TITLE: BANKRUPTCY PREDICTION USING MACHINE LEARNING

Submitted By : Deelasha Mulmi

Submitted To : Code Rush

OCTOBER, 2022

1.1 INTRODUCTION:

Bankruptcy prediction is the problem of detecting financial distress in businesses which will lead to eventual bankruptcy. Bankruptcy prediction has been studied since at least 1930s. The early models of bankruptcy prediction employed univariate statistical models over financial ratios. The univariate models were followed by multi-variate statistical models such as the famous Altman Z-score model. The recent advances in the field of Machine learning have led to the adoption of Machine learning algorithms for bankruptcy prediction. Machine Learning methods are increasingly being used for bankruptcy prediction using financial ratios.

1.2 Questions

a) How accurate will be the model?

At the end of the project, we will get to know how accurate will be the model.Both algorithms will perform accurately or not will be the main goal of this project.

b) Comparison of two different algorithms.

We will be building the model using two machine learning algorithms. One is Logistic Regression and another one is Support Vector Machine(SVM).After building the model, we will compare the accuracy, roc and auc of both algorithms.

2.1 Data:

The data were collected from the Taiwan Economic Journal for the years 1999 to 2009. Bankruptcy was defined based on the business regulations of the Taiwan Stock Exchange. The data was obtained from UCI Machine Learning Repository: https://archive.ics.uci.edu/ml/datasets/Taiwanese+Bankruptcy+Prediction

The dataset have 6819 rows and 96 columns.

2.2 Methodology:

2.2.1 Data Preprocessing:

The first step of building a predictive model is data pre-processing and cleaning. The original data from Taiwan Stock Exchange had 96 columns and 6819 rows. This data covered firms established between 1999 and 2009. The dataset contained firms that belonged to 2 classes: bankrupt and non-bankrupt. The dataset contains 220 bankrupt firms and 6599 non-bankrupt firms. As we check if there is any null values present in the dataset using function isna().any(). We got to know that there is not any null values in our dataset. To check if we have any duplicate data we use data.duplicated().sum().The duplicity is zero. i.e we don’t have any duplicate data in our dataset.

For the exploratory analysis, we calculate the value count of column Bankrupt? And plot the count plot of Bankrupt? Which is shown in given below figure. We have also plot the histogram of Bankrupt? Column which is also shown below. We have also plot the regplot and relplot between Bankrupt? and Liability to Equity

```
# To check whether data have duplicacy or not
data.duplicated().sum()
```

```
0
```

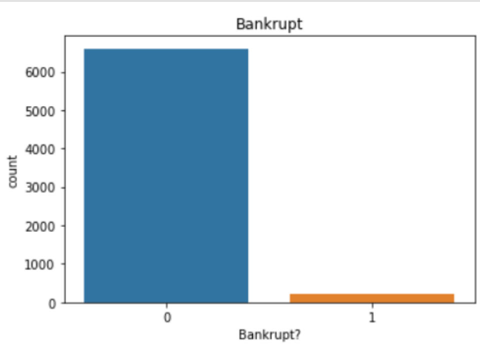
```
Data Cleaning
```

```
# Data cleaning
data.isna().any()
```

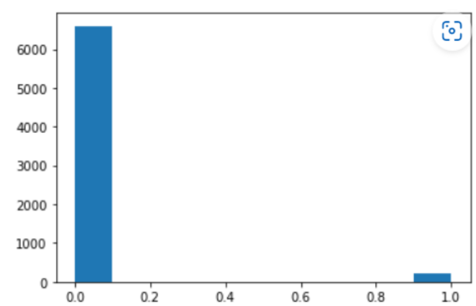
```
Bankrupt?                                                False
 ROA(C) before interest and depreciation before interest False
 ROA(A) before interest and % after tax                  False
 ROA(B) before interest and depreciation after tax       False
 Operating Gross Margin                                  False
                                                          ...
 Liability to Equity                                     False
 Degree of Financial Leverage (DFL)                      False
 Interest Coverage Ratio (Interest expense to EBIT)      False
 Net Income Flag                                         False
 Equity to Liability                                     False
Length: 96, dtype: bool
```

Figure : Data inspection

```
plt.figure()
sns.countplot(x = 'Bankrupt?',data = data )
plt.title('Bankrupt')
plt.show()
```




```
plt.hist(x='Bankrupt?',data=data)
plt.show()
```



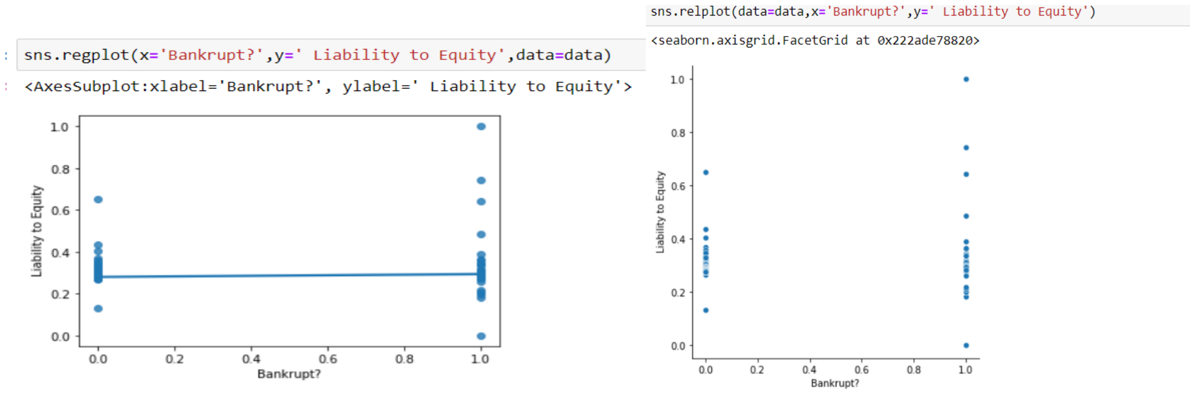


Figure: Plotting of data

2.2.2 Fitting training data into models

The dataset was then shuffled and split into training set containing 80% of the samples and test set containing 20% of the samples. The training and testing data was scaled using standard scalar() which is provided by scikit-learn library. The purpose of creating a test set is to test the accuracy of the models on data that the models have not been trained on.

The training data set was fitted into two machine learning models. These models are: Logistic Regression, Support Vector Machine (SVM). After fitting, the models were then used to predict for samples in the test set to assess their relative performance.

2.2.3 Performance analysis

For comparing the performance of the models, we decided to use Accuracy score, Receiver Operating Curve (ROC) and Area Under ROC Curve (AUC). Accuracy score can be used because we are training our models using a cleaned dataset. However, to get a better idea of the True Positive Rate (TPR) and False Positive

Rate (FPR) we decided to employ ROC and AUC metrics as well. We have also calculated the classification report.

2.2.4 Predicting bankruptcy

The goal is to compare the accuracy of the two models and to predict whether a company will be bankrupt or not. The models were built and analyzed using the same approach for both algorithms. After building the model, we compare the accuracies of two algorithms.

2.2 Result

We have build the two models using two algorithms i.e Logistic regression and Support Vector Machine(SVM). After building the models and analyzing it we compare the accuracy, ROC and AUC of the models build by using two algorithms. We can observe the result in given figure.

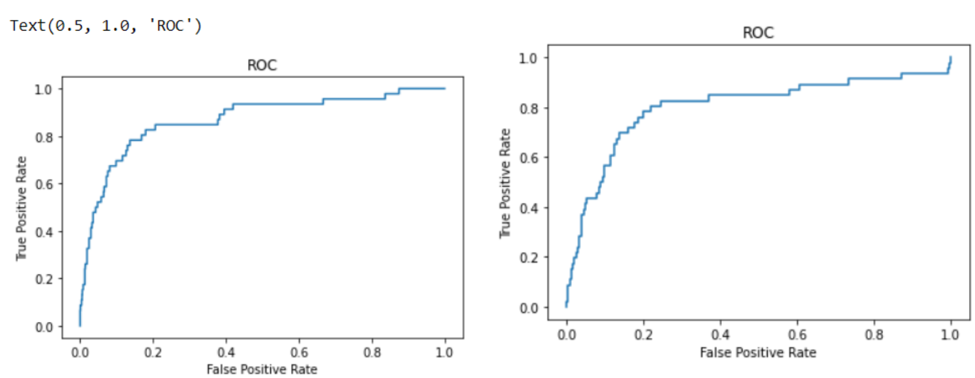


Figure : Roc of Logistic Regression Model and SVM model

```
print('Area under the curve is',auc)
```

```
Area under the curve is 0.8023685425875832
```

```
print('Area under the curve is',auc)
```

```
Area under the curve is 0.8686827868311671
```

Figure: AUC of Logistic Regression and SVM.

AUC for the model using Logistic Regression algorithm is 0.80236 whereas the AUC for the model using SVM is 0.86868.

We have calculated the train as well as test accuracy of both models which is shown in below the figure.

```
from sklearn.metrics import accuracy_score
acc1 = accuracy_score(y_test, y_pred)
print("Accuracy score for Logistic Regression Model: {:.2f} %".format(acc1*100))
```

```
Accuracy score for Logistic Regression Model: 96.63 %
```

```
from sklearn.metrics import accuracy_score
acc2 = accuracy_score(y_train, trained_model.predict(x_train))
print("Accuracy score of training for Logistic Regression Model: {:.2f} %".format(acc2*100))
```

```
Accuracy score of training for Logistic Regression Model: 97.20 %
```

```
from sklearn.metrics import accuracy_score
acc3 = accuracy_score(y_test, y_pred2)
print("Accuracy score for SVM  Model: {:.2f} %".format(acc3*100))
```

```
Accuracy score for SVM  Model: 96.70 %
```

```
trained_model2 = classifier.fit(x_train,y_train)
trained_model2.fit(x_train,y_train)
```

```
SVC(probability=True)
```

```
from sklearn.metrics import accuracy_score
acc4 = accuracy_score(y_train, trained_model2.predict(x_train))
print("Accuracy score of training for SVM Model: {:.2f} %".format(acc4*100))
```

```
Accuracy score of training for SVM Model: 97.18 %
```

Figure : Train and Test Accuracy of Logistic Regression and SVM

The test accuracy of model using logistic regression is 96.63% whereas the test accuracy of model using SVM is 96.70%. The train accuracy using Logistic Regression is 97.20% and train accuracy using SVM is 97.18%.

3. Conclusions and Discussions

3.1 Conclusion

From above analysis we can conclude that the two machine learning algorithms, Logistic Regression and Support Vector Machine (SVM) produce accurate predictions of whether a firm will go bankrupt. Both algorithms perform well with 96% test accuracy and 97% train accuracy.

3.2 Future Research

We identify the following areas for further research:

.

● Train deep neural networks with different topologies—Another interesting area of research would be to apply different types of deep neural networks.

4. Appendices:

4.1 Theories:

Logistic Regression:

The logistic regression model is a two class model. It selects different features and weights to classify the samples, and calculates the probability of the samples belonging to a certain class with each log function. That is, a sample will have a certain probability, belong to a class, there will be a certain probability, belong to another class; the probability of large class is the sample belongs to the class.

Support Vector Machine (SVM):

SVM is a supervised machine learning algorithm that can be used for both classification or regression challenges. However, it is mostly used in classification problems. In the SVM algorithm, we plot each data item as a point in n-dimensional space (where n is a number of features you have) with the value of each feature being the value of a particular coordinate. Then, we perform classification by finding the hyper-plane that differentiates the two classes very well.

4.2 Codes

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

```
data = pd.read_csv(r'E:\data.csv')
```

```
data
```

	Bankrupt?	ROA(C) before interest and depreciation before interest	ROA(A) before interest and % after tax	ROA(B) before interest and depreciation after tax	Operating Gross Margin	Realized Sales Gross Margin	Operating Profit Rate	Pre-tax net Interest Rate	After-tax net Interest Rate	Non-industry income and expenditure/revenue	...	Net Income to Total Assets	Total assets to GNP price	No credi Interva
0	1	0.370594	0.424389	0.405750	0.601457	0.601457	0.998969	0.796887	0.808809	0.302646	...	0.716845	0.009219	0.62287
1	1	0.464291	0.538214	0.516730	0.610235	0.610235	0.998946	0.797380	0.809301	0.303556	...	0.795297	0.008323	0.62365
2	1	0.426071	0.499019	0.472295	0.601450	0.601364	0.998857	0.796403	0.808388	0.302035	...	0.774670	0.040003	0.62384
3	1	0.399844	0.451265	0.457733	0.583541	0.583541	0.998700	0.796967	0.808966	0.303350	...	0.739555	0.003252	0.62292
4	1	0.465022	0.538432	0.522298	0.598783	0.598783	0.998973	0.797366	0.809304	0.303475	...	0.795016	0.003878	0.62352
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
6814	0	0.493687	0.539468	0.543230	0.604455	0.604462	0.998992	0.797409	0.809331	0.303510	...	0.799927	0.000466	0.62362
6815	0	0.475162	0.538269	0.524172	0.598308	0.598308	0.998992	0.797414	0.809327	0.303520	...	0.799748	0.001959	0.62393
6816	0	0.472725	0.533744	0.520638	0.610444	0.610213	0.998984	0.797401	0.809317	0.303512	...	0.797778	0.002840	0.62415
6817	0	0.506264	0.559911	0.554045	0.607850	0.607850	0.999074	0.797500	0.809399	0.303498	...	0.811808	0.002837	0.62395
6818	0	0.493053	0.570105	0.549548	0.627409	0.627409	0.998080	0.801987	0.813800	0.313415	...	0.815956	0.000707	0.62668

Data Inspection

```
]: data.shape
```

```
]: (6819, 96)
```

```
]: data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6819 entries, 0 to 6818
Data columns (total 96 columns):
 #   Column                                                    Non-Null Count  Dtype
---  ------                                                    --------------  -----
 0   Bankrupt?                                                 6819 non-null   int64
 1    ROA(C) before interest and depreciation before interest  6819 non-null   float64
 2    ROA(A) before interest and % after tax                   6819 non-null   float64
 3    ROA(B) before interest and depreciation after tax        6819 non-null   float64
 4    Operating Gross Margin                                   6819 non-null   float64
 5    Realized Sales Gross Margin                              6819 non-null   float64
 6    Operating Profit Rate                                    6819 non-null   float64
 7    Pre-tax net Interest Rate                                6819 non-null   float64
 8    After-tax net Interest Rate                              6819 non-null   float64
 9    Non-industry income and expenditure/revenue              6819 non-null   float64
 10   Continuous interest rate (after tax)                     6819 non-null   float64
 11   Operating Expense Rate                                   6819 non-null   float64
 12   Research and development expense rate                    6819 non-null   float64
 13   Cash flow rate                                           6819 non-null   float64
 14   Interest-bearing debt interest rate                      6819 non-null   float64
 15   Tax rate (A)                                             6819 non-null   float64
 16   Net Value Per Share (B)                                  6819 non-null   float64
 17   Net Value Per Share (A)                                  6819 non-null   float64
 18   Net Value Per Share (C)                                  6819 non-null   float64
 19   Persistent EPS in the Last Four Seasons                  6819 non-null   float64
 20   Cash Flow Per Share                                      6819 non-null   float64
 21   Revenue Per Share (Yuan ¥)                               6819 non-null   float64
 22   Operating Profit Per Share (Yuan ¥)                      6819 non-null   float64
```

```
# For descriptive statistics
data.describe()
```

	Bankrupt?	ROA(C) before interest and depreciation before interest	ROA(A) before interest and % after tax	ROA(B) before interest and depreciation after tax	Operating Gross Margin	Realized Sales Gross Margin	Operating Profit Rate	Pre-tax net Interest Rate	After-tax net Interest Rate	Non-industry income and expenditure/revenue	...	Net Inc to As
count	6819.000000	6819.000000	6819.000000	6819.000000	6819.000000	6819.000000	6819.000000	6819.000000	6819.000000	6819.000000	...	6819.00
mean	0.032263	0.505180	0.558625	0.553589	0.607948	0.607929	0.998755	0.797190	0.809084	0.303623	...	0.80
std	0.176710	0.060686	0.065620	0.061595	0.016934	0.016916	0.013010	0.012869	0.013601	0.011163	...	0.04
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.00
25%	0.000000	0.476527	0.535543	0.527277	0.600445	0.600434	0.998969	0.797386	0.809312	0.303466	...	0.79
50%	0.000000	0.502706	0.559802	0.552278	0.605997	0.605976	0.999022	0.797464	0.809375	0.303525	...	0.81
75%	0.000000	0.535563	0.589157	0.584105	0.613914	0.613842	0.999095	0.797579	0.809469	0.303585	...	0.82
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	...	1.00

8 rows × 96 columns

```
# To check whether data have duplicacy or not
data.duplicated().sum()
```

```
0
```

Data Cleaning

```
# Data cleaning
data.isna().any()
```

```
Bankrupt?                                                   False
 ROA(C) before interest and depreciation before interest    False
 ROA(A) before interest and % after tax                     False
 ROA(B) before interest and depreciation after tax          False
 Operating Gross Margin                                     False
                                                            ...
 Liability to Equity                                        False
 Degree of Financial Leverage (DFL)                         False
 Interest Coverage Ratio (Interest expense to EBIT)         False
 Net Income Flag                                            False
 Equity to Liability                                        False
Length: 96, dtype: bool
```

Exploratory Analysis

```
# Checking the target columns value
data['Bankrupt?'].value_counts()
```

```
0    6599
1     220
Name: Bankrupt?, dtype: int64
```

Predictive Modelling Using Logistic Regression

```
from sklearn.model_selection import train_test_split
```

```
x_train, x_test, y_train, y_test = train_test_split(
    data.drop(labels=['Bankrupt?'], axis=1),
    data['Bankrupt?'],
    test_size=0.2,
    random_state=0)
```

```
from sklearn.preprocessing import StandardScaler
```

```
st= StandardScaler()
x_train =st.fit_transform(x_train)
x_test = st.fit_transform(x_test)
```

```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
model.fit(x_train, y_train)
```

```
E:\anaconda\lib\site-packages\sklearn\linear_model\_logistic.py:763: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

```
LogisticRegression()
```

```
y_pred= model.predict(x_test)
```

```
y_train.shape
```

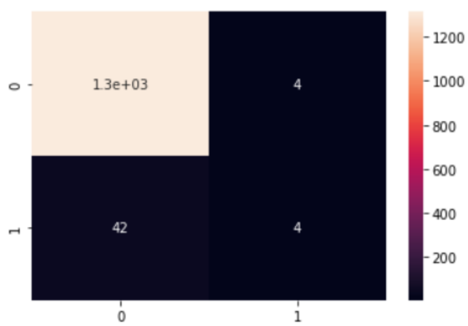
3]:

```
from sklearn.metrics  import confusion_matrix
cm =confusion_matrix(y_test,y_pred)
print(cm)
```

```
[[1314    4]
 [  42    4]]
```

4]:

```
import seaborn as  sns
ax = sns.heatmap(cm,annot=True)
```



5]:

```
from sklearn.metrics import classification_report
target= ['No Bankrupt','Bankrupt']
print(classification_report(y_test,y_pred,target_names =target))
```

```
              precision    recall  f1-score   support

 No Bankrupt       0.97      1.00      0.98      1318
    Bankrupt       0.50      0.09      0.15        46

    accuracy                           0.97      1364
   macro avg       0.73      0.54      0.57      1364
```

```
from sklearn.svm import SVC
```

```
classifier = SVC(kernel = 'linear', random_state = 0)
classifier = SVC(probability=True)
classifier.fit(x_train, y_train)
```

```
SVC(probability=True)
```

```
y_pred2=classifier.predict(x_test)
```

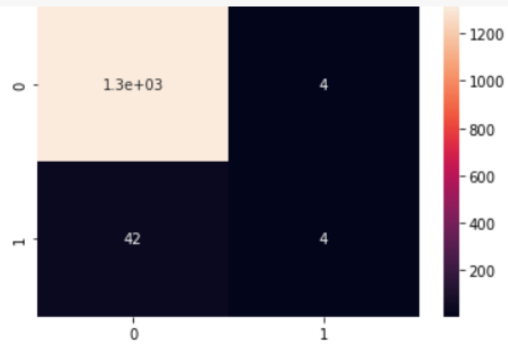
```
y_pred2.shape
```

```
(1364,)
```

```
from sklearn.metrics  import confusion_matrix
cm2 =confusion_matrix(y_test,y_pred2)
print(cm2)
```

```
[[1318    0]
 [  45    1]]
```

```
ax = sns.heatmap(cm,annot=True)
```




```
]: from sklearn.metrics import classification_report
   target= ['No Bankrupt','Bankrupt']
   print(classification_report(y_test,y_pred2,target_names =target))
```

```
              precision    recall  f1-score   support

 No Bankrupt       0.97      1.00      0.98      1318
    Bankrupt       1.00      0.02      0.04        46

    accuracy                           0.97      1364
   macro avg       0.98      0.51      0.51      1364
weighted avg       0.97      0.97      0.95      1364
```

Figure: Codes for obtaining the results

5. References:

[1] Zhang W(2017) Machine Learning Methods of Bankruptcy Prediction Using Accounting Ratios. Journal of Financial Risk Managemnet,6.

[2] Yachao L ,Wang Y (2018) Machine Learning Approaches to Predicting Company Bankruptcy. Open journal of Business and Management ,6.

[3] Narvekar A, Guha D (2021) Bankruptcy prediction using machine learning and an application to the case of the COVID-19 recession Data Science in Finance and Economics,1.