

1.INTRODUCTION

Anticipating business bankruptcy is a critical aspect of financial management and risk assessment for both small startups and established corporations. Understanding the signs and signals of impending financial distress is essential for business owners, investors, creditors, and stakeholders to take proactive measures and make informed decisions. This introduction will provide an overview of the importance of anticipating business bankruptcy, the key factors to consider, and some common warning signs to watch out for.

1.1 OVERVIEW

Anticipating business bankruptcy is a crucial aspect of financial management and risk assessment for companies. It involves the proactive identification and analysis of warning signs and risk factors that may lead to financial distress and potential insolvency. By understanding the factors contributing to a company's financial health and monitoring key indicators, businesses, investors, creditors, and stakeholders can take timely and informed actions to mitigate the risk of bankruptcy. Here's an overview of the key aspects of anticipating business bankruptcy:

1.Importance of Anticipation: Anticipating bankruptcy is essential for several reasons. It allows businesses to take preventive measures to address financial challenges, protect stakeholders' interests, maintain operational continuity, and ensure compliance with legal and regulatory requirements. Timely detection of financial distress can help avoid the disruptive consequences of bankruptcy.

2.Key Factors to Consider:

- **Financial Statements:** Careful analysis of financial statements, including income statements, balance sheets, and cash flow statements, is fundamental to assessing a company's financial health.
- **Market and Industry Trends:** Understanding the broader economic environment and industry-specific trends is crucial for evaluating a company's potential challenges and vulnerabilities.
- **Debt and Liquidity:** High levels of debt, poor liquidity, and an overreliance on short-term financing can be early indicators of financial distress.
- **Operational Efficiency:** Inefficient operations, declining profitability, and cash flow issues can signal bankruptcy risk.
- **Management and Governance:** The competency and integrity of a company's management team play a significant role in its financial stability.

3.Common Warning Signs:

- **Consistent Losses:** Prolonged periods of financial losses or diminishing profitability are

red flags.

- **Cash Flow Issues:** Persistent negative cash flows, inability to meet financial obligations, or reliance on short-term loans can indicate financial trouble.
- **Increasing Debt Burden:** Rapidly accumulating debt or a high debt-to-equity ratio may suggest financial distress.
- **Declining Market Share:** A decrease in market share can signify operational and financial challenges.
- **Legal Issues:** Ongoing legal disputes, regulatory violations, or pending lawsuits can adversely affect a company's financial stability.
- **Credit Rating Downgrades:** A declining credit rating can lead to higher borrowing costs and restricted access to capital.

4.Mitigation and Response: Once potential bankruptcy risks are identified, businesses can take various actions to mitigate these risks, such as implementing cost-cutting measures, refinancing or restructuring debt, seeking additional capital, or considering mergers and acquisitions. Seeking professional financial advice is often a wise step in this process.

5.Continuous Monitoring: Anticipating bankruptcy is an ongoing process that requires regular monitoring of financial indicators and reassessment of risk factors. By staying vigilant and proactive, companies can adapt to changing circumstances and make informed decisions to safeguard their financial stability.

1.2 PURPOSE

The purpose of anticipating business bankruptcy is multifaceted and serves the interests of various stakeholders, including business owners, investors, creditors, and the broader economy. Here are the primary purposes of anticipating business bankruptcy:

- Risk Mitigation
- Stakeholder Protection
- Operational Continuity
- Regulatory Compliance
- Optimal Use of Resources
- Preservation of Economic Value
- Access to Financing
- Creditor Relations
- Strategic Decision-Making
- Customer and Supplier Confidence

2. LITERATURE SURVEY

2.1 EXISTING PROBLEM

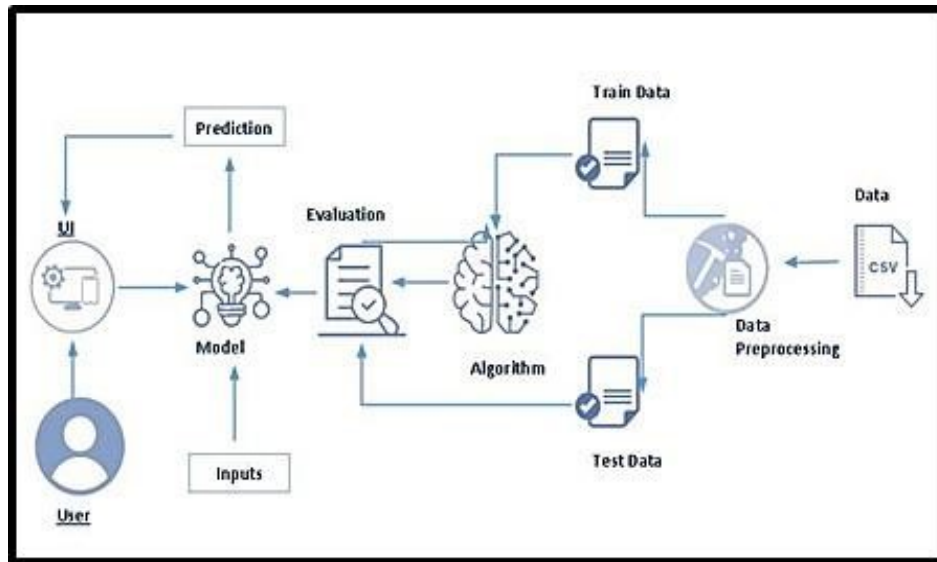
Existing problems in anticipating business bankruptcy include complexities in financial data analysis, creative accounting practices that obscure true financial health, time lags in reporting, external economic factors influencing risk, information asymmetry between stakeholders, hidden liabilities not disclosed in financial statements, and overreliance on historical data for predictions. Additionally, legal and ethical considerations may hinder early detection, while a lack of universal metrics and behavioral factors further complicate the process. These issues make it challenging to accurately predict bankruptcy and underline the need for advanced analytical tools, transparency, and industry-specific insights for more effective anticipation

2.2 PURPOSED SOLUTION

Proposed solutions for anticipating business bankruptcy include leveraging advanced data analytics and AI for early detection, developing real-time financial monitoring systems, conducting comprehensive financial due diligence, employing expert advisory services, and assessing management behavior. Scenario planning, regular financial audits, credit risk assessments, and stress testing can improve preparedness. Diversifying supplier and customer relationships, ensuring access to diverse funding sources, and maintaining legal and regulatory compliance are crucial. Strengthening customer and supplier relationships, practicing strategic financial management, and fostering transparency through continuous monitoring are essential to enhance bankruptcy risk anticipation and mitigation.

3. THEORETICAL ANALYSIS

3.1 BLOCK DIAGRAM



3.2 HARDWARE / SOFTWARE DESIGNING

To complete this project, you must require following software's, concepts and packages

1. Visual studio
2. Python packages:
 - a. Open Jupyter in vs code then install packages
 - b. Type "pip install numpy" and click enter
 - c. Type "pip install pandas" and click enter.
 - d. Type "pip install seaborn" and click enter.
 - e. Type "pip install matplotlib" and click enter.
 - f. Type "pip install pickle" and click enter.
 - g. Type "pip install Flask" and click enter

4.EXPERIMENTAL INVESTIGATION

1. Know fundamental concepts and techniques used for machine learning.
2. Gain a broad understanding about data.
3. Have knowledge on pre-processing the data/transformation techniques on outlier and some visualization concepts.

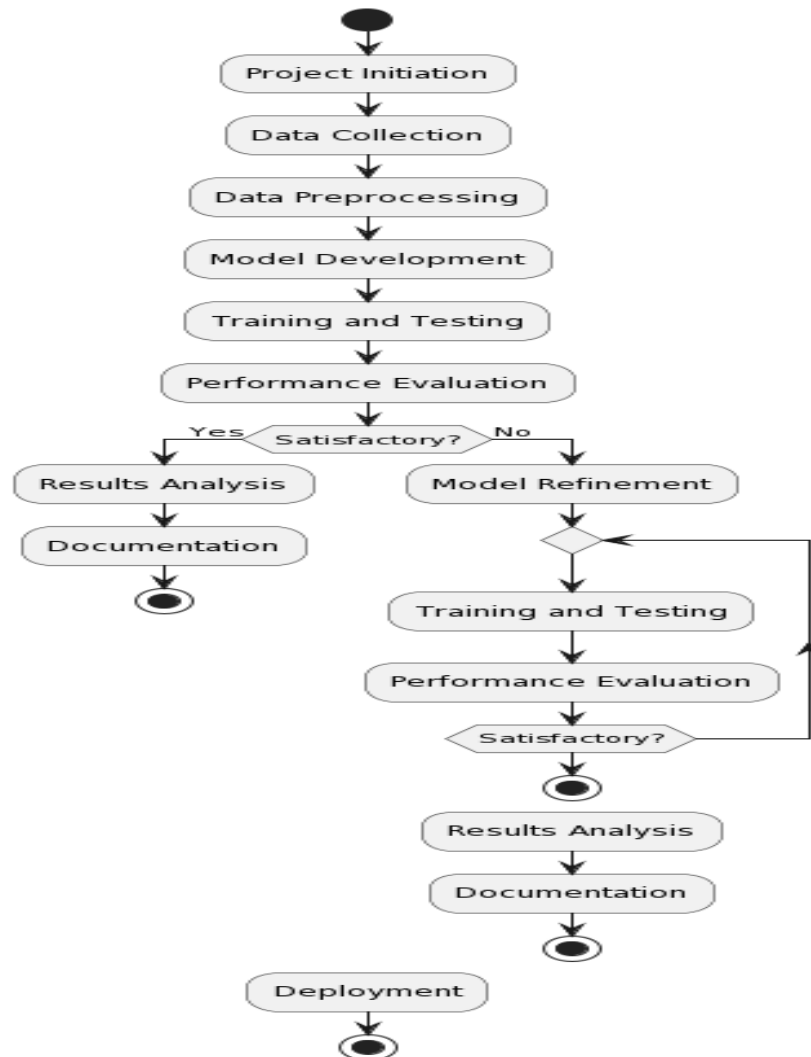
Project Flow:

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

To accomplish this, we have to complete all the activities listed below,

- Define Problem / Problem Understanding
 - Specify the business problem
 - Business requirements
 - Literature Survey
 - Social or Business Impact.
- Data Collection & Preparation
 - Collect the dataset
 - Data Preparation
- Exploratory Data Analysis
 - Descriptive statistical
 - Visual Analysis
- Model Building
 - Training the model in multiple algorithms
 - Testing the model
- Performance Testing
 - Testing model with multiple evaluation metrics
- Model Deployment
 - Save the best model
 - Integrate with Web Framework

5. FLOWCHART



6. RESULTS

Activity 1: Importing the libraries

Import the necessary libraries as shown in the image.

```
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt # plotting
import numpy as np # linear algebra
import os # accessing directory structure
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

#Import the libraries
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.metrics import f1_score, make_scorer, accuracy_score, recall_score, precision_score
from sklearn.metrics import confusion_matrix
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC

import random

import warnings
warnings.filterwarnings("ignore")
# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will list all files under the
#input directory
```

Activity 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 csv file.

```
data = pd.read_csv("C:/Users/Bobby/Desktop/Internship/Project 3/1year.csv")
```

data

	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
...
7007	0.038665	0.071884	0.48884	7.8004	221.01	0.038665	0.045892	11.068	1.0765	0.7956	1
7008	0.001091	0.8516	0.003463	1.0086	-44.467	0.086248	0.001091	0.17429	1.0297	0.14842	1
7009	-0.091442	0.7055	-0.047216	0.92568	-7.2952	0	-0.090374	0.41744	9.1345	0.2945	1
7010	0.13809	3.3357	-2.364	0.29128	-88.382	-3.3963	0.13809	-0.70021	9.9852	-2.3357	1
7011	0.098271	0.8333	0.000426	1.0005	-43.191	0	0.12838	0.20019	2.5144	0.16682	1

7012 rows × 11 columns

Activity 3: Data Preparation

Data preparation, also known as data preprocessing, is a crucial step in the data analysis and machine learning lifecycle. It involves cleaning, transforming, and organizing raw data into a format that is suitable for analysis or model training. Properly prepared data is essential for achieving accurate and meaningful insights or building robust machine learning models. As we have understood how the data is, let's pre-process the collected data.

The download data set 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.

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

Getting the Preliminary Information about the Dataset:

Let's find the shape of our dataset first. To find the shape of our data, the `df.shape` method is used. To find the data type, `df.info()` function is used.


```
print("Shape of Data:",data.shape)
r, c = data.shape
print("Number of Rows:",r)
print("Number of Columns:",c)
```

```
Shape of Data: (7012, 11)
Number of Rows: 7012
Number of Columns: 11
```

```
data.info()
```

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

Handling Missing Values:

Check and number of missing values and its percentage for the all the columns and filled 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.

```
data.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
```

```
data.isnull().sum().sum()
```

```
0
```

```
(data.eq('?')).any()
```

```
Attr1      True  
Attr2      True  
Attr3      True  
Attr4      True  
Attr5      True  
Attr6      True  
Attr7      True  
Attr8      True  
Attr9      True  
Attr10     True  
class      False  
dtype: bool
```

with “ ? ” in it. So, the meaning of the “ ? ” was same as Null or NA, so we identified and replaced all the ? with NAN As we checked above, we were not able to see any missing values in all the column but there were some columns.

```
data.replace('?',np.NaN,inplace=True)
```

```
data.head()
```

	A1_Score	A2_Score	A3_Score	A4_Score	A5_Score	A6_Score	A7_Score	A8_Score	A9_Score	A10_Score	...	gender	ethnicity	jundice	austim	contry_
0	1	1	1	1	0	0	1	1	0	0	...	f	White-European	no	no	'United
1	1	1	0	1	0	0	0	1	0	1	...	m	Latino	no	yes	
2	1	1	0	1	1	0	1	1	1	1	...	m	Latino	yes	yes	
3	1	1	0	1	0	0	1	1	0	1	...	f	White-European	no	yes	'United
4	1	0	0	0	0	0	0	1	0	0	...	f	NaN	no	no	

5 rows × 21 columns

```
data.replace('?',np.NaN,inplace=True)
```

```
data.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
```

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

```
data.update(data[['Attr1','Attr2','Attr3','Attr4','Attr5','Attr6','Attr7','Attr8','Attr9','Attr10']].  
            fillna(0))
```

```
data=data.fillna(data.mean())
```

Dropping Unwanted Columns:

In the dataset there were many columns which were not needed for the classification of the loan

types. All these columns were removed to make it easier for models to focus on the remaining columns.

- used_app_before
- ethnicity
- ontry_of_res
- age_desc
- relation

Activity 4: Descriptive Statistical:

Descriptive analysis is to study the basic features of data with the statistical process. Here pandas has 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.

```
: {column:len(data[column].unique()) for column in data.columns}
: {'Attr1': 6618,
  'Attr2': 6593,
  'Attr3': 6691,
  'Attr4': 6274,
  'Attr5': 6806,
  'Attr6': 4202,
  'Attr7': 6661,
  'Attr8': 6671,
  'Attr9': 5500,
  'Attr10': 6621,
  'class': 2}
```

```
print("Information about the Dataset")
data.info()
```

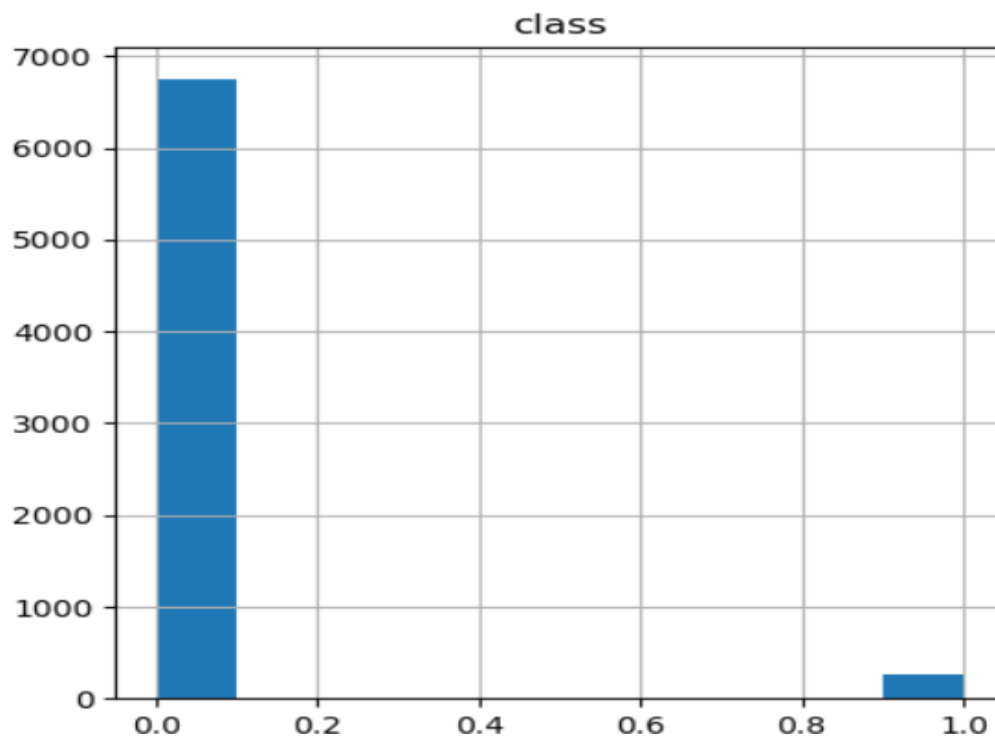
```
Information about the Dataset
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7012 entries, 0 to 7011
Data columns (total 11 columns):
 #   Column      Non-Null Count  Dtype  
---  --
 0   Attr1       7012 non-null   object  
 1   Attr2       7012 non-null   object  
 2   Attr3       7012 non-null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

Visual Analysis

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.

Class Analysis:

```
data.hist(figsize = (5,5))  
array([[<Axes: title={'center': 'class'}>]], dtype=object)
```



Activity 5: Model Building:

Dealing With Imbalanced Data

Feature Scaling¶

Scale all features in order to have zero mean and unit variance

```
: X_scaled = pd.DataFrame(StandardScaler(copy=False).fit_transform(X))
X_scaled.columns = X.columns
```

Target class Imbalance : SMOTE

very low % of the companies has Bankrupted in the dataset, making it imbalanced target class problem. Hence positive target class (Bankrupt=1) is under-represented. This could be challenging as lack of positive class in the train data may lead machine learning model to have poor performance in terms of detecting positive class in the unseen data. SMOTE (Synthetic Minority Oversampling Technique) proposed by Chawla et al 2002, is a well applied technique to handle such scenario. SMOTE actually creates as many synthetic examples for minority class as are required so that finally two target class are well represented. It does so by synthesising samples that are close to the feature space, for the minority target class. More about SMOTE

```
sm = SMOTE(random_state=123)
X_sm, y_sm = sm.fit_resample(X_scaled, y)

print(f'''Shape of X before SMOTE:{X_scaled.shape}
Shape of X after SMOTE:{X_sm.shape}''', "\n\n")

print(f'''Target Class distributuion before SMOTE:\n{y.value_counts(normalize=True)}
Target Class distributuion after SMOTE : \n{y_sm.value_counts(normalize=True)}''')
```

```
Shape of X before SMOTE:(7012, 10)
Shape of X after SMOTE:(13512, 10)
```

```
Target Class distributuion before SMOTE:
0    0.963491
1    0.036509
Name: class, dtype: float64
Target Class distributuion after SMOTE :
0    0.5
1    0.5
Name: class, dtype: float64
```

Train-Test Split

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"

```
X = data.drop(["class"],axis=1)
```

X

	Attr1	Attr2	Attr3	Attr4	Attr5	Attr6	Attr7	Attr8	Attr9	Attr10
0	0.20055	0.37951	0.39641	2.0472	32.351	0.38825	0.24976	1.3305	1.1389	0.50494
1	0.20912	0.49988	0.47225	1.9447	14.786	0	0.25834	0.99601	1.6996	0.49788
2	0.24866	0.69592	0.26713	1.5548	-1.1523	0	0.30906	0.43695	1.309	0.30408
3	0.081483	0.30734	0.45879	2.4928	51.952	0.14988	0.092704	1.8661	1.0571	0.57353
4	0.18732	0.61323	0.2296	1.4063	-7.3128	0.18732	0.18732	0.6307	1.1559	0.38677
...
7007	0.038665	0.071884	0.48884	7.8004	221.01	0.038665	0.045892	11.068	1.0765	0.7956
7008	0.001091	0.8516	0.003463	1.0086	-44.467	0.086248	0.001091	0.17429	1.0297	0.14842
7009	-0.091442	0.7055	-0.047216	0.92568	-7.2952	0	-0.090374	0.41744	9.1345	0.2945
7010	0.13809	3.3357	-2.364	0.29128	-88.382	-3.3963	0.13809	-0.70021	9.9852	-2.3357
7011	0.098271	0.8333	0.000426	1.0005	-43.191	0	0.12838	0.20019	2.5144	0.16682

7012 rows × 10 columns

```
y=data['class']
```

y

```
0      0
1      0
2      0
3      0
4      0
..
7007    1
7008    1
7009    1
7010    1
7011    1
Name: class, Length: 7012, dtype: int64
```



```
X_train , X_test , y_train ,y_test = train_test_split(X_sm,y_sm,test_size= 0.3)
```

```
print("Shape of X_train: ",X_train.shape)
print("Shape of y_train: ",y_train.shape)
print("Shape of X_test: ",X_test.shape)
print("Shape of y_test: ",y_test.shape)
```

```
Shape of X_train: (9458, 10)
Shape of y_train: (9458,)
Shape of X_test: (4054, 10)
Shape of y_test: (4054,)
```

Training and Testing The Model in 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.

Support Vector Classifier:

A function named “svc” is created and train and test data are passed as the parameters. Inside the function, “SVC” algorithm is initialised 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 is calculated.

SVC

```
: from sklearn.svm import SVC
  svm=SVC(kernel='rbf',random_state=0)
  svm.fit(X_train,y_train)
```

```
: SVC
  SVC(random_state=0)
```

```
: y_pred_svc=svm.predict(X_test)
```

```
: print('Training Set: ',svm.score(X_train,y_train))
  print('Testing Set: ',svm.score(X_test,y_test))
```

```
Training Set: 0.6478113766123916
Testing Set: 0.6428219042920572
```

```
: accuracy_SVC=svm.score(X_test,y_test)
  print('Accuracy_SVM: ', accuracy_SVC*100)
```

```
Accuracy_SVM: 64.28219042920573
```

Decision Tree Classifier

A function named “dt” is created and train and test data are passed as the parameters. Inside the function, DecisionTreeClassifier algorithm is initialised and training data is passed to the model with the .fit() function. Test data is predicted with .predict() function and saved in a new variable. For evaluating the model, overfitting and accuracy is calculated.

Decision Tree

```
: dt= DecisionTreeClassifier()

: dt.fit(X_train,y_train)
: ▾ DecisionTreeClassifier
  DecisionTreeClassifier()

: y_pred_dt=dt.predict(X_test)

: print('Training Set: ',dt.score(X_train,y_train))
: print('Test Set: ',dt.score(X_test,y_test))

Training Set:  0.9998942694015648
Test Set:  0.8897385298470646

: print("Accuracy:",metrics.accuracy_score(y_test, y_pred_dt)*100)

Accuracy: 88.97385298470645

: accuracy_dt=accuracy_score(y_test,y_pred_dt)
: print('Accuracy DT: ', accuracy_dt*100)

Accuracy DT:  88.97385298470645
```

Random Forest Classifier:

A function named “rand_forest” is created and train and test data are passed as the parameters. Inside the function, RandomForestClassifier algorithm is initialised 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 is calculated.

Random Forest

```
rand_forest = RandomForestClassifier(random_state=42)
```

```
# Fit the model to the data  
rand_forest.fit(X_train, y_train)
```

```
▼      RandomForestClassifier  
RandomForestClassifier(random_state=42)
```

```
predictionRF= rand_forest .predict(X_test)  
# check the accuracy on the training set  
print('Training set : ',rand_forest.score(X_train, y_train))  
print('Testing set  : ',rand_forest.score(X_test, y_test))
```

```
Training set : 0.9998942694015648  
Testing set  : 0.9501726689689196
```

```
accuracy_RF=rand_forest.score(X_test, y_test)  
print ("Accuracy_RF:",accuracy_RF*100)
```

```
Accuracy_RF: 95.01726689689195
```

Activity6. Performance Testing & Hyperparameter Tuning:

Comparing All The Models.

For comparing the above five models, the accuracy_df function is used.

Below is the accuracy comparison of all the models and we can clearly see that accuracy for Logistics Regression, Decision Tree and Random Forest is 95 percent so we can take any of this model for our classification purpose.

Comparison of Accuracies

```
: # Create a table to compare the accuracies of each model
accuracy_df = pd.DataFrame({
    'Model': ['SVM', 'DecisionTree', 'Randomforest'],
    'Accuracy': [accuracy_SVC*100, accuracy_dt*100, accuracy_RF*100]
})
print(accuracy_df)
```

	Model	Accuracy
0	SVM	64.282190
1	DecisionTree	88.973853
2	Randomforest	95.017267

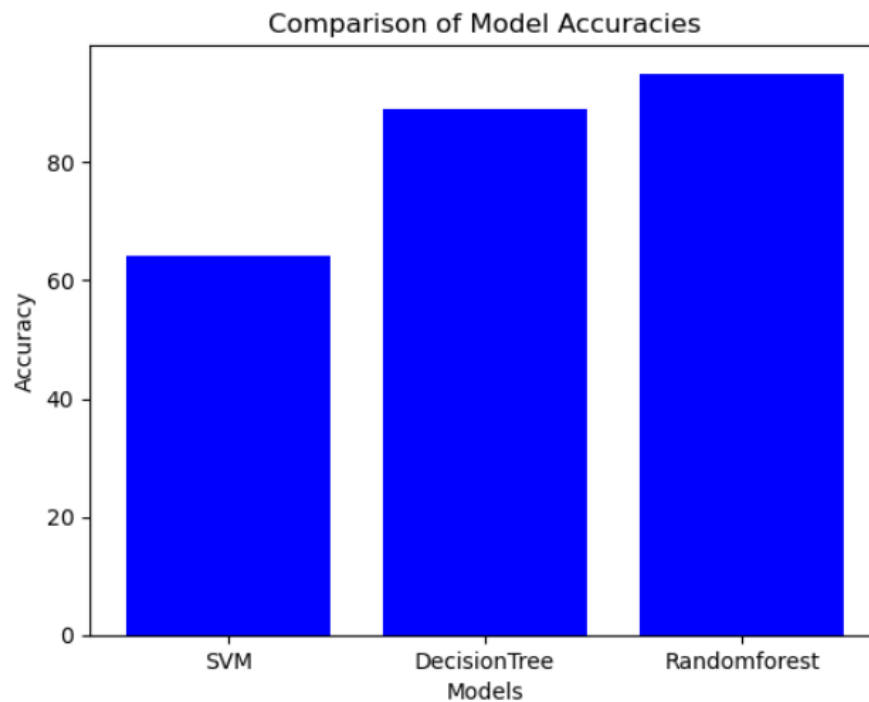
Graphical Representation of the Model Comparison

```
models = ['SVM', 'DecisionTree', 'Randomforest']

accuracies = [accuracy_SVC*100, accuracy_dt*100, accuracy_RF*100]
plt.bar(models, accuracies, color='blue')

# Add title and axis labels
plt.title('Comparison of Model Accuracies')
plt.xlabel('Models')
plt.ylabel('Accuracy')
```

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



Activity7. Model Deployment:

Save 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.

Exporting the model

```
import pickle
pickle.dump(rand_forest, open('C:/Users/Bobby/Desktop/Internship/Project 3/model.pkl', 'wb'))
```

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.

```
model= pickle.load(open("C:/Users/Bobby/Desktop/Internship/Project 3/model.pkl", 'rb'))
print(model.predict([[0.20055,0.37951,0.39641,2.0472,32.351,0.38825,0.24976,1.3305,1.1389,0.50494]]))

[0]
```

We can see above that our model has predicted "0", that means model has classified this as Bankruptcy Detection of Company.

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

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 user where he has to 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

Building Html Pages:

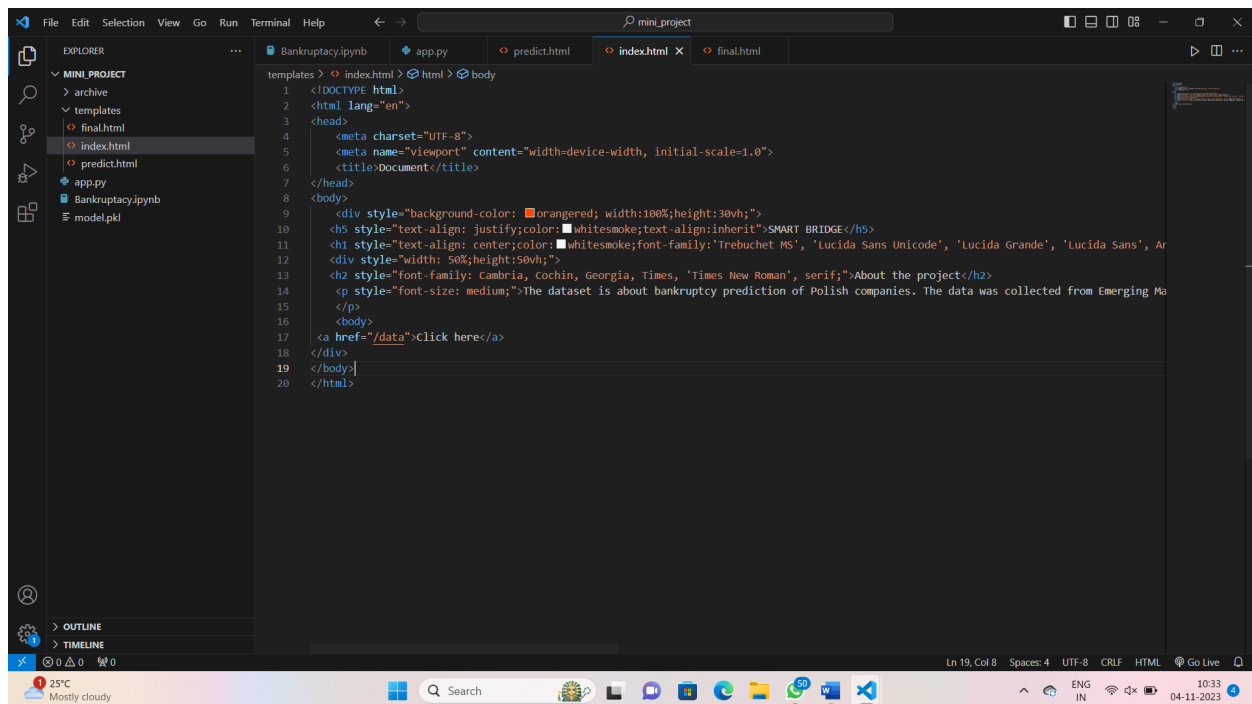
For this project create two HTML files namely and save them in the templates folder.

- Index.html
- Predict .html

1.index.html:

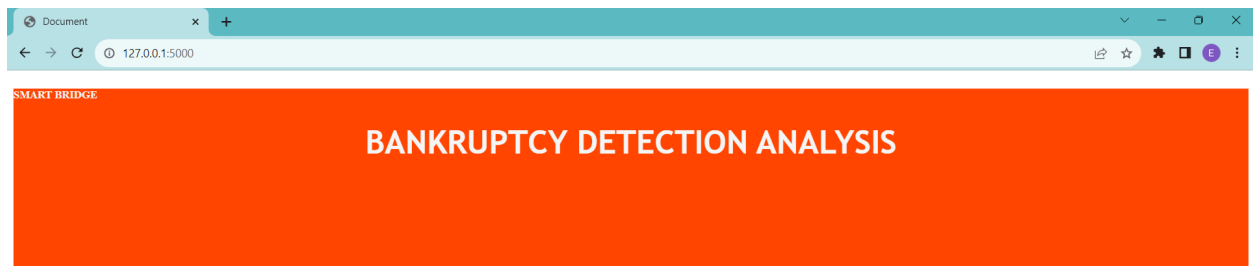
This page consists of basic information about user and it is home page that it is going to be displayed first

Input:



```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <title>Document</title>
7 </head>
8 <body>
9   <div style="background-color: #f08080; width:100%;height:30vh;">
10    <h5 style="text-align: justify;color:#f0f0f0;text-align:inherit">SMART BRIDGE</h5>
11    <h1 style="text-align: center;color:#f0f0f0;font-family:'Trebuchet MS', 'Lucida Sans Unicode', 'Lucida Grande', 'Lucida Sans', Arial, sans-serif">SMART BRIDGE</h1>
12    <div style="width: 50%;height:50vh;">
13      <h2 style="font-family: Cambria, Cochin, Georgia, Times, 'Times New Roman', serif;">About the project</h2>
14      <p style="font-size: medium;">The dataset is about bankruptcy prediction of Polish companies. The data was collected from Emerging Markets Data Analytics</p>
15    </div>
16  </div>
17  <a href="/data">Click here</a>
18 </div>
19 </body>
20 </html>
```

Output:



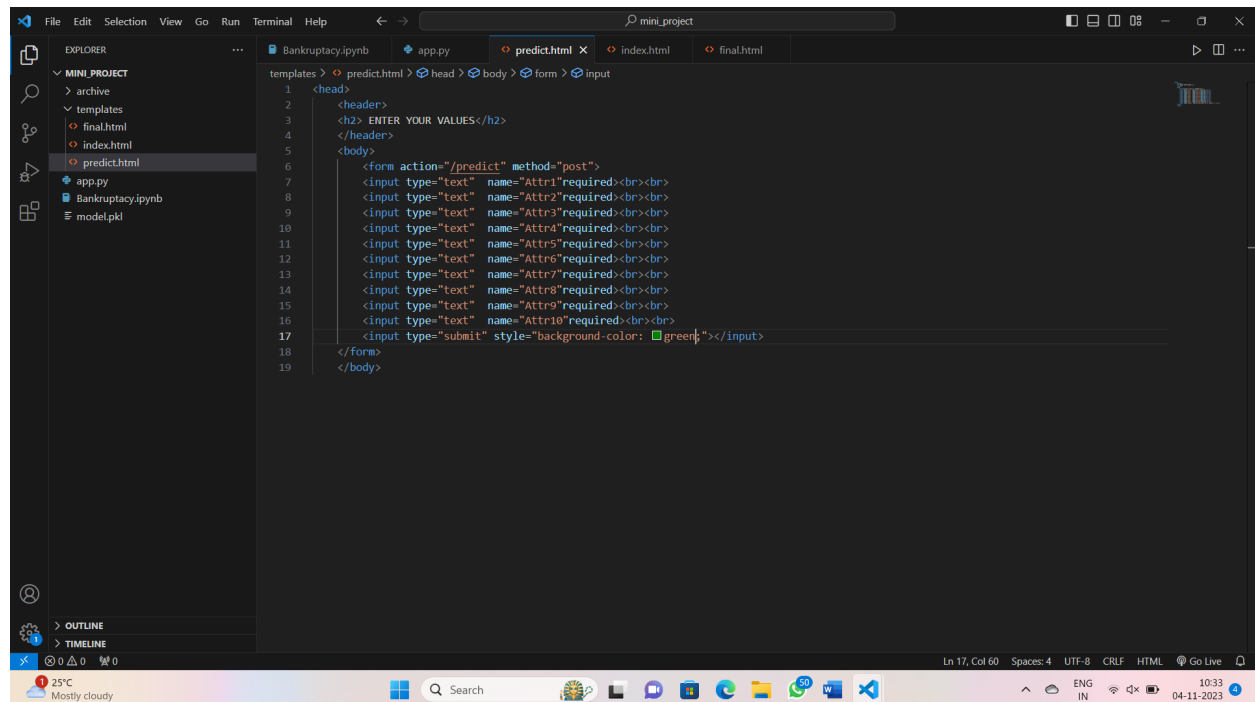
About the project

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 analysed in the period 2000-2012, while the still operating companies were evaluated for year 2007.

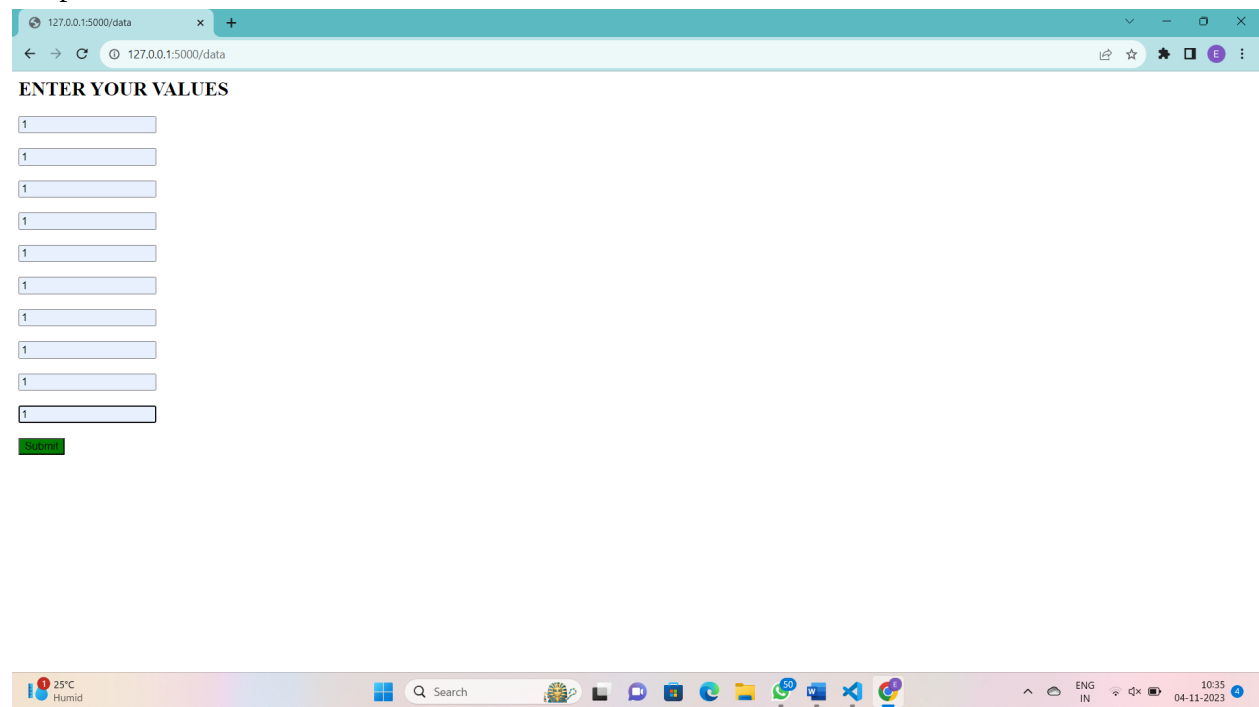
[Click here](#)

Predict.html:

Input

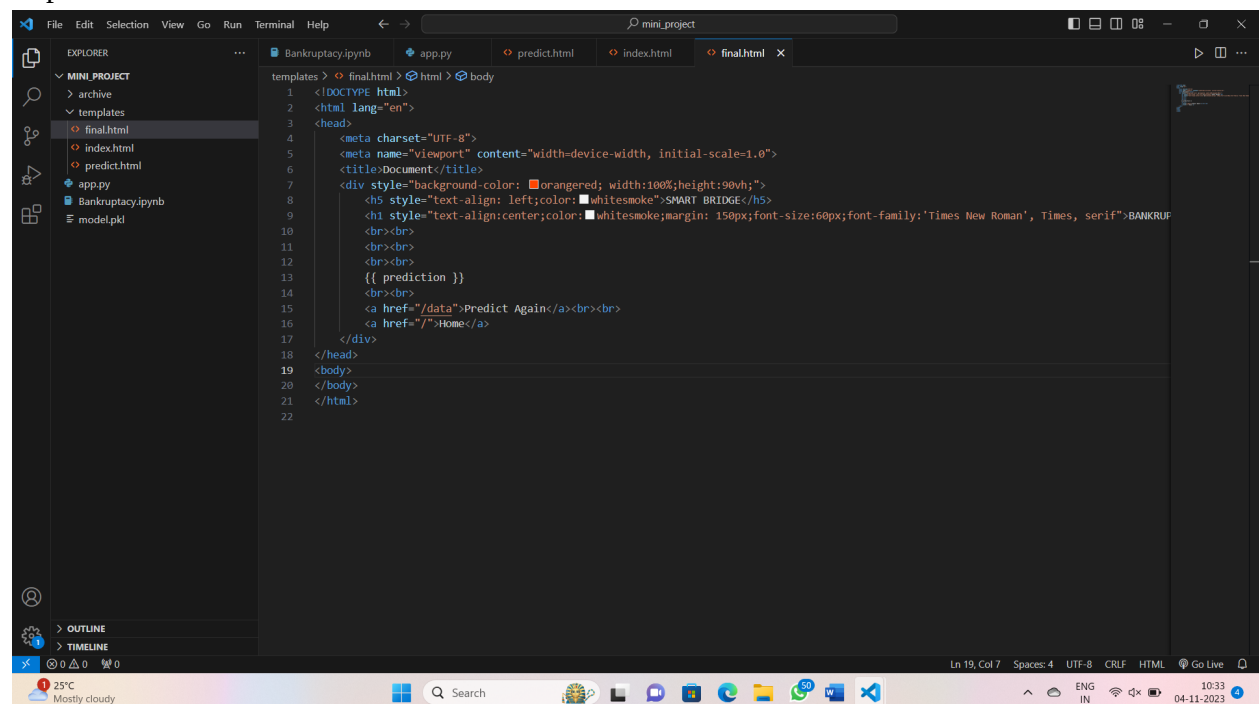


Output:

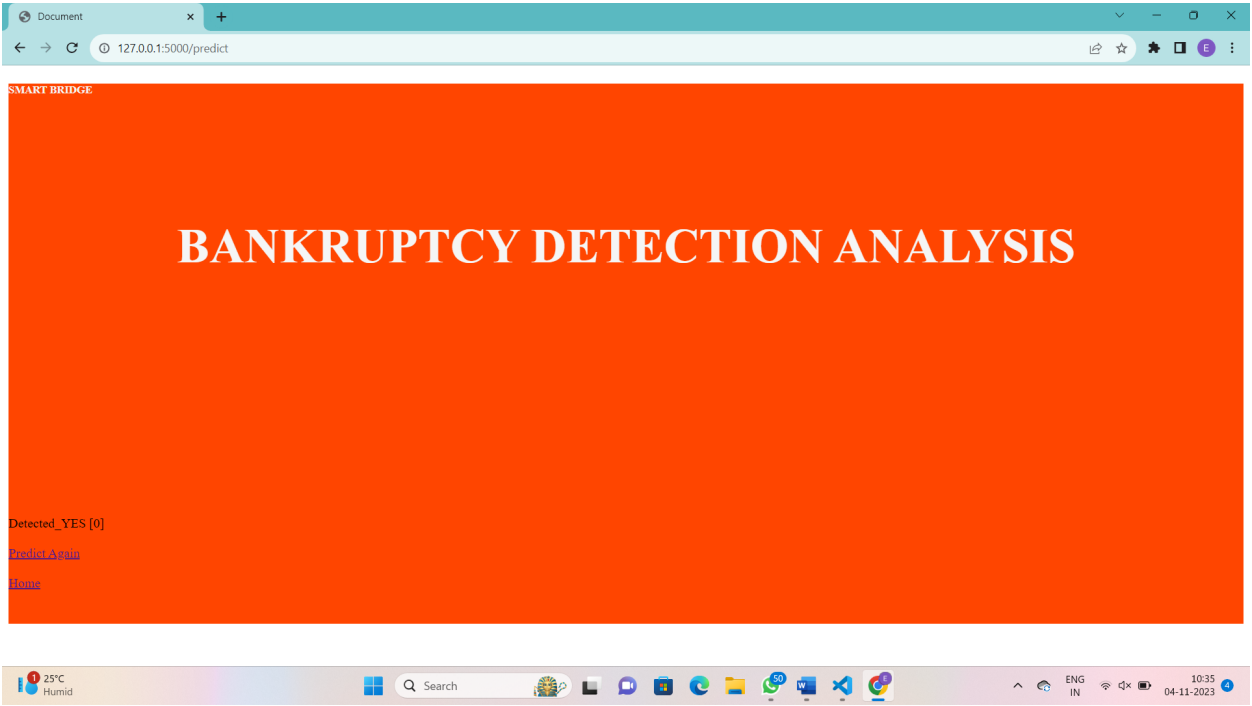


Final.html:

Input:



Output:



7. ADVANTAGES AND DISADVANTAGES

Advantages:

1. **Early Intervention:** Anticipating bankruptcy allows businesses to intervene at an early stage to address financial problems, potentially avoiding insolvency.
2. **Stakeholder Protection:** It safeguards the interests of stakeholders, such as shareholders, employees, suppliers, and creditors, by enabling them to take preventive actions to minimize losses.
3. **Operational Continuity:** Early detection of financial distress helps businesses maintain operational continuity and serve their customers without disruptions.
4. **Regulatory Compliance:** Anticipating bankruptcy helps businesses comply with legal and regulatory requirements related to financial solvency and reporting.
5. **Optimal Resource Allocation:** It allows for efficient resource allocation, including reallocating funds, renegotiating contracts, or securing additional financing to improve financial stability.

Disadvantages:

1. **False Alarms:** Predicting bankruptcy is not foolproof, and false alarms can lead to unnecessary panic or drastic actions that harm the business.
2. **Resource Intensive:** Implementing advanced analytics and monitoring systems can be resource-intensive, requiring financial expertise and technology investments.
3. **Legal and Ethical Challenges:** Accusing a business of financial distress without concrete evidence can lead to legal and ethical challenges, potentially damaging reputations.
4. **Information Asymmetry:** Stakeholders may not have access to the same level of information as business owners, leading to disparities in bankruptcy anticipation.
5. **Overreliance on Metrics:** Relying solely on financial metrics can lead to a myopic view of a company's health, neglecting non-financial factors that contribute to bankruptcy risks.
6. **External Factors:** Economic and industry trends beyond a company's control can significantly impact its financial health, making anticipation challenging.

8.APPLICATIONS

Risk Management for Lenders and Creditors: Financial institutions, lenders, and creditors use bankruptcy anticipation to assess the creditworthiness of borrowers and clients. This helps them make informed decisions about extending loans and credit lines.

Investment Decision-Making: Investors, including equity investors and venture capitalists, employ bankruptcy anticipation to evaluate the financial health of potential investment targets. It aids in making investment decisions that align with risk tolerance and return expectations.

Supplier and Vendor Risk Assessment: Companies rely on bankruptcy anticipation to assess the financial stability of suppliers and vendors. This helps ensure the reliability of the supply chain and mitigate the risk of disruptions due to supplier bankruptcies.

Corporate Governance and Compliance: Business leaders and boards of directors use bankruptcy anticipation to meet their fiduciary responsibilities. It ensures that the company complies with regulatory requirements and maintains financial transparency.

Insolvency Professionals: Professionals in insolvency, such as bankruptcy attorneys, financial advisors, and forensic accountants, utilize bankruptcy anticipation to guide their clients through financial distress and bankruptcy proceedings.

Debt Restructuring: Businesses facing financial difficulties employ bankruptcy anticipation to identify problems early and take proactive measures, such as debt restructuring, to avoid bankruptcy.

9. CONCLUSION

In conclusion, anticipating business bankruptcy is a critical process with far-reaching implications for businesses, investors, creditors, and various stakeholders. The ability to proactively identify and address bankruptcy risks is essential for safeguarding financial health, protecting interests, and maintaining operational continuity. While it is not without its challenges, the advantages of bankruptcy anticipation, such as early intervention, risk mitigation, and regulatory compliance, far outweigh the disadvantages. By leveraging advanced data analytics, expert advisory services, and a multifaceted approach to financial analysis, stakeholders can enhance their ability to anticipate and manage bankruptcy risks effectively. Ultimately, this proactive approach is vital for promoting financial stability, resilience, and long-term success in an ever-changing business landscape.

10.FUTURE SCOPE

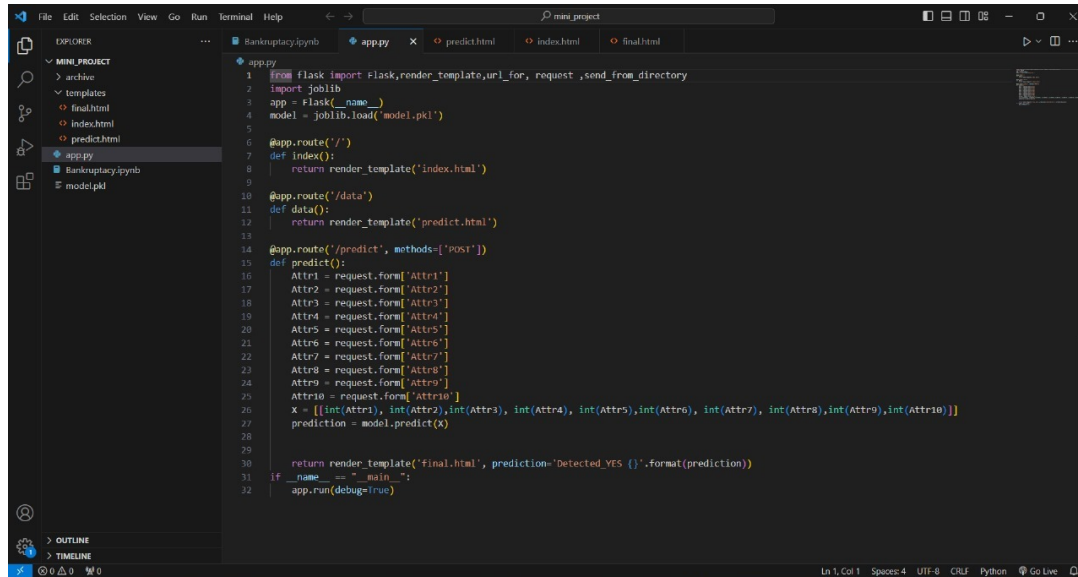
The future scope for anticipating business bankruptcy is promising, driven by advancements in data analytics, artificial intelligence, and financial technology. Real-time monitoring, enhanced predictive models, and the integration of big data will lead to more accurate and timely bankruptcy risk assessment. Moreover, an increased focus on environmental, social, and governance (ESG) factors, global economic integration, and cross-industry applications will broaden the scope of bankruptcy anticipation. As regulations and business landscapes evolve, the demand for education and training in bankruptcy risk management will grow. The field's future lies in its ability to adapt to changing circumstances, offering more sophisticated tools and strategies for businesses and stakeholders

BIBLIOGRAPHY

1. <https://www.investopedia.com/terms/b/bankruptcy.asp>
2. <https://hbr.org/topic/bankruptcy>
3. <https://www.uscourts.gov/services-forms/bankruptcy/bankruptcy-basics>
4. <https://www.thebalance.com/bankruptcy-4073961>

APPENDIX

source code of flask
app.py



```
1 from flask import Flask, render_template, url_for, request, send_from_directory
2 import joblib
3 app = Flask(__name__)
4 model = joblib.load('model.pkl')
5
6 @app.route('/')
7 def index():
8     return render_template('index.html')
9
10 @app.route('/data')
11 def data():
12     return render_template('predict.html')
13
14 @app.route('/predict', methods=['POST'])
15 def predict():
16     Attr1 = request.form['Attr1']
17     Attr2 = request.form['Attr2']
18     Attr3 = request.form['Attr3']
19     Attr4 = request.form['Attr4']
20     Attr5 = request.form['Attr5']
21     Attr6 = request.form['Attr6']
22     Attr7 = request.form['Attr7']
23     Attr8 = request.form['Attr8']
24     Attr9 = request.form['Attr9']
25     Attr10 = request.form['Attr10']
26     x = [[int(Attr1), int(Attr2), int(Attr3), int(Attr4), int(Attr5), int(Attr6), int(Attr7), int(Attr8), int(Attr9), int(Attr10)]]
27     prediction = model.predict(x)
28
29     return render_template('final.html', prediction='Detected_YES {}'.format(prediction))
30
31 if __name__ == '__main__':
32     app.run(debug=True)
```