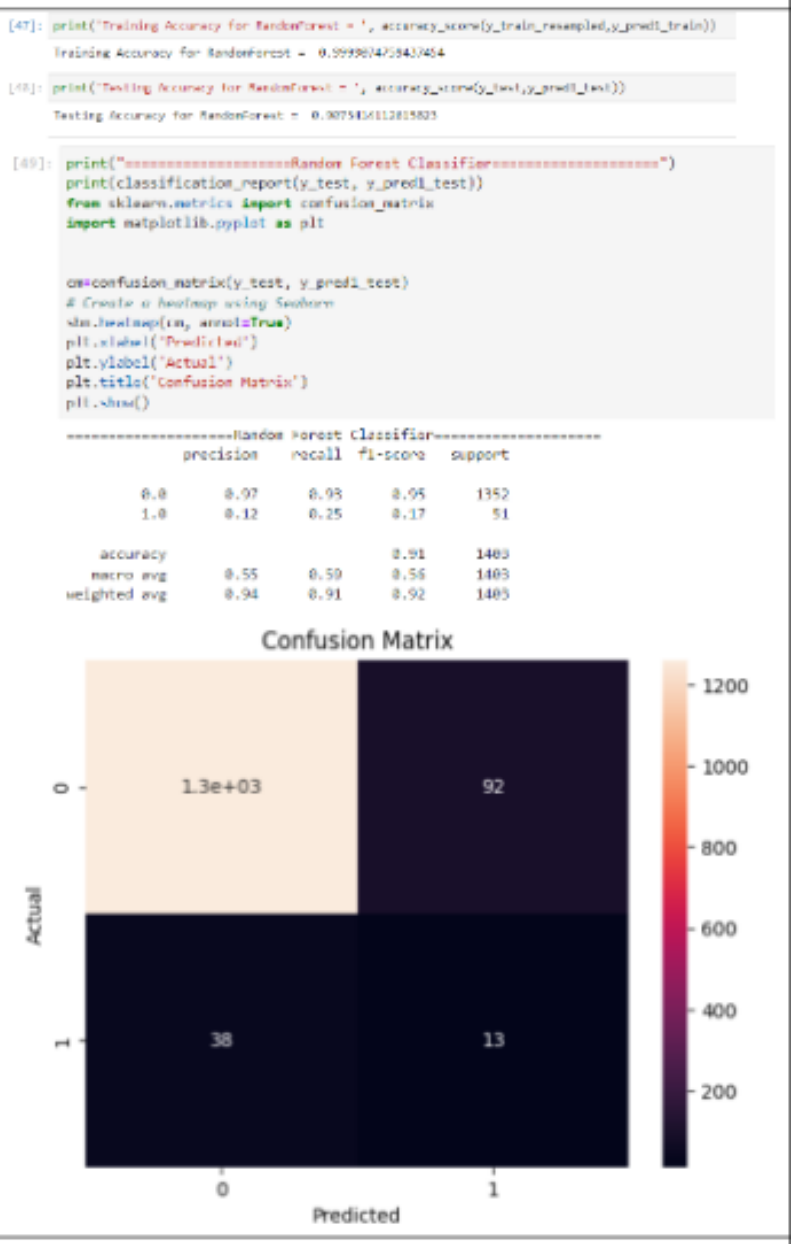
```css
#mainNavbar .navbar-brand{
    color: azure;
    font-size: 2rem;
}

#mainNavbar .navbar-brand:hover{
    color: #ffb703
}

.intro{
    color: #023047
}

.desc{
    color: #219ebc;
}

.background-image {
    background-image:
url('https://images.unsplash.com/photo-1526304640581-d334cdbbf45e?q=80&w=1000&auto=format&fit=crop&ixlib=rb-4.0.3&ixid=M3wxMjA3fDB8MHxleHBsb3JlLWZlZWR8N3x8fGVufDB8fHx8fA%3D%3D');
    background-size: cover;
    background-repeat: repeat-x;
    height: 100%; /* Adjust the height as needed */
}

label {
    display: block;
```

```
    margin-bottom: 5px;
}

input {
    margin-bottom: 10px;
    padding: 8px;
    width: 200px;
}

button {
    padding: 10px;
    background-color: #4CAF50;
    color: white;
    border: none;
    border-radius: 5px;
    cursor: pointer;
}

#result {
    margin-top: 30px;
    font-size: 2rem;
    text-align: center;
    color: #023047;
    margin-bottom: 50px;
    font-weight: 700;
}

#predictionForm{
    text-align: center;
```

```
    margin: 20px;
}


.form-label{
    font-weight: 700;
    color: #023047;
}
```

8. PERFORMANCE TESTING

1.Random Forest Classifier Performance

S.No.	Parameter	Values	Screenshot
1.	Metrics	**Classification Model:** **1) Random Forest classifier** • Accuracy score • Classification report • Confusion matrix	

```
[47]: print('Training Accuracy for RandomForest = ', accuracy_score(y_train_resampled,y_predt_train))
      Training Accuracy for RandomForest =  0.9998747583437454

[48]: print('Testing Accuracy for RandomForest = ', accuracy_score(y_test,y_predt_test))
      Testing Accuracy for RandomForest =  0.9075156112815823

[49]: print("====================Random Forest Classifier====================")
      print(classification_report(y_test, y_predt_test))
      from sklearn.metrics import confusion_matrix
      import matplotlib.pyplot as plt


      cm=confusion_matrix(y_test, y_predt_test)
      # Create a heatmap using Seaborn
      sns.heatmap(cm, annot=True)
      plt.xlabel('Predicted')
      plt.ylabel('Actual')
      plt.title('Confusion Matrix')
      plt.show()
```

```
====================Random Forest Classifier====================
              precision    recall  f1-score   support

         0.0       0.97      0.93      0.95      1352
         1.0       0.12      0.25      0.17        51

    accuracy                           0.91      1403
   macro avg       0.55      0.59      0.56      1403
weighted avg       0.94      0.91      0.92      1403
```

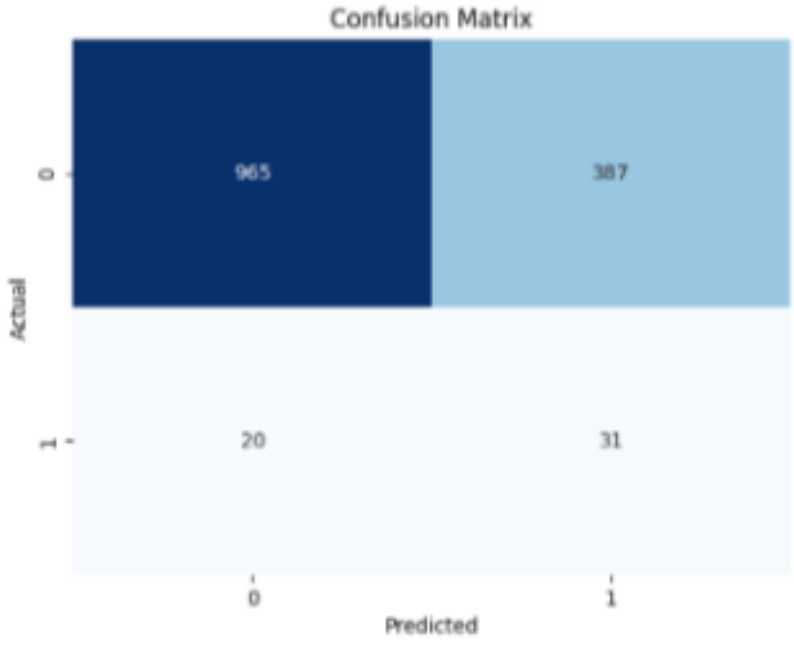
2. SVM Vector Machine Performance

- Accuracy score
- Classification report
- Confusion matrix

```
[61]: print('Training Accuracy for SVM = ', accuracy_score(y_train_resampled,y_pred4_train))
Training Accuracy for SVM =  0.8035714285714286

[62]: print('Testing Accuracy for SVM = ', accuracy_score(y_test,y_pred4_test))
Testing Accuracy for SVM =  0.7099073414112615
```

```
[63]: print("====================SVM====================")
print(classification_report(y_test, y_pred4_test))
from sklearn.metrics import confusion_matrix
import matplotlib.pyplot as plt


cm = confusion_matrix(y_test, y_pred4_test)

# Create a heatmap using Seaborn
sbn.heatmap(cm, annot=True, fmt="d", cmap="Blues", cbar=False)
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()
```

```
====================SVM====================
              precision    recall  f1-score   support

         0.0       0.98      0.71      0.83      1352
         1.0       0.07      0.61      0.13        51

    accuracy                           0.71      1403
   macro avg       0.53      0.66      0.48      1403
weighted avg       0.95      0.71      0.80      1403
```



3. K-Nearest Neighbours (KNN) Performance

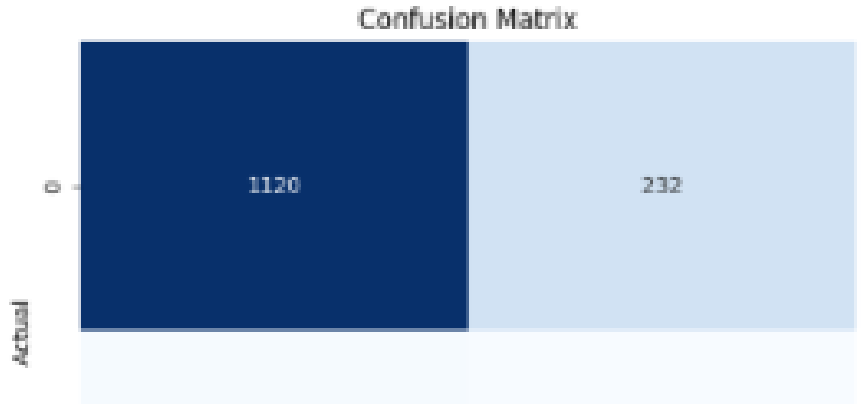
3) K-Nearest Neighbors (KNN)

- Accuracy score
- Classification report
- Confusion matrix

```
[51]: print('Training Accuracy for KNN = ', accuracy_score(y_train_resampled,y_pred2_train))

Training Accuracy for KNN =  0.9194893672854795

[52]: print('Testing Accuracy for KNN = ', accuracy_score(y_test,y_pred2_test))

Testing Accuracy for KNN =  0.8125445527598432
```

```
[53]: print("====================KNN====================")
print(classification_report(y_test, y_pred2_test))
from sklearn.metrics import confusion_matrix
import matplotlib.pyplot as plt


cm = confusion_matrix(y_test, y_pred2_test)

# Create a heatmap using Seaborn
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", cbar=False)
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()
```

```
--------------------KNN--------------------
              precision    recall  f1-score   support

         0.0       0.97      0.82      0.89      1352
         1.0       0.08      0.39      0.13        51

    accuracy                           0.81      1403
   macro avg       0.53      0.61      0.51      1403
weighted avg       0.94      0.81      0.87      1403
```



4. Logistic Regression Performance

<table>
<tr>
<td>

4) **Logistic Regression**
- Accuracy score
- Classification report
- Confusion matrix

</td>
<td>

</td>
</tr>
</table>

5.Decision Tree Classifier Performance

5) Decision tree classifier

- Accuracy score
- Classification report
- Confusion matrix

```python
[66]: print('Training Accuracy for DecisionTree : ', accuracy_score(y_train_resampled,y_pred5_train))
```

Training Accuracy for DecisionTree : [illegible]

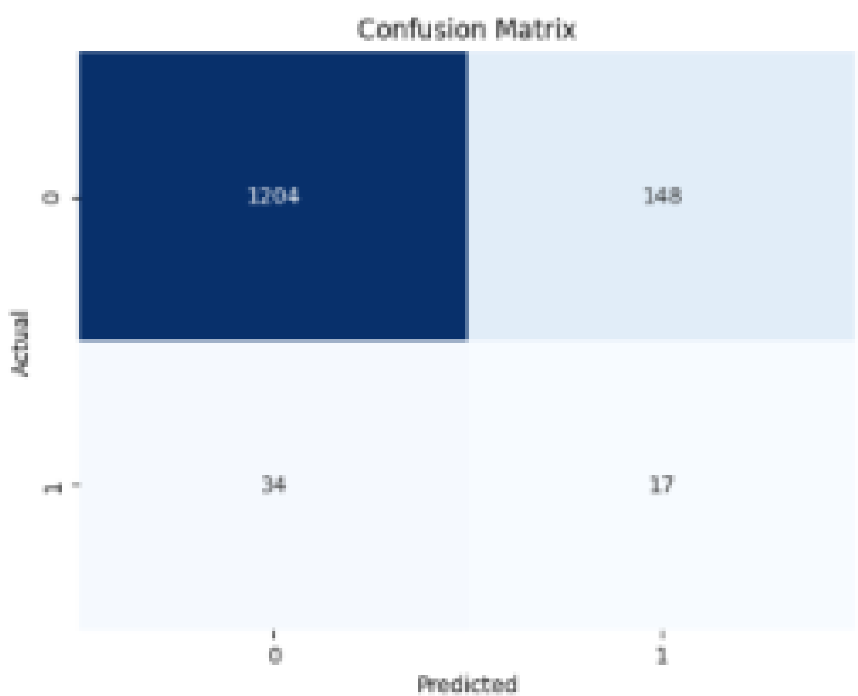
```python
[67]: print('Testing Accuracy for DecisionTree : ', accuracy_score(y_test,y_pred5_test))
```

Testing Accuracy for DecisionTree : [illegible]

```python
[68]: print("======================Decision Trees======================")
print(classification_report(y_test, y_pred5_test))
from sklearn.metrics import confusion_matrix
import matplotlib.pyplot as plt


cm = confusion_matrix(y_test, y_pred5_test)

# Create a heatmap using Seaborn
sbn.heatmap(cm, annot=True, fmt="d", cmap="Blues", cbar=False)
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()
```

======================Decision Trees======================

	precision	recall	f1-score	support
0.0	0.97	0.89	0.93	1352
1.0	0.10	0.33	0.16	51
accuracy			0.87	1403
macro avg	0.54	0.61	0.54	1403
weighted avg	0.94	0.87	0.90	1403

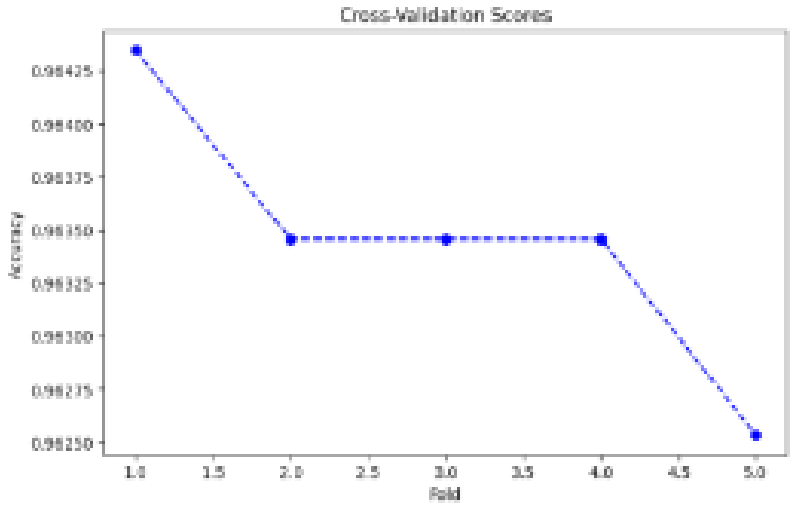


6.Model Tuning

Tune the Model	• Hyperparameter Tuning for random forest classifier • Validation Method	

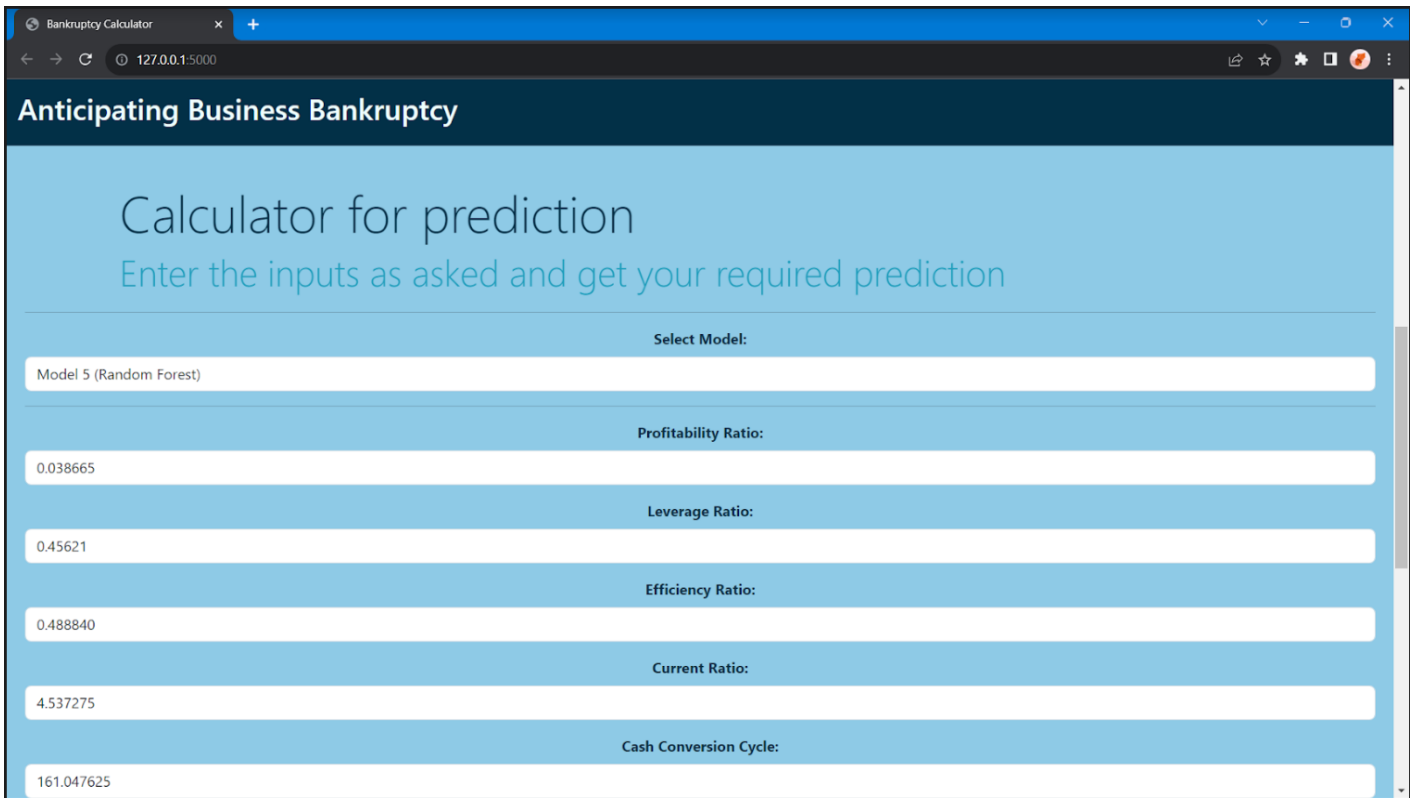
```
from sklearn.model_selection import train_test_split, GridSearchCV, cross_val_score
param_grid = {
    'n_estimators': [50, 100, 200],
    'max_depth': [None, 10, 20, 30],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}

# Create GridSearchCV
grid_search = GridSearchCV(rfc, param_grid, cv=5, scoring='accuracy', n_jobs=-1)

# Fit the model
grid_search.fit(X_train, y_train)

# Print the best parameters
print("Best Parameters:", grid_search.best_params_)

# Cross-validation scores
cv_scores = cross_val_score(grid_search.best_estimator_, X_train, y_train, cv=5)

# Print cross-validation scores
print("Cross-validation Scores:", cv_scores)

# Plotting cross-validation scores
plt.figure(figsize=(8, 5))
plt.plot(range(1, 6), cv_scores, marker='o', linestyle='--', color='b')
plt.title('Cross-Validation Scores')
plt.xlabel('Fold')
plt.ylabel('Accuracy')
plt.show()
```

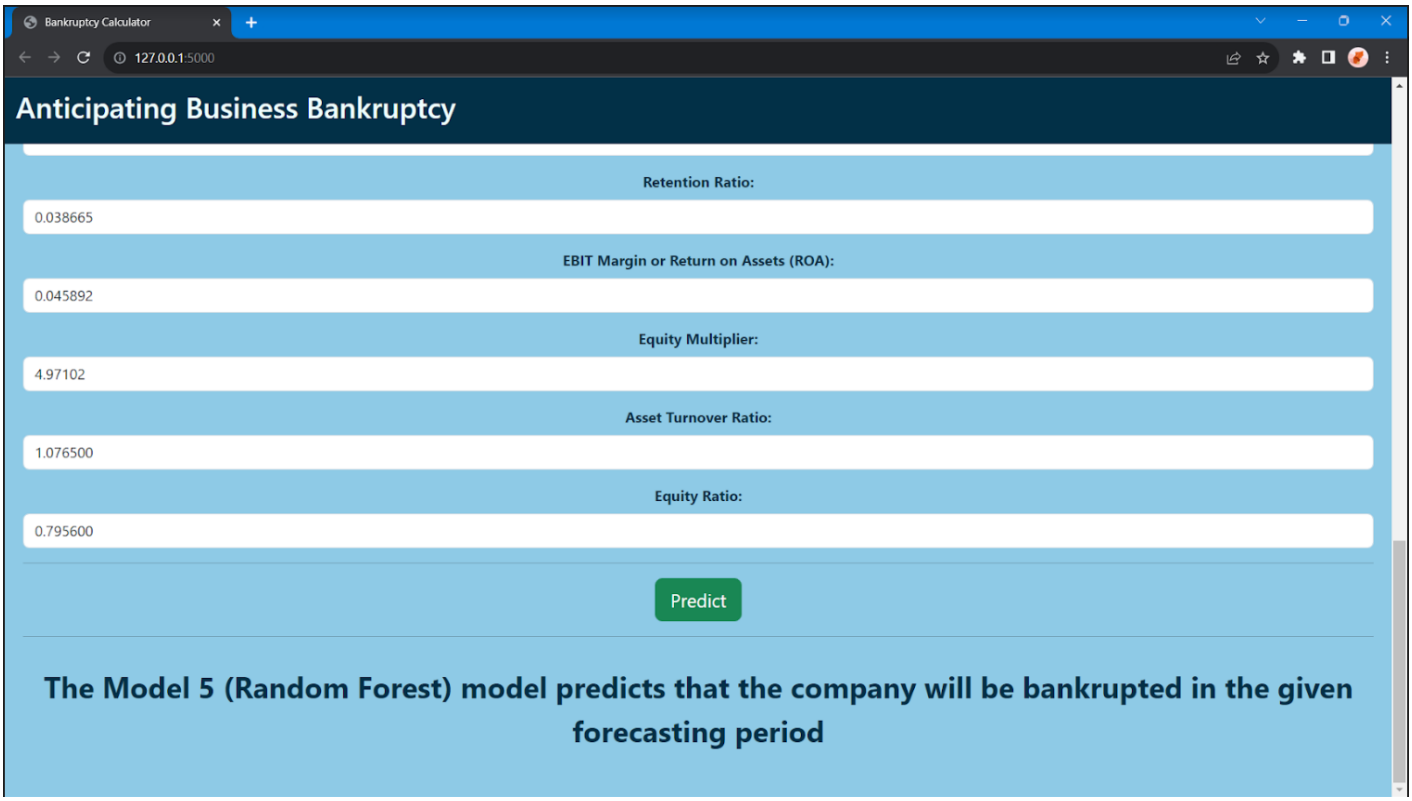
```
Best Parameters: {'max_depth': 30, 'min_samples_leaf': 2, 'min_samples_split': 2, 'n_estimators': 50}
Cross-validation Scores: [0.96434858 0.96345831 0.96345831 0.96345831 0.96353346]
```



9. RESULTS

The user interface to gather user inputs for a prediction






```
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with watchdog (windowsapi)
 * Debugger is active!
 * Debugger PIN: 135-690-792
127.0.0.1 - - [23/Nov/2023 12:05:18] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [23/Nov/2023 12:05:18] "GET /static/style.css HTTP/1.1" 200 -
127.0.0.1 - - [23/Nov/2023 12:05:22] "GET /favicon.ico HTTP/1.1" 404 -
127.0.0.1 - - [23/Nov/2023 12:06:03] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [23/Nov/2023 12:06:03] "GET /static/style.css HTTP/1.1" 304 -
C:\Users\hrish\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\base.py:439: UserWarning: X does not have valid feature names, but MinMax
Scaler was fitted with feature names
  warnings.warn(
 * Detected change in 'C:\\Users\\hrish\\AppData\\Local\\Programs\\Python\\Python310\\Lib\\site-packages\\sklearn\\base.py', reloading
[2023-11-23 12:07:47,620] INFO in app: Received prediction request for model: model5
[2023-11-23 12:07:47,630] INFO in app: Prediction using Model 5 (Random Forest): [1.]
127.0.0.1 - - [23/Nov/2023 12:07:47] "POST /predict HTTP/1.1" 200 -
 * Restarting with watchdog (windowsapi)
 * Debugger is active!
 * Debugger PIN: 135-690-792
```

10. ADVANTAGES AND DISADVANTAGES

ADVANTAGES

1. Forecasting model: The aim of the project is to develop a forecasting model that combines various financial indicators to predict the financial position of the company. This can provide valuable information about the company and long-term market performance.

2. Risk assessment: By studying financial indicators and other factors indicating the risk of bankruptcy, the project helps stakeholders to understand the causes of financial risks. This can help make informed decisions and implement strategies to reduce financial risks.

3. Making decisions: Insights gained through projects, analysis and forecasting can help stakeholders such as investors and lenders make decisions about their investment in a particular company. This allows them to assess the financial condition and future prospects of the company.

DISADVANTAGES

1. Data limitations: The accuracy and reliability of bankruptcy prediction models are highly dependent on the availability and quality of data. If the datasets provided have limitations, such as incomplete or inconsistent data, this may affect the accuracy of the predictive model.

2. Model accuracy: Although the project aims to improve the accuracy of bankruptcy prediction models, there is always the possibility of errors or misclassifications. Forecasting models are based on historical data and assumptions and may not take into account all the complex and dynamic changes in the business environment.

3. External factors: the project focuses on analyzing financial indicators to predict bankruptcy. However, external factors such as changes in economic policy, market conditions or unexpected events (such as natural disasters) can significantly affect the health of business and the economy. These external factors may not be fully accounted for in the forecasting model.

11. CONCLUSION

This project focused on predicting corporate bankruptcy using user-provided datasets. Through the analysis of various financial indicators and related ratios, we aim to develop a forecasting model that would allow us to assess the financial health of the company and offer long-term perspectives. Through our analysis, we identify the key factors and patterns that contribute to financial crises and bankruptcy. The developed predictive models can help stakeholders make decisions and implement strategies to mitigate financial risks. However, it is important to recognize that bankruptcy forecasting is a complex and evolving field and that more research is needed to make forecasting models more accurate and reliable.

12. FUTURE SCOPE

Although this project has achieved significant progress in the prediction of bankruptcy in companies, there are many avenues for future research. Some areas that could be explored in the future are:

> **Improving forecasting models:** Continue to refine and improve forecasting models by including additional variables, exploring different statistical methods, and considering the impact of macroeconomic factors.

> **Benchmarking:** Compare bankruptcy forecasting patterns in different countries or industries to identify similarities, differences, and trends in each scenario.

> **Combining Qualitative Factors:** Exploring the integration of qualitative factors such as management quality, industry dynamics, and corporate governance practices to increase the accuracy of bankruptcy prediction models.

13. APPENDIX

> Project Github Link and Source Code:
https://github.com/smartinternz02/SI-GuidedProject-603802-1697647310

> Project Demo Link:

https://youtu.be/OMnMjSiHggg