Bankruptcy prediction

of Polish companies

KV Varun S20190020222
Vineesh P S20190020243
Amruth P S20190020242

1 Data-set description

The data-set is about bankruptcy prediction of Polish companies.The bankrupt companies were analyzed in the period 2000-2012, while the still operating companies were evaluated from 2007 to 2013.

It contains financial rates from 5^{th} year of the forecasting period and corresponding class label that indicates bankruptcy status after 1 year. The data contains 5910 instances (financial statements), 410 represents bankrupted companies, 5500 firms that did not bankrupt in the forecasting period.

Each instance/company has 64 attributes on which the classification is evaluated. This makes the data-set multivariate.

Class labels
We have two class labels representing the bankruptcy status of respective company at the end of forecasting period.

- 1 - Indicates that the company went bankrupt.
- 0 - Indicates that the company didn't went bankrupt.

Attributes (X)
Few of the 64 attributes are:

1. X1 net profit / total assets
2. X3 working capital / total assets
3. X19 gross profit / sales

Data-set repository
https://archive.ics.uci.edu/ml/datasets/Polish+companies+bankruptcy+data

2 Problem statement

- Our objective is to classify (binary classification) the given data and to predict corresponding labels with good accuracy for test data containing bank's details.
- Use feature extraction methods to increase accuracy of the model.

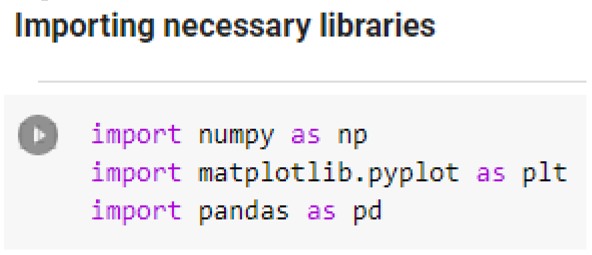
3 Methodology

1. Importing necessary libraries
2. Importing bankruptcy data-set
3. Processing missing values
4. Splitting data-set into training and test sets
5. Training the logistic regression model, Linear Support Vector Machine, Polynomial Support Vector Machine, Radial basis function SVM, K-NN model
6. Predicting the target labels of test-set
7. Performing Feature extraction
8. Comparing results after and before feature extraction.

Python code implement the above methodology
https://github.com/kvv1618/Bankruptcy-prediction

4 Explanation

1. Step 1

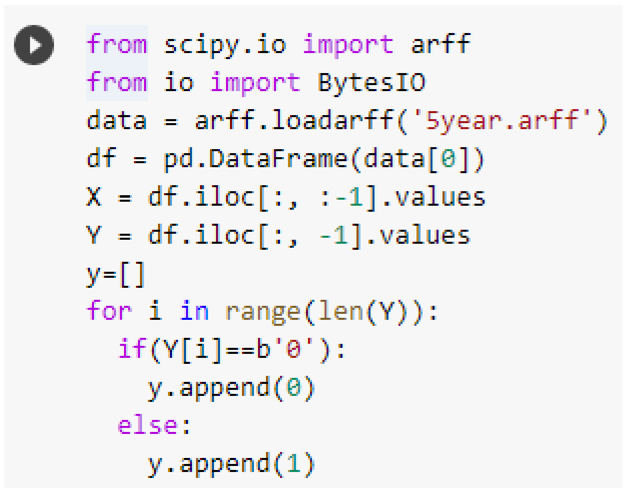


Importing necessary libraries

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

2. Step 2

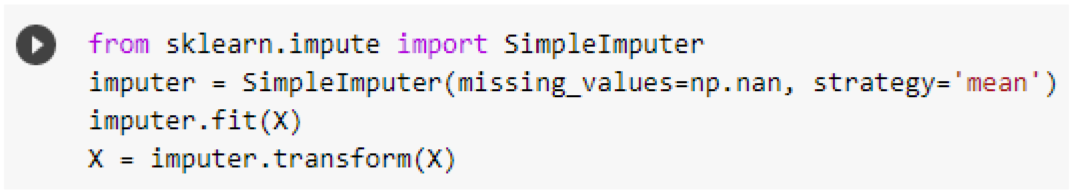
Importing bankruptcy dataset



```python
from scipy.io import arff
from io import BytesIO
data = arff.loadarff('5year.arff')
df = pd.DataFrame(data[0])
X = df.iloc[:, :-1].values
Y = df.iloc[:, -1].values
y=[]
for i in range(len(Y)):
  if(Y[i]==b'0'):
    y.append(0)
  else:
    y.append(1)
```

3. Step 3

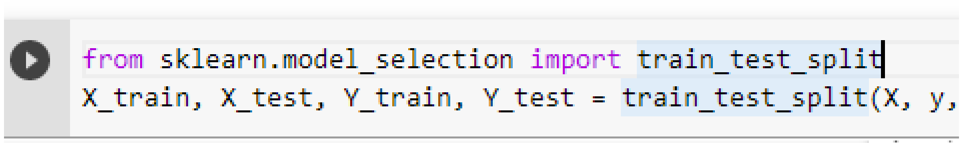
Processing missing values



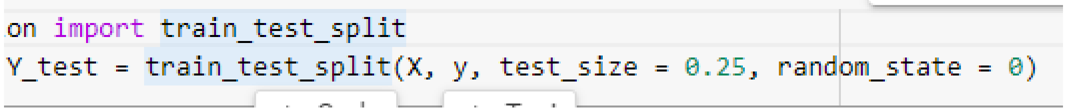
```python
from sklearn.impute import SimpleImputer
imputer = SimpleImputer(missing_values=np.nan, strategy='mean')
imputer.fit(X)
X = imputer.transform(X)
```

4. Step 4

Splitting dataset into training and test sets

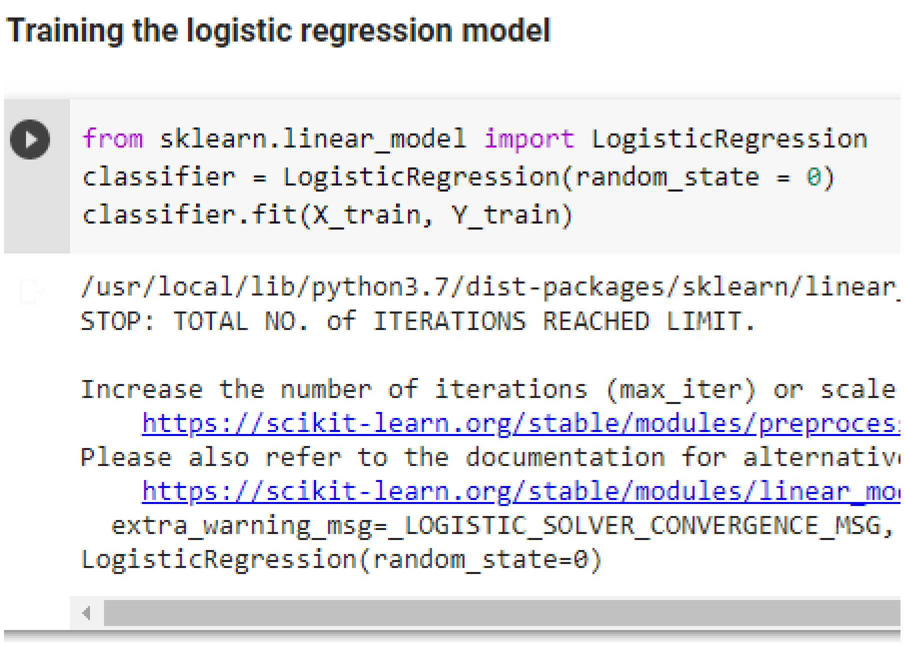


```python
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X, y,
```



```python
on import train_test_split
Y_test = train_test_split(X, y, test_size = 0.25, random_state = 0)
```

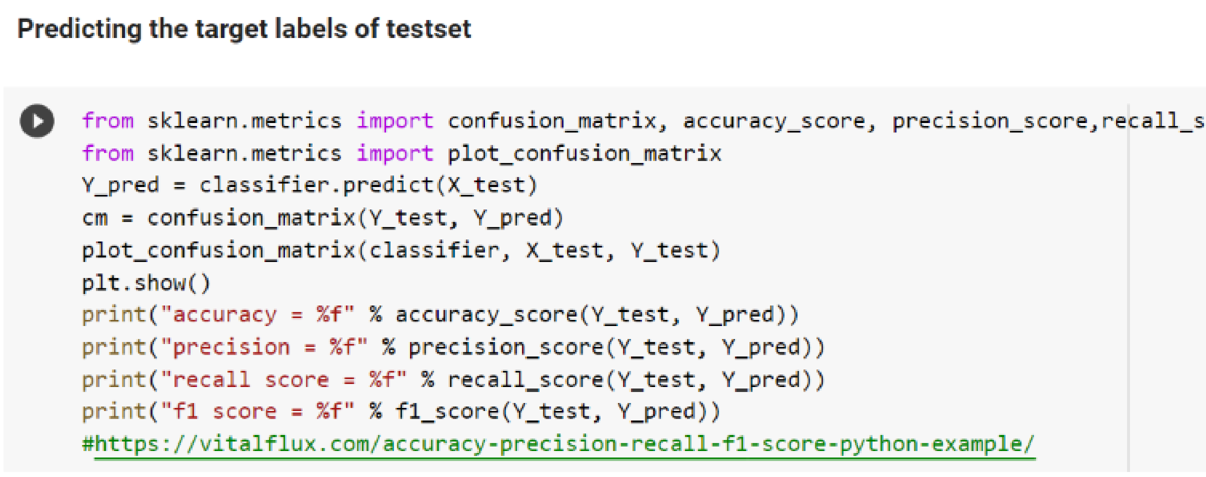
5. Logistic regression before feature extraction



Training the logistic regression model

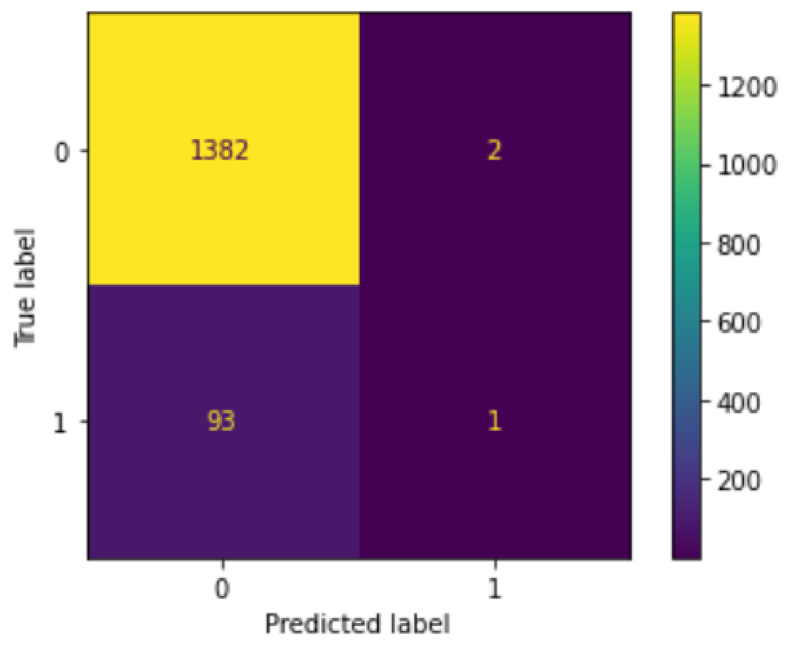
```
from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression(random_state = 0)
classifier.fit(X_train, Y_train)
```

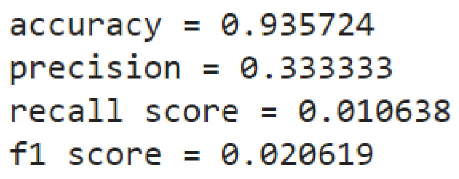
```
/usr/local/lib/python3.7/dist-packages/sklearn/linear_
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale
    https://scikit-learn.org/stable/modules/preproces
Please also refer to the documentation for alternativ
    https://scikit-learn.org/stable/modules/linear_mo
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG,
LogisticRegression(random_state=0)
```



Predicting the target labels of testset

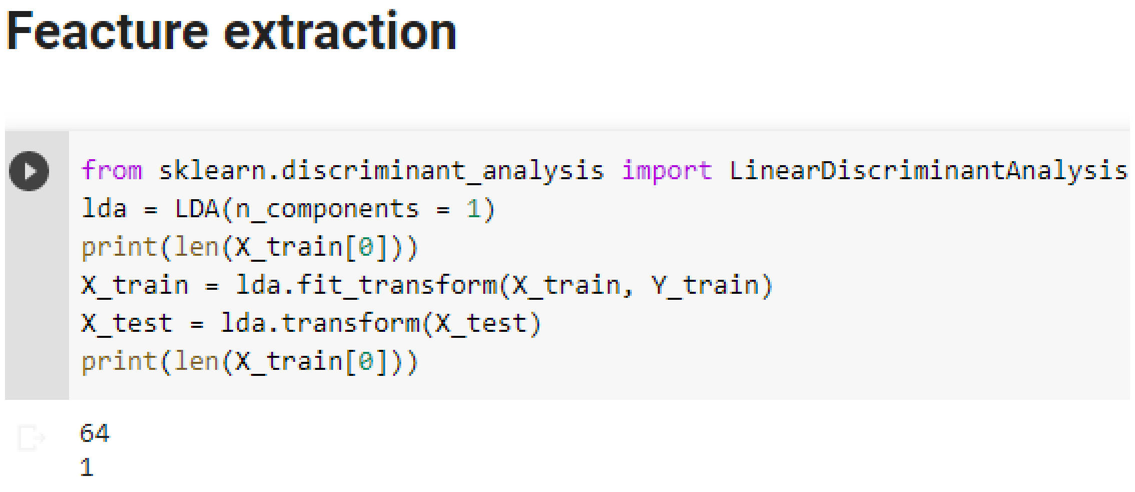
```
from sklearn.metrics import confusion_matrix, accuracy_score, precision_score,recall_s
from sklearn.metrics import plot_confusion_matrix
Y_pred = classifier.predict(X_test)
cm = confusion_matrix(Y_test, Y_pred)
plot_confusion_matrix(classifier, X_test, Y_test)
plt.show()
print("accuracy = %f" % accuracy_score(Y_test, Y_pred))
print("precision = %f" % precision_score(Y_test, Y_pred))
print("recall score = %f" % recall_score(Y_test, Y_pred))
print("f1 score = %f" % f1_score(Y_test, Y_pred))
#https://vitalflux.com/accuracy-precision-recall-f1-score-python-example/
```





```
accuracy = 0.935724
precision = 0.333333
recall score = 0.010638
f1 score = 0.020619
```

6. Feature extraction

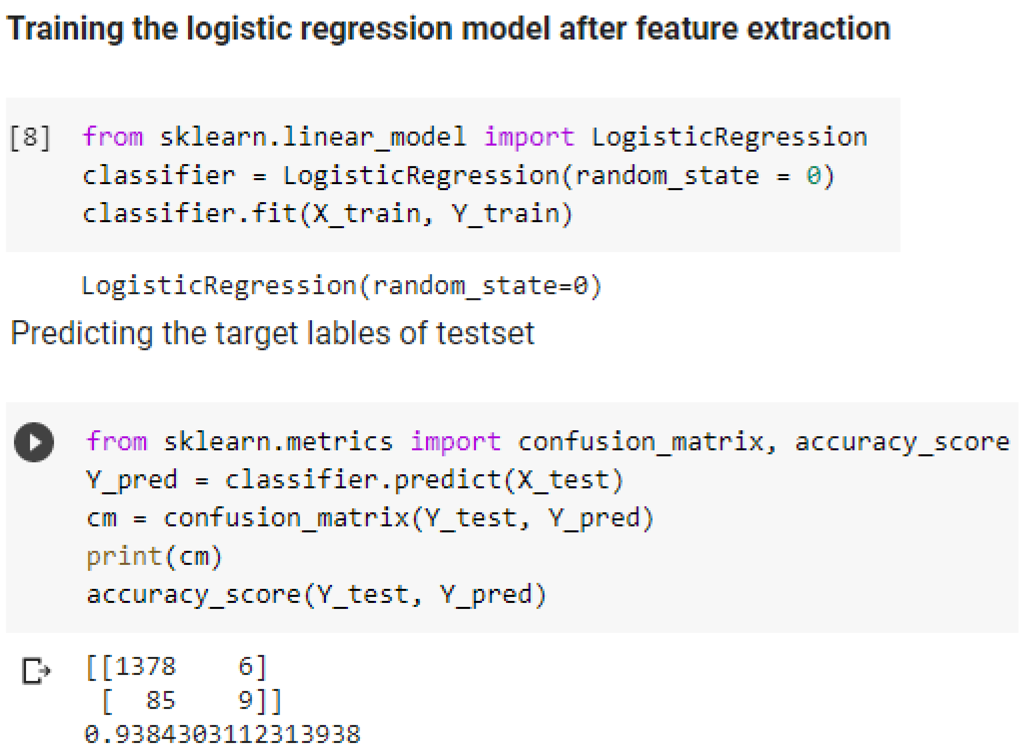


Feacture extraction

```
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
lda = LDA(n_components = 1)
print(len(X_train[0]))
X_train = lda.fit_transform(X_train, Y_train)
X_test = lda.transform(X_test)
print(len(X_train[0]))
```

```
64
1
```

7. Logistic regression after feature extraction



Training the logistic regression model after feature extraction

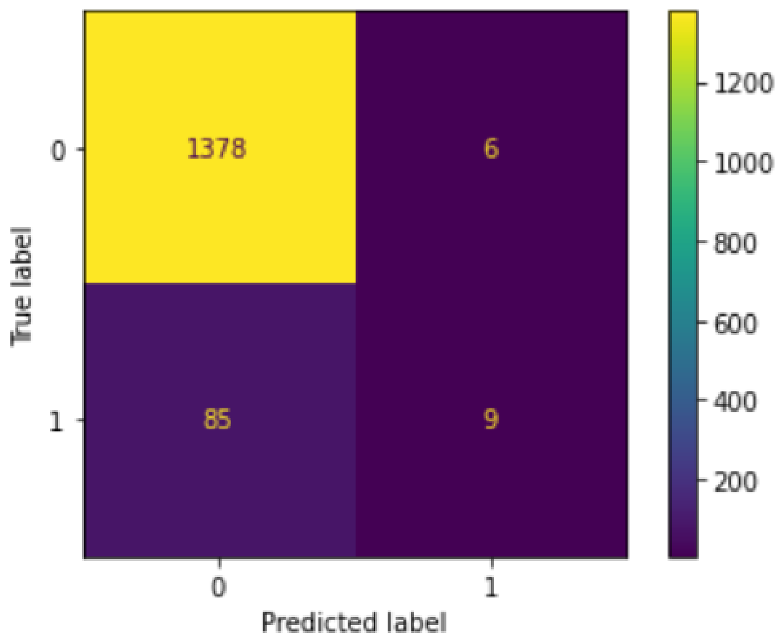
```
[8] from sklearn.linear_model import LogisticRegression
    classifier = LogisticRegression(random_state = 0)
    classifier.fit(X_train, Y_train)
```

```
LogisticRegression(random_state=0)
```

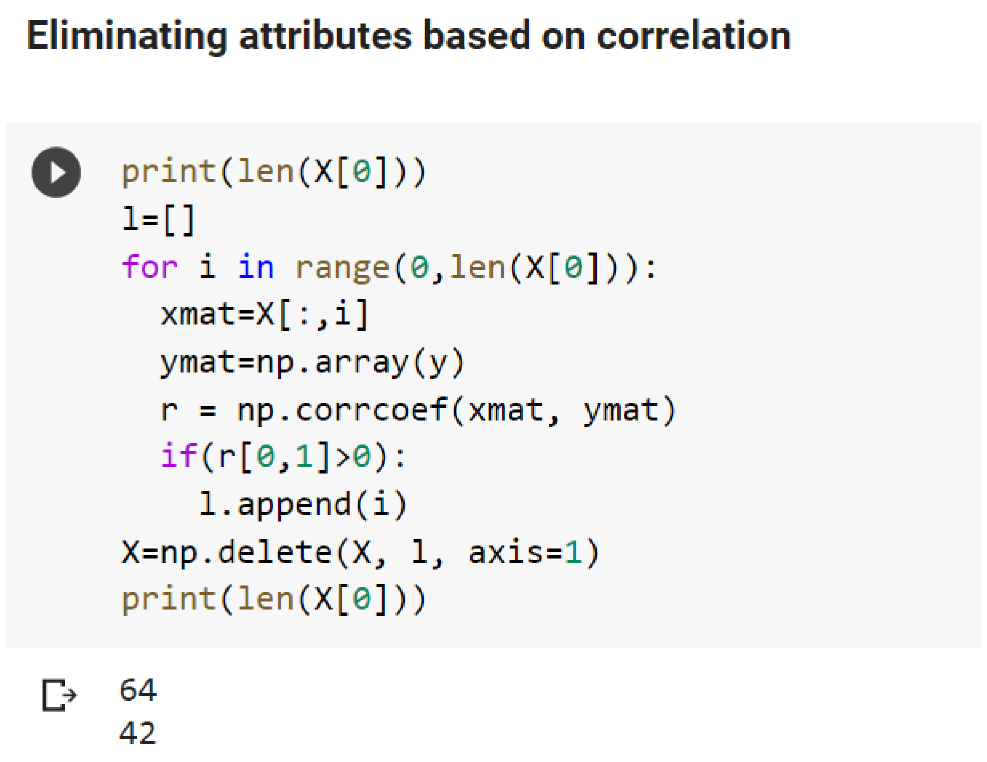
Predicting the target lables of testset

```
from sklearn.metrics import confusion_matrix, accuracy_score
Y_pred = classifier.predict(X_test)
cm = confusion_matrix(Y_test, Y_pred)
print(cm)
accuracy_score(Y_test, Y_pred)
```

```
[[1378    6]
 [  85    9]]
0.9384303112313938
```



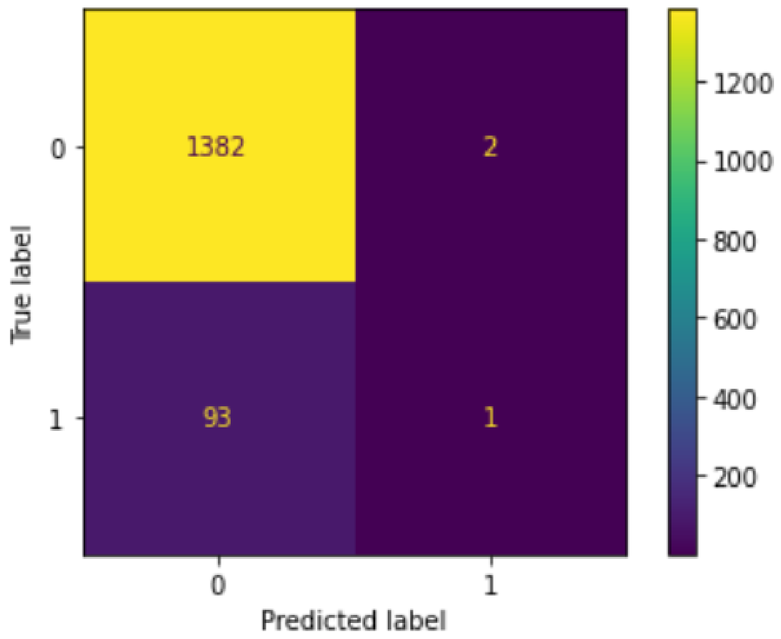
```
accuracy = 0.938430
precision = 0.600000
recall score = 0.095745
f1 score = 0.165138
```



Eliminating attributes based on correlation

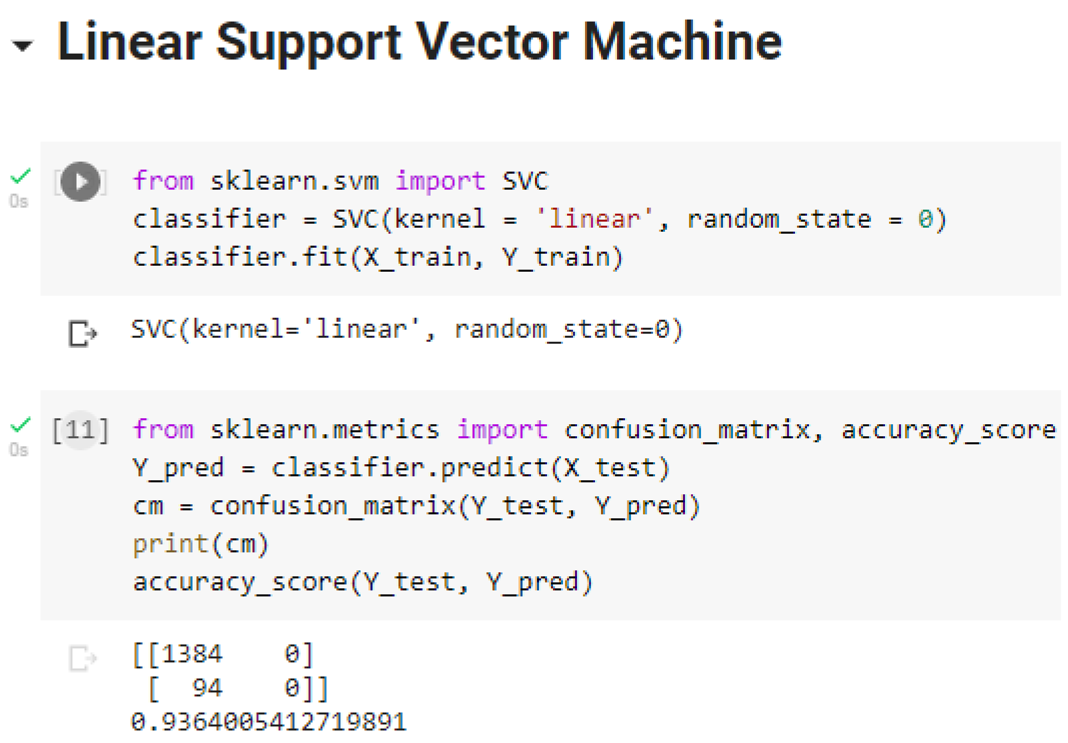
```
print(len(X[0]))
l=[]
for i in range(0,len(X[0])):
  xmat=X[:,i]
  ymat=np.array(y)
  r = np.corrcoef(xmat, ymat)
  if(r[0,1]>0):
    l.append(i)
X=np.delete(X, l, axis=1)
print(len(X[0]))
```

```
64
42
```



```
accuracy = 0.935724
precision = 0.333333
recall score = 0.010638
f1 score = 0.020619
```

8. Step 8



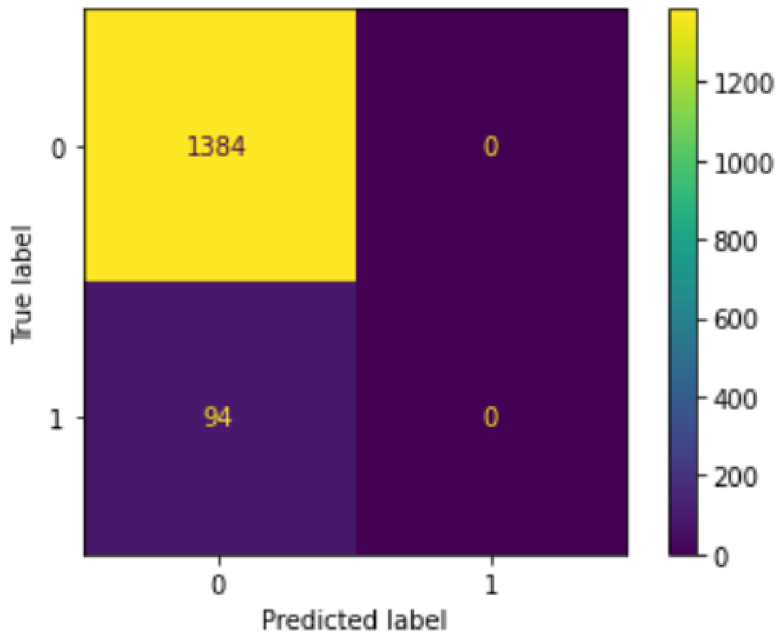
Linear Support Vector Machine

```
from sklearn.svm import SVC
classifier = SVC(kernel = 'linear', random_state = 0)
classifier.fit(X_train, Y_train)
```

```
SVC(kernel='linear', random_state=0)
```

```
from sklearn.metrics import confusion_matrix, accuracy_score
Y_pred = classifier.predict(X_test)
cm = confusion_matrix(Y_test, Y_pred)
print(cm)
accuracy_score(Y_test, Y_pred)
```

```
[[1384    0]
 [  94    0]]
0.9364005412719891
```



```
/usr/local/lib/python3.7/dist-packages/sklearn/
  _warn_prf(average, modifier, msg_start, len(r
accuracy = 0.936401
precision = 0.000000
recall score = 0.000000
f1 score = 0.000000
```

9. Step 9

Polynomial Support Vector Machine

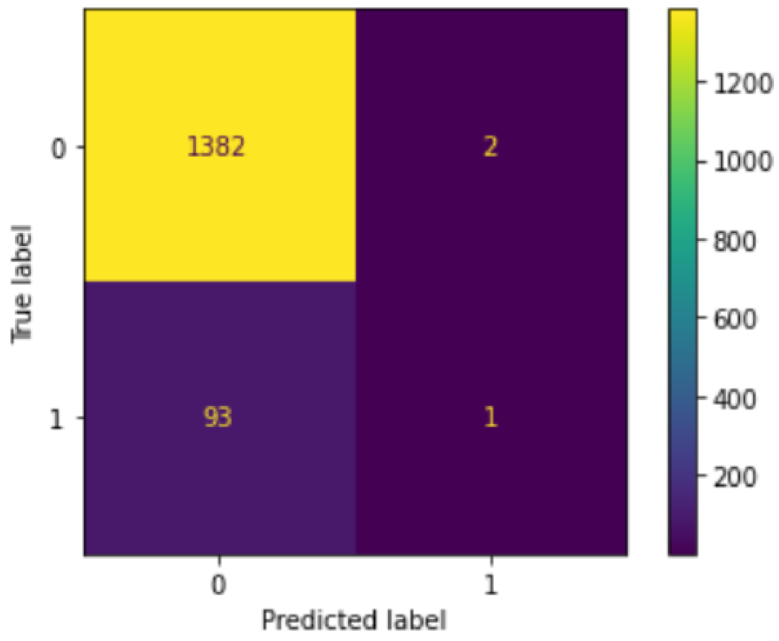
[12]

```
from sklearn.svm import SVC
classifier = SVC(kernel = 'poly',degree=3, random_state = 0)
classifier.fit(X_train, Y_train)
```

```
SVC(kernel='poly', random_state=0)
```

```
from sklearn.metrics import confusion_matrix, accuracy_score
Y_pred = classifier.predict(X_test)
cm = confusion_matrix(Y_test, Y_pred)
print(cm)
accuracy_score(Y_test, Y_pred)
```

```
[[1382    2]
 [  93    1]]
0.935723951285521
```



```
accuracy = 0.935724
precision = 0.333333
recall score = 0.010638
f1 score = 0.020619
```

10. Step 10

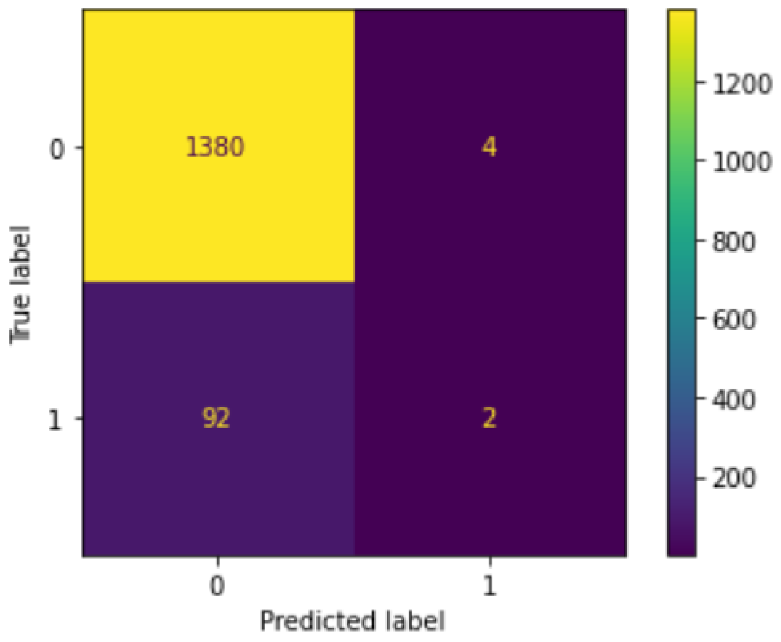
Radial basis function SVM

```
[14] from sklearn.svm import SVC
     classifier = SVC(kernel = 'rbf', random_state = 0)
     classifier.fit(X_train, Y_train)
```

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

```
from sklearn.metrics import confusion_matrix, accuracy_score
Y_pred = classifier.predict(X_test)
cm = confusion_matrix(Y_test, Y_pred)
print(cm)
accuracy_score(Y_test, Y_pred)
```

```
[[1380    4]
 [  92    2]]
0.9350473612990527
```



```
accuracy = 0.935047
precision = 0.333333
recall score = 0.021277
f1 score = 0.040000
```

11. Step 11

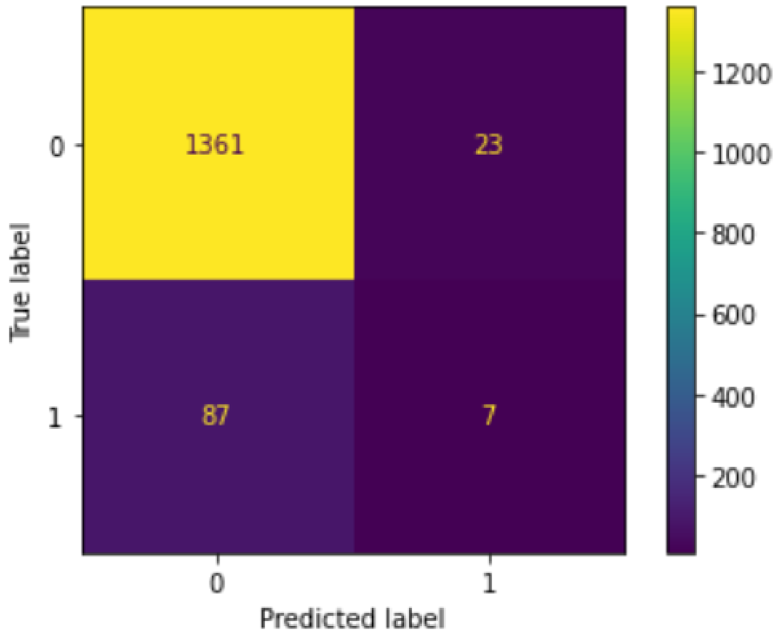
```
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(n_neighbors = 5,
classifier.fit(X_train, Y_train)
```

```
KNeighborsClassifier()
```

```
Classifier(n_neighbors = 5, metric = 'minkowski', p = 2)
 Y_train)
```

```
from sklearn.metrics import confusion_matrix, accuracy_score
Y_pred = classifier.predict(X_test)
cm = confusion_matrix(Y_test, Y_pred)
print(cm)
accuracy_score(Y_test, Y_pred)
```

```
[[1361   23]
 [  87    7]]
0.925575101488498
```



```
accuracy = 0.925575
precision = 0.233333
recall score = 0.074468
f1 score = 0.112903
```

5 Observations

Logistic regression is a supervised learning classification algorithm used to predict the probability of a target variable. We have initially trained the data-set on this model before applying any feature extraction and we got an accuracy score of 93.5723951285521 percent.

The confusion matrix seem to be biased towards False negatives i.e company is in-fact bankrupted but is predicted to be not, this is because the data-set is itself biased. We have huge number of companies which are not bankrupted(5500), having label 0. There are very few companies(410) having label 1. The data-set is inclined towards label 0.

LDA is supervised learning dimensionality reduction technique and aims to maximize the distance between the mean of each class and minimize the spreading within the class itself.This is applied on the training data to reduce the dimensionality of the matrix. Since the total number of objective classes are 2, we reduce the training set to a single column for each instance.
After feature extraction results of logistic regression have improved in terms of accuracy. The new accuracy was 93.84303112313938 percent.

6 Contributions

- Amruth P
  Selecting data-set from UCI machine learning Repository
  Extraction of required data
  Importing and prepossessing the data.
- Vineesh P
  Training Linear regression classification model on the data.
  Training Linear and Polynomial support vector machine on the data after feature extraction.
  Predicting classification labels for the above mentioned models.
- Varun KV
  Implementing feature extraction.
  Training Radial basis support vector machine and K-NN models on the data after feature extraction.
  Predicting classification labels for the above mentioned models.