Bankruptcy Prediction

Group 30

Niraj Chowdhary

Nachiket Rane

Prashmika Jaiswal

Outline

Bankruptcy is different from **insolvency.** Bankruptcy is of the several legal statuses that an insolvent person/entity may end up with.

Bankruptcy is a legal status of a person or other entity that cannot repay debts to creditors.

In most jurisdictions, bankruptcy is imposed by a court order, often initiated by the debtor.

Bankruptcy Prediction

- Bankruptcy prediction is the art of predicting bankruptcy and various measures of financial distress of public firms.
- The history of bankruptcy prediction includes application of numerous statistical tools which gradually became available, and involves deepening appreciation of various pitfalls in early analysis.
- The area is well-suited for testing of increasingly sophisticated, data-intensive forecasting approaches.

Challenges in Bankruptcy Prediction

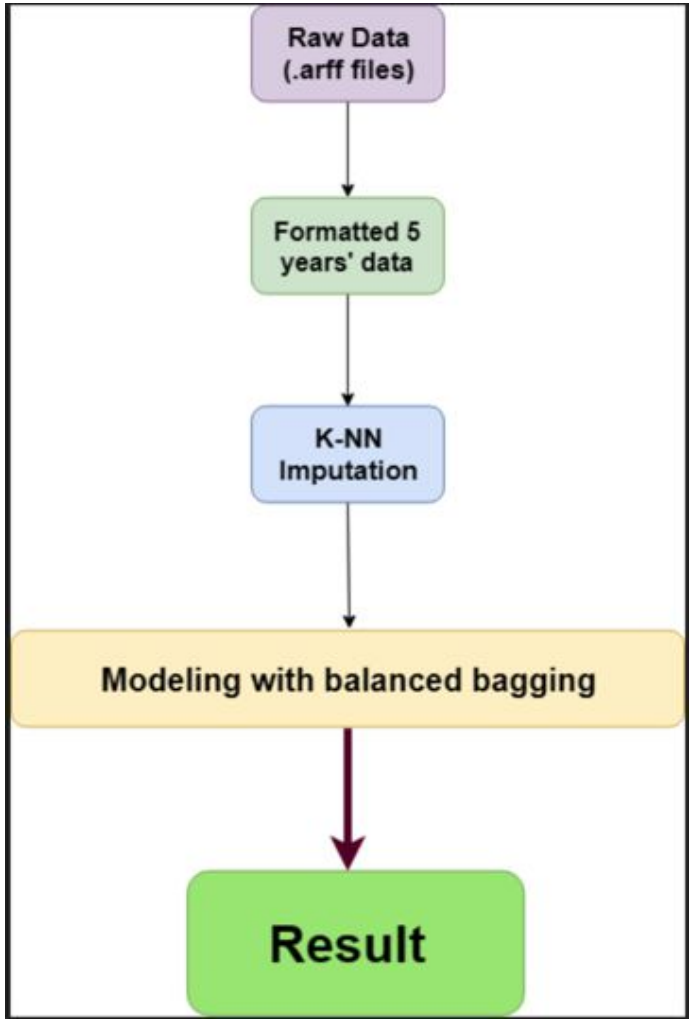
The quantity of research is a function of the availability of data: for public firms which went bankrupt or did not, numerous accounting ratios that might indicate danger can be calculated, and numerous other potential explanatory variables are also available.

Since all the companies don't operate on the same timelines, it is difficult to gather meaningful data about bankrupt companies or otherwise.

Some companies might go bankrupt in the first year itself , while others perform well in the first 2 years but might go bankrupt in the 3rd year.

Difficulty in feature extraction and generation of synthetic features based on some basic features.

Proposed System Analysis



Data Understanding

Polish Dataset

1. Four Year Data set with sixty-four (64) attributes
2. Data Collected from 6 EMIS in the period 2000-2012

id	Attr1	Attr2	Attr3	Attr4	Attr5	Attr6
1	0.20055	0.37951	0.39641	2.0472	32.351	0.38825
2	0.20912	0.49988	0.47225	1.9447	14.786	0
3	0.24866	0.69592	0.26713	1.5548	-1.1523	0
4	0.081483	0.30734	0.45879	2.4928	51.952	0.14988
5	0.18732	0.61323	0.2296	1.4063	-7.3128	0.18732
6	0.22822	0.49794	0.35969	1.7502	-47.717	0
7	0.11109	0.64744	0.28971	1.4705	2.5349	0
8	0.53232	0.027059	0.70554	53.954	299.58	0

Features

The importance of features can be decided by calculating the total number of the feature being observed in the nodes of forest structure by total number of nodes in trees that constitute the forest.

Important Features:

X25 (adjusted share of equity in financing of assets)

X40 (current ratio, the most frequently used ratio in the integrated models)

X52 (liabilities turnover ratio)

ID	Description
X1	net profit / total assets
X2	total liabilities / total assets
X3	working capital / total assets
X4	current assets / short-term liabilities
X5	[(cash + short-term securities + receivables - short-term liabilities) / (operating expenses - depreciation)] * 365
X6	retained earnings / total assets
X7	EBIT / total assets
X8	book value of equity / total liabilities
X9	sales / total assets
X10	equity / total assets
X11	(gross profit + extraordinary items + financial expenses) / total assets
X12	gross profit / short-term liabilities
X13	(gross profit + depreciation) / sales
X14	(gross profit + interest) / total assets
X15	(total liabilities * 365) / (gross profit + depreciation)
X16	(gross profit + depreciation) / total liabilities
X17	total assets / total liabilities
X18	gross profit / total assets
X19	gross profit / sales
X20	(inventory * 365) / sales
X21	sales (n) / sales (n-1)
X22	profit on operating activities / total assets
X23	net profit / sales
X24	gross profit (in 3 years) / total assets
X25	(equity - share capital) / total assets
X26	(net profit + depreciation) / total liabilities
X27	profit on operating activities / financial expenses
X28	working capital / fixed assets
X29	logarithm of total assets
X30	(total liabilities - cash) / sales
X31	(gross profit + interest) / sales
X32	(current liabilities * 365) / cost of products sold

Features

Secondary Features:

X36 (adjusted share of equity in financing of assets)

X42 (current ratio, the most frequently used ratio in the integrated models)

X58 (liabilities turnover ratio)

ID	Description
X33	operating expenses / short-term liabilities
X34	operating expenses / total liabilities
X35	profit on sales / total assets
X36	total sales / total assets
X37	(current assets - inventories) / long-term liabilities
X38	constant capital / total assets
X39	profit on sales / sales
X40	(current assets - inventory - receivables) / short-term liabilities
X41	total liabilities / ((profit on operating activities + depreciation) * (12/365))
X42	profit on operating activities / sales
X43	rotation receivables + inventory turnover in days
X44	(receivables * 365) / sales
X45	net profit / inventory
X46	(current assets - inventory) / short-term liabilities
X47	(inventory * 365) / cost of products sold
X48	EBITDA (profit on operating activities - depreciation) / total assets
X49	EBITDA (profit on operating activities - depreciation) / sales
X50	current assets / total liabilities
X51	short-term liabilities / total assets
X52	(short-term liabilities * 365) / cost of products sold)
X53	equity / fixed assets
X54	constant capital / fixed assets
X55	working capital
X56	(sales - cost of products sold) / sales
X57	(current assets - inventory - short-term liabilities) / (sales - gross profit - depreciation)
X58	total costs /total sales
X59	long-term liabilities / equity
X60	sales / inventory
X61	sales / receivables
X62	(short-term liabilities *365) / sales
X63	sales / short-term liabilities
X64	sales / fixed assets

Data Organization

Data File	Description	# Instances	# Bankrupt instances in this forecasting period	# Non-Bankrupt instances in this forecasting period
Year 1	Data for: 1st year of the forecasting period Label indicates: Bankruptcy status after 5 years.	7027	271	6756
Year 2	Data for: 2nd year of the forecasting period Label indicates: Bankruptcy status after 4 years.	10173	400	9773
Year 3	Data for: 3rd year of the forecasting period Label indicates: Bankruptcy status after 3 years.	10503	495	10008
Year 4	Data for: 4th year of the forecasting period Label indicates: Bankruptcy status after 2 years.	9792	515	9277
Year 5	Data for: 5th year of the forecasting period Label indicates: Bankruptcy status after 1 year.	5910	410	5500

Data Quality Assessment: Missing Data

Data Set	# Total Instances	# Instances with missing values	# Instances that would remain if all rows with missing values were dropped	% Data loss if rows with missing values were dropped
Year 1	7027	3833	3194	54.54 %
Year 2	10173	6085	4088	59.81 %
Year 3	10503	5618	4885	53.48 %
Year 4	9792	5023	4769	51.29 %
Year 5	5910	2879	3031	48.71 %

Implementation

Data Cleaning

Missing data causes 3 problems:

1. Missing data can introduce a substantial amount of bias.
2. Makes the handling and analysis of the data more difficult.
3. Create reductions in efficiency.

Cleaning Data

```
df=df.replace('?',0)
df2=df2.replace('?',0)
df=df.astype(float)
df.Y=df.Y.astype(int)
df2=df2.astype(float)
df2.Y=df2.Y.astype(int)
df=df.replace(0,np.NaN)
df.Y=df.Y.replace(np.NaN,0)
df.Y=df.Y.astype(int)
df2=df2.replace(0,np.NaN)
df2.Y=df2.Y.replace(np.NaN,0)
df2.Y=df2.Y.astype(int)
```

O/P:

	X1	X2	X3	X4	X5	X6	\
0	0.200550	0.379510	0.396410	2.04720	32.35100	0.388250	
1	0.209120	0.499880	0.472250	1.94470	14.78600	NaN	
2	0.248660	0.695920	0.267130	1.55480	-1.15230	NaN	
3	0.081483	0.307340	0.458790	2.49280	51.95200	0.149880	
4	0.187320	0.613230	0.229600	1.40630	-7.31280	0.187320	
5	0.228220	0.497940	0.359690	1.75020	-47.71700	NaN	
6	0.111090	0.647440	0.289710	1.47050	2.53490	NaN	
7	0.532320	0.027059	0.705540	53.95400	299.58000	NaN	
8	0.009020	0.632020	0.053735	1.12630	-37.84200	NaN	
9	0.124080	0.838370	0.142040	1.16940	-91.88300	NaN	
10	0.240010	0.443550	0.188350	1.44000	-21.16500	-0.931900	
11	-0.027117	0.111480	0.119890	2.07540	-31.64300	-0.084883	
12	0.266690	0.349940	0.611470	3.02430	43.08700	0.559830	
13	0.067731	0.198850	0.081562	2.95760	90.60600	0.212650	
14	-0.029182	0.211310	0.452640	7.57460	57.84400	0.010387	
15	-0.033801	1.154000	-0.205990	0.82150	-74.45100	-0.104130	
16	0.270530	0.299130	0.468700	2.56690	73.39500	0.727930	
17	0.028084	0.242310	0.432240	3.01280	47.93500	0.021598	

K- Nearest Neighbour Imputation (KNN)

Library: ***fancyimpute***

KNN: Nearest Neighbour imputations which weighs samples using the mean squared difference on features for which two rows have observed data

```
#----------------------DEALING WITH MISSING DATA----------------
res=pd.DataFrame(fancyimpute.KNN(k=100,verbose=True).fit_transform(df))
res2=pd.DataFrame(fancyimpute.KNN(k=100,verbose=True).fit_transform(df2))
#---------------------------------------------------------------
```

KNN Imputation Processing

```
Imputing row 1/37495 with 0 missing, elapsed time: 3308.548
Imputing row 101/37495 with 2 missing, elapsed time: 3308.915
Imputing row 201/37495 with 7 missing, elapsed time: 3309.083
Imputing row 301/37495 with 0 missing, elapsed time: 3309.251
Imputing row 401/37495 with 0 missing, elapsed time: 3309.411
Imputing row 501/37495 with 6 missing, elapsed time: 3309.582
Imputing row 601/37495 with 2 missing, elapsed time: 3309.727
Imputing row 701/37495 with 2 missing, elapsed time: 3309.893
Imputing row 801/37495 with 0 missing, elapsed time: 3310.260
Imputing row 901/37495 with 0 missing, elapsed time: 3310.548
Imputing row 1001/37495 with 2 missing, elapsed time: 3310.763
Imputing row 1101/37495 with 5 missing, elapsed time: 3310.933
Imputing row 1201/37495 with 6 missing, elapsed time: 3311.137
Imputing row 1301/37495 with 2 missing, elapsed time: 3311.341
Imputing row 1401/37495 with 2 missing, elapsed time: 3311.556
Imputing row 1501/37495 with 1 missing, elapsed time: 3311.749
Imputing row 1601/37495 with 3 missing, elapsed time: 3311.919
```

Table filled with new values

```
In [16]: print(res)
```

	0	1	2	3	4	5	\
0	0.200550	0.379510	0.396410	2.04720	32.35100	0.388250	
1	0.209120	0.499880	0.472250	1.94470	14.78600	0.460364	
2	0.248660	0.695920	0.267130	1.55480	-1.15230	0.488373	
3	0.081483	0.307340	0.458790	2.49280	51.95200	0.149880	
4	0.187320	0.613230	0.229600	1.40630	-7.31280	0.187320	
5	0.228220	0.497940	0.359690	1.75020	-47.71700	0.387304	
6	0.111090	0.647440	0.289710	1.47050	2.53490	0.612595	
7	0.532320	0.027059	0.705540	53.95400	299.58000	0.598300	
8	0.009020	0.632020	0.053735	1.12630	-37.84200	0.131246	
9	0.124080	0.838370	0.142040	1.16940	-91.88300	0.448292	
10	0.240010	0.443550	0.188350	1.44000	-21.16500	-0.931900	
11	-0.027117	0.111480	0.119890	2.07540	-31.64300	-0.084883	
12	0.266690	0.349940	0.611470	3.02430	43.08700	0.559830	
13	0.067731	0.198850	0.081562	2.95760	90.60600	0.212650	
14	-0.029182	0.211310	0.452640	7.57460	57.84400	0.010387	
15	-0.033801	1.154000	-0.205990	0.82150	-74.45100	-0.104130	
16	0.270530	0.299130	0.468700	2.56690	73.39500	0.727930	

Balancing the Dataset

Looking at the numbers of Bankrupt class label, we can figure out that they are a ***minority*** when compare with the non-bankrupt class label.

If this imbalance is not cured, in the modeling stage that follows, the models will not have seen enough data from the minority class label and they train and hence perform poorly.

SMOTE OVERSAMPLING

Library: ***imbalanced learn***

SMOTE: Generates new samples of the classes which are under-represented

```
#----------------------------------------BALANCING THE DATASET-------------------
X_train,X_test,Y_train,Y_test=model_selection.train_test_split(X,Y,test_size=0.20,random_state=7)
print(X_test.shape)
print("Before OverSampling, counts of label '1': {}".format(sum(Y_train==1)))
print("Before OverSampling, counts of label '0': {} \n".format(sum(Y_train==0)))
sm = SMOTE(random_state=2)
X_train_res, Y_train_res = sm.fit_sample(X_train, Y_train.ravel())

print('After OverSampling, the shape of train_X: {}'.format(X_train_res.shape))
print('After OverSampling, the shape of train_y: {} \n'.format(Y_train_res.shape))

print("After OverSampling, counts of label '1': {}".format(sum(Y_train_res==1)))
print("After OverSampling, counts of label '0': {}".format(sum(Y_train_res==0)))
#------------------------------------------------------------
```

SMOTE Results

```
(7499, 63)
Before OverSampling, counts of label '1': 1335
Before OverSampling, counts of label '0': 28661

After OverSampling, the shape of train_X: (57322, 63)
After OverSampling, the shape of train_y: (57322,)

After OverSampling, counts of label '1': 28661
After OverSampling, counts of label '0': 28661
```

Model Comparison (using Cross Validation)

Cross-validation is a resampling procedure used to evaluate machine learning models on a limited data sample. It estimates the skill of a machine learning model on unseen data.

We've used this method to compare four models:

1. Logistic Regression
2. Naive Bayesian
3. Decision Tree Classifier
4. Balanced Bagging Classifier

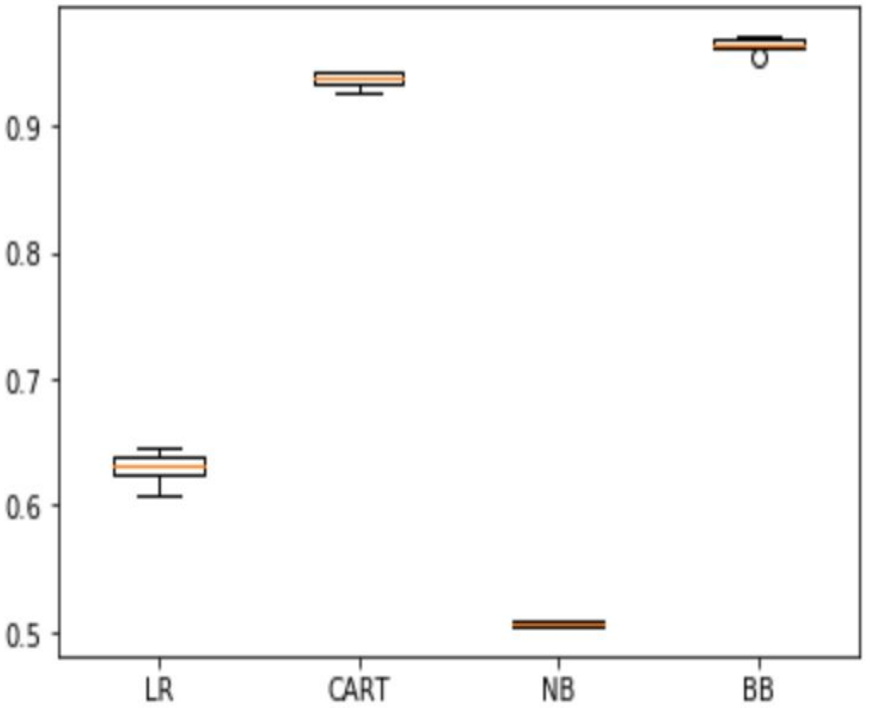
Cross Validation Score Evaluation

```
#------------------------------TRAINING THE MODELS ------------------------------
seed = 7
scoring = 'accuracy'
models = []
models.append(('LR', LogisticRegression()))
models.append(('NB', GaussianNB()))
models.append(('CART', DecisionTreeClassifier()))
models.append(('BB', BalancedBaggingClassifier()))
results = []
names = []
msg=''
for name, model in models:

    cv_results = model_selection.cross_val_score(model, X_train_res, Y_train_res, cv=10, scoring=scoring)
    results.append(cv_results)
    names.append(name)
    msg+= "%s--> %f \n "  %(name, cv_results.mean()*100)

popupmsg(msg)
#---------------------------------------------------------------

#-------------------------------------algo comparision chart-------------------------------
fig = plt.figure()
fig.suptitle('Algorithm Comparison')
ax = fig.add_subplot(111)
plt.boxplot(results)
ax.set_xticklabels(names)
plt.show()
#-----------------------------------------------------
```

Algorithm Comparison

BB showed best evaluation score

```
LR: 0.630788
CART: 0.937337
NB: 0.506996
BB: 0.964900
```



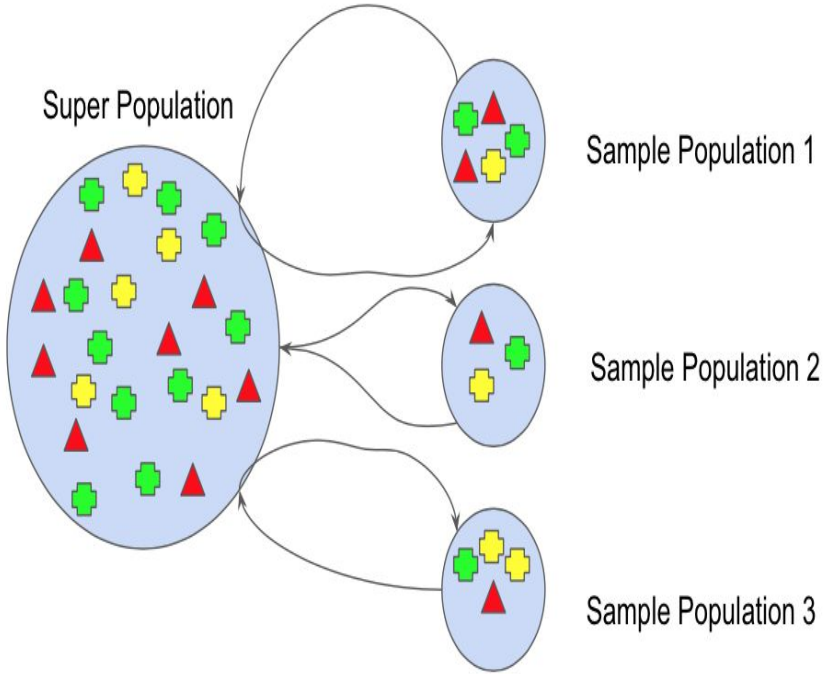
MODELING

One classifier may be severely affected when the training dataset cannot well characterize the actual underlying distribution or the presumed model is biased.

The strategy of ***models' ensemble*** can avoid the one-sidedness originated from the training dataset and hypothesis, receiving a better capability of generalization.

Bootstrapping

Random Sampling with Replacement

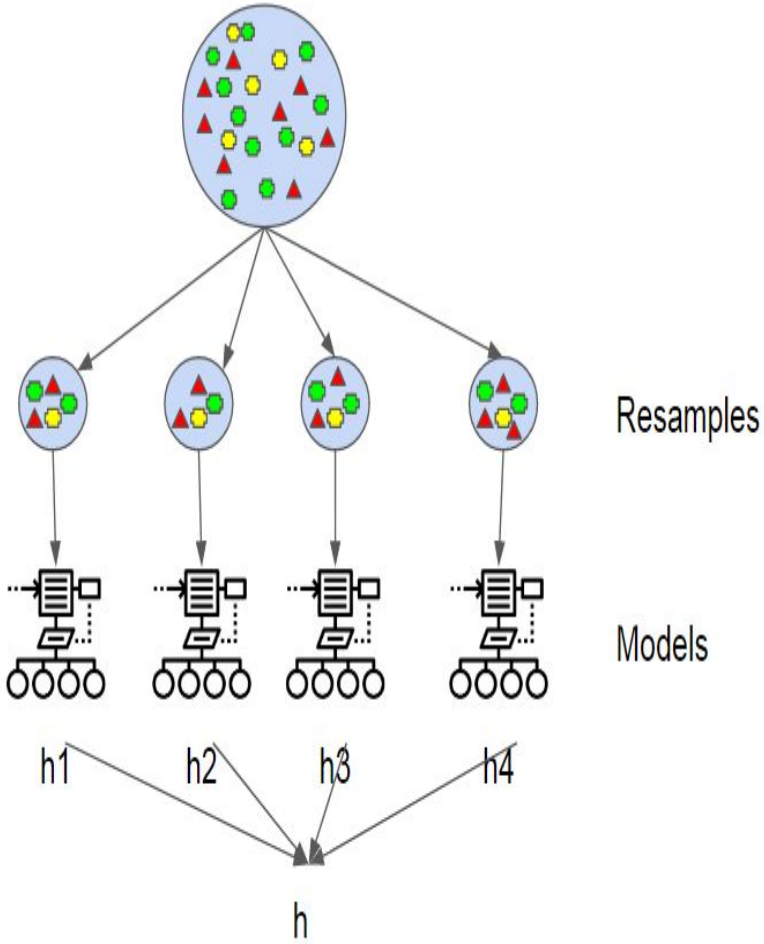


Bagging

Bootstrap + **Ag**gregation

⇩

Bagging



BALANCED BAGGING (BB)

Module: ***imlearn.ensemble***

BB: A Bagging classifier with additional balancing for better prediction accuracy. It includes an additional step to balance the training set at fit time.

BB Classifier Performance

```
#---------------------------------TESTING OUR MODEL-----------------------------------
balbag= BalancedBaggingClassifier();
balbag.fit(X_train_res, Y_train_res)

predictions = balbag.predict(X_test)
accur="Accuracy of test data:"+str(accuracy_score(Y_test, predictions)*100)
popupmsg(accur)
print(confusion_matrix(Y_test, predictions))
```

Accuracy of the test data: 0.945326043472463

Confusion Matrix:

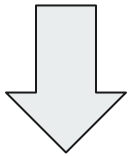
```
[[6877  276]
 [ 134  212]]
```

Classification (Bankruptcy Prediction)

Base Estimator: RandomForestClassifier()

```
balbag= BalancedBaggingClassifier(RandomForestClassifier());
predictions2=balbag.predict(X2)
popupmsg("Accuracy of unseen data:"+str(accuracy_score(Y2, predictions2)*100))
predictions2=predictions2.astype(int)

output=''
c=1;
for i in predictions2:

    if i==1:
       output+="Input "+str(c)+"-->Bankrupt\n"
    else:
        output+="Input "+str(c)+"-->Not Bankrupt\n"
    c=c+1;

popupmsg(output)
#--------------------------------END-----------------------------------END-------------------------------------
```

Predicted = 0



No Bankruptcy

```
X=[ 5.38190000e-02  8.59890000e-01  5.69720000e-02  1.12570000e+00
 -1.22350000e+03  4.56968942e-01  5.73430000e-02  1.62950000e-01
  6.19520000e+00  1.40110000e-01  1.06750000e-01  1.26560000e-01
  1.47250000e-02  5.73430000e-02  3.44050000e+03  1.06090000e-01
  1.16290000e+00  5.73430000e-02  9.25600000e-03  1.29780000e+01
  1.01220000e+00  1.06750000e-01  8.68710000e-03  1.74490000e-01
  1.34860000e-01  1.01990000e-01  2.16070000e+00  1.16280000e-01
  3.23360000e+00  1.18460000e-01  1.64140000e-02  2.72720000e+01
  1.34380000e+01  7.08060000e+00  1.31340000e-01  6.19520000e+00
  7.12290000e-01  5.46920000e-01  2.12000000e-02  2.78080000e-01
  2.03820000e-01  1.72310000e-02  2.26270000e+01  9.64890000e+00
  2.44320000e-01  6.39550000e-01  1.32060000e+01  7.28680000e-02
  1.17620000e-02  5.93160000e-01  4.53080000e-01  7.44150000e-02
  2.85980000e-01  1.11630000e+00  9.75640000e+01  2.12000000e-02
  3.84110000e-01  9.90740000e-01  2.90340000e+00  2.81240000e+01
  3.78280000e+01  2.66940000e+01  1.36740000e+01], Predicted=0.0
```

Predicted = 1

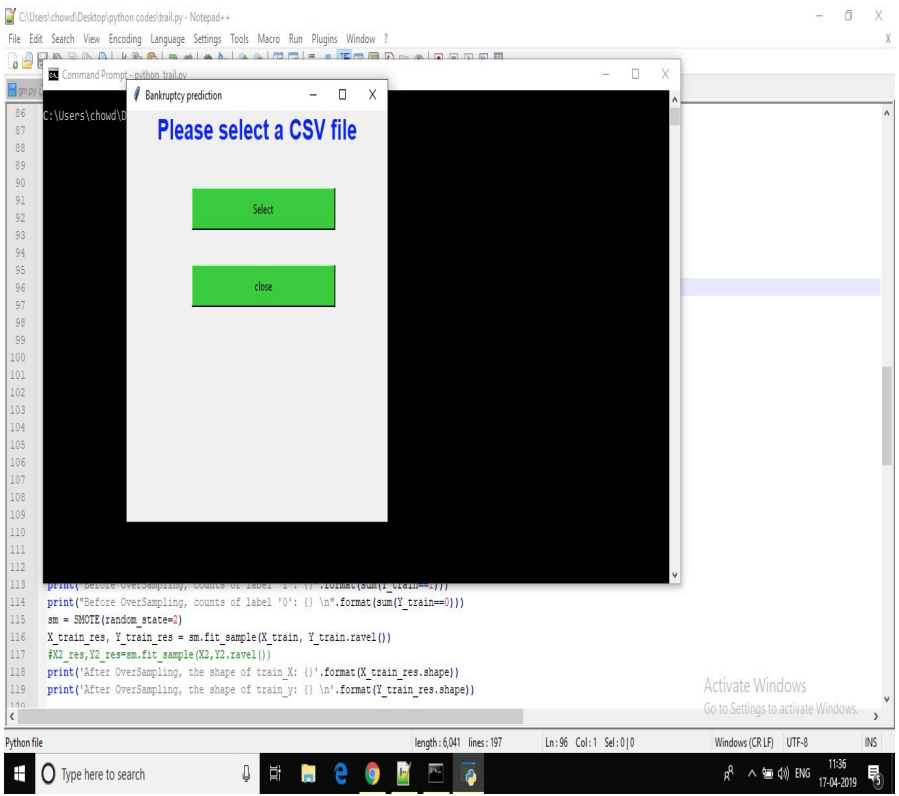
Bankruptcy

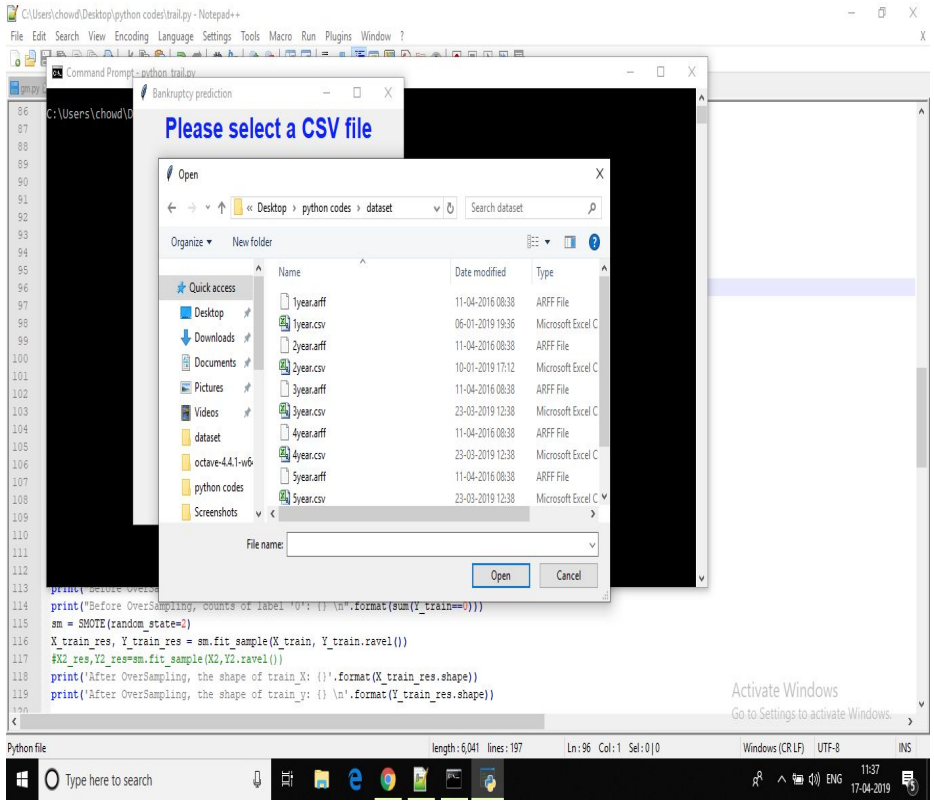
```
X=[-1.7710e-02  6.4132e-01  1.0461e-01  1.2421e+00 -3.4098e+01 -9.4964e-02
 -2.0314e-02  5.4698e-01  9.9926e-01  3.5079e-01 -2.0314e-02 -4.7011e-02
  4.0803e-02 -2.0314e-02  4.4107e+03  8.2754e-02  1.5593e+00 -2.0314e-02
 -1.5618e-02  6.1260e+01  1.3627e+00 -4.6700e-04 -1.3616e-02 -1.0971e-01
  3.5079e-01  8.6814e-02 -3.5860e-03  2.2580e-01  4.7566e+00  4.3851e-01
 -1.5618e-02  1.2117e+02  3.0122e+00 -7.2800e-04 -4.6700e-04  1.3062e+00
  1.5221e+00  5.6000e-01 -3.5900e-04  1.7033e-01  2.8915e-01 -3.5900e-04
  1.2996e+02  6.8703e+01 -8.1129e-02  7.3690e-01  6.1215e+01 -7.3853e-02
 -5.6780e-02  8.3691e-01  4.3212e-01  3.3198e-01  7.5721e-01  1.2088e+00
  5.9729e+03 -7.4200e-04 -5.0487e-02  1.0007e+00  5.9637e-01  5.9582e+00
  5.3127e+00  1.2126e+02  3.0100e+00], Predicted=1.0
```

GUI

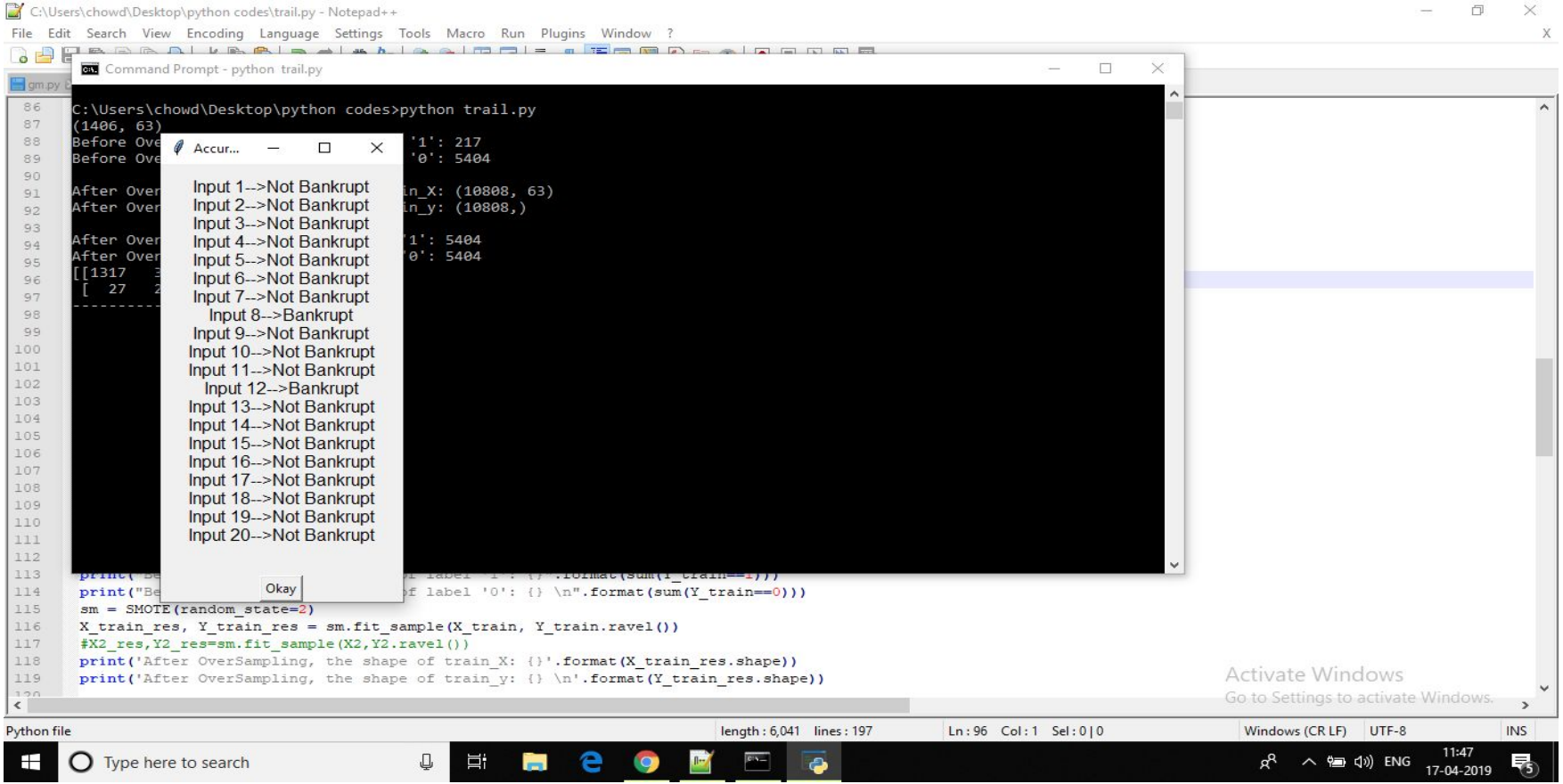
This simple GUI makes using the system by varied people possible. It provides a clean experience to novice as well as expert users.

File Selection





Predictions




Thank You
Conclusion and Future Work