## FINANCIAL DISTRESS PREDECTION

Abstract:

## This study has two objectives, namely, first aims to develop a discriminant model from the components of the cooperative assessment aspect in order to predict the occurrence of financial financial distress in legal entities in West Jakarta. Second, aims to find out the most dominant factors in predicting financial financial distress in cooperatives.This study uses as many samples as savings and loan cooperatives that provide financial statements to researchers for two years, from 2015 to 2016, both those which are still active and those who are not active. Data collection method is done by purposive sampling.

## This data set deals with the financial distress prediction for a sample of campanies

## First column:Company represents sample compaies

## Second column:Time shows different time periods that data belongs to Time series length varies between 1 to 14 for each company

## Third column:The target variable is denoted by “Finacial Distress” if it is greater than -0.50 the company should be considered as healthy (0).Otherwise,it would be regarded as financially distressed(1)

## Fourth column:the features denoted by x1 to x3,esome finaccial and non fincaial charcterstics of the sampled comapnys these fetures belong to the prwevious time perod which shoul be used to predict whether the company finaccaly distressed or not.

Methodology:

* Multilinear Regression

## Multiple linear regression (MLR), also known simply as multiple regression, is a statistical technique that uses several explanatory variables to predict the outcome of a response variable. The goal of multiple linear regression is to model the [linear relationship](https://www.investopedia.com/terms/l/linearrelationship.asp) between the explanatory (independent) variables and response (dependent) variables.

* RandomForest

## Random forest is aregression isasupervised learning algorthim and bagging techinique tha uises an ensemble learning method for regression in machine learing

## IMPORTING LIBARIRES

#### Import pandas as pd

#### import numpy as np

#### import matplotlib.pyplot as plt

#### from sklearn.model\_selection import train\_test\_split

#### from sklearn.linear\_model import LinearRegression

#### from sklearn.metrics import confusion\_matrix,accuracy\_score

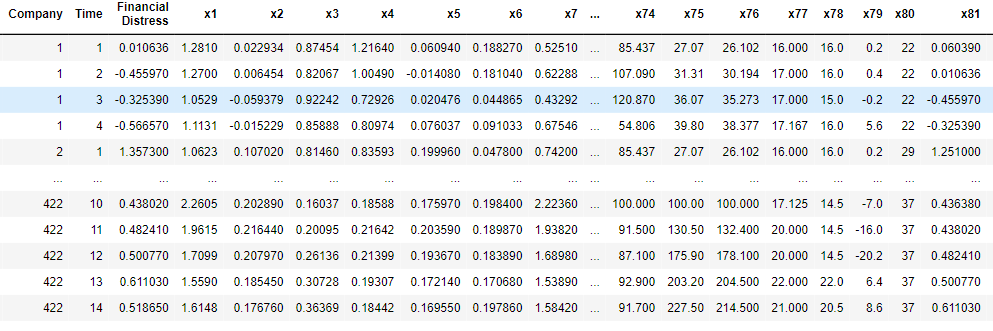
#### from sklearn.ensemble import RandomForestRegressor

## REDAING DATASET

#### df=pd.read\_csv('Financial Distress.csv.zip')

#### df

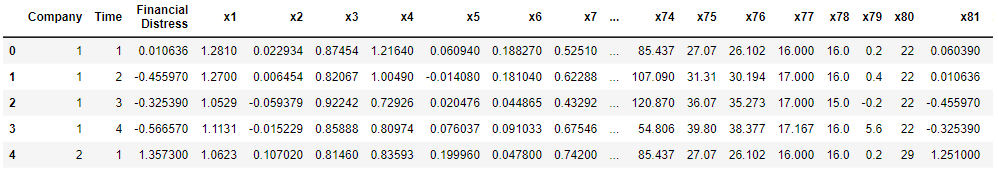
## OUTPUT



## TO CHECK FIRST FIVE COLUMNS

#### df.head()

## output



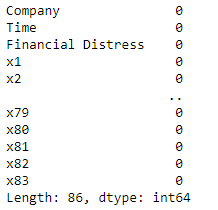
#### df.shape

## output

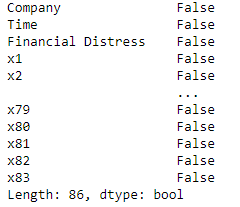


## CHECKING MISSING VALUE

#### df.isnull().sum()



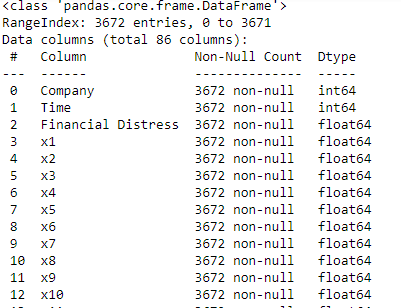
#### df.isnull().any()



## INFORMATION ABOUT DATASET

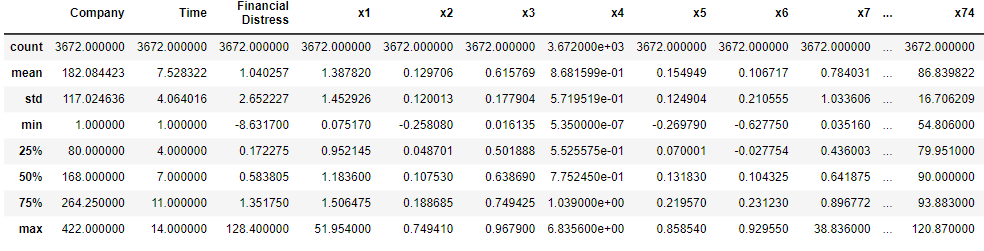
#### df.info()

## output:



#### df.describe()

## output:



#### print('lenghth of dataset:',len(df))

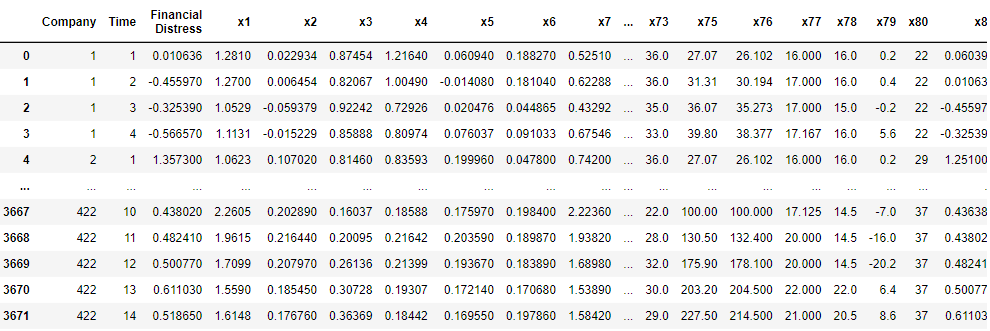


## SPLITTING THEDATASET

#### x=df.drop('x74',axis=1)

#### x

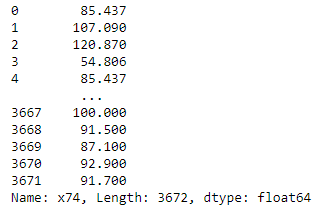
## output:



#### y=df['x74']

#### y

## output:



#### from sklearn.model\_selection import train\_test\_split

#### x\_train,x\_test,y\_train,y\_test=train\_test\_split(x,y,random\_state=0)

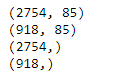
#### print(x\_train.shape)

#### print(x\_test.shape)

#### print(y\_train.shape)

#### print(y\_test.shape)

## output:



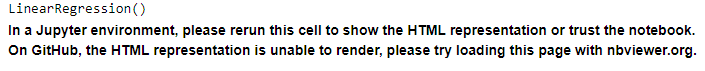
## MODEL TRAINING

#### from sklearn.linear\_model import LinearRegression

#### regressor=LinearRegression()

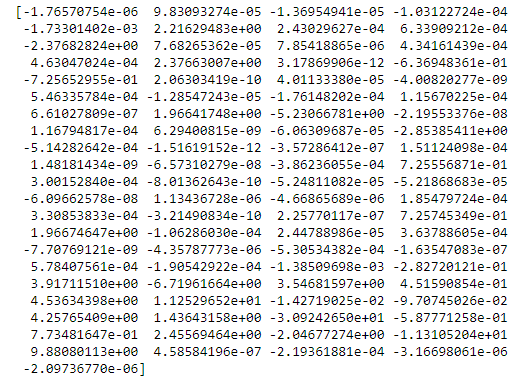
#### regressor.fit(x\_train,y\_train)

## output:



#### print(regressor.coef\_)

## output:



#### print(regressor.intercept\_)

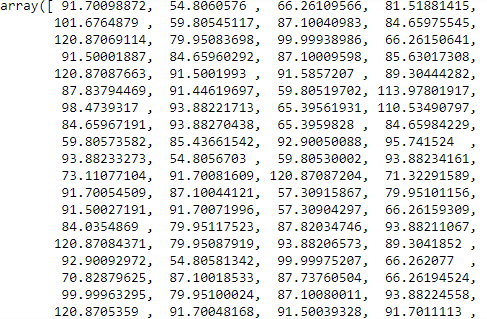
## output:

-46.268254777815955

#### PREDICTIONS

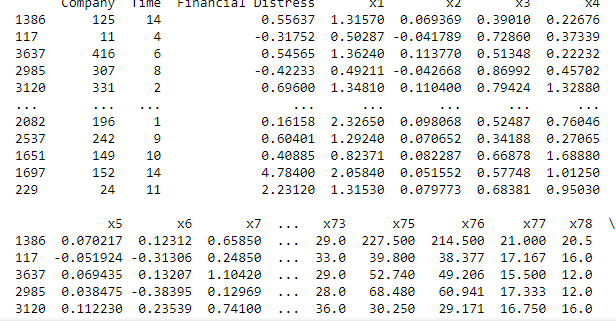
#### predicted=regressor.predict(x\_test)

#### predicted



#### print(x\_test)

## output:



## **Predicted shape**

#### output:

(918,)

#### dframe=pd.DataFrame(y\_test,predicted)

#### dframe

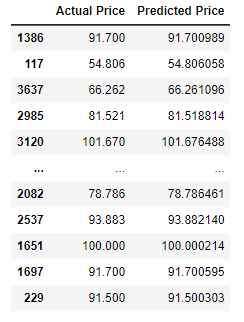
## output:

| **x74** |
| --- |
| **91.700989** | NaN |
| **54.806058** | NaN |
| **66.261096** | NaN |
| **81.518814** | NaN |
| **101.676488** | NaN |
| **...** | ... |
| **78.786461** | NaN |
| **93.882140** | NaN |
| **100.000214** | NaN |
| **91.700595** | NaN |
| **91.500303** | NaN |
|  |  |
|  |  |

#### TO CHECK THE ACTUAL PRICE,PREDICTED PRICE & DIFFERENCE dfr=pd.DataFrame({'Actual Price':y\_test,'Predicted Price':predicted})

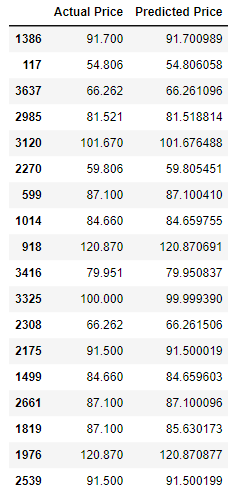
#### dfr

## output:



#### dfr.head(60)

## output:



#### EVALUATING THE MODEL

#### from sklearn.metrics import confusion\_matrix,accuracy\_score

#### train\_accuracy=regressor.score(x\_train,y\_train)

#### print('train\_accuracy:',train\_accuracy)

#### R\_test\_accuracy=regressor.score(x\_test,y\_test)

#### print('test\_accuracy:',R\_test\_accuracy)

#### output:



#### import math

#### from sklearn import metrics

#### print('Mean Absolute Error:',metrics.mean\_absolute\_error(y\_test,predicted))

#### print('Mean Squared Error:',metrics.mean\_squared\_error(y\_test,predicted))

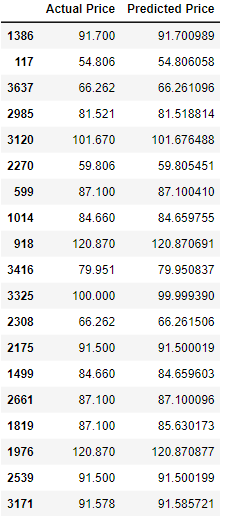
#### print('Root Mean Squared Error:',math.sqrt(metrics.mean\_squared\_error(y\_test,predicted)))

## output:

graph=dfr.head(20)

#### graph

## output:



## PLOTTING THE GRAPH FOR DIOFFERENCE BETWEEN ACTUAL PRICE AND PREDICTEDPRICE

#### graph.plot(kind='bar')

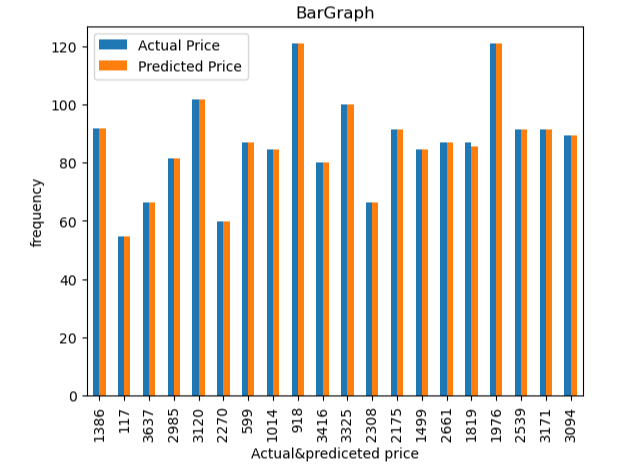
#### plt.title('BarGraph')

#### plt.xlabel('Actual&prediceted price')

#### plt.ylabel('frequency')

#### plt.show()

## output:



RESULT:In this case x-axis and y-axis represents the actual and predicted price in thisgraph ptredection price is almost sameas actual price

#### TRAINIG THE MODEL FOR RANDOM FORESTREGRESSOR

#### from sklearn.metrics import confusion\_matrix,accuracy\_scorein\_accuracy

#### x\_train,x\_test,y\_train,y\_test=train\_test\_split(x,y,test\_size=0.20,random\_state=100)

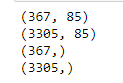
#### print(x\_train.shape)

#### print(x\_test.shape)

#### print(y\_train.shape)

#### print(y\_test.shape)

## output:



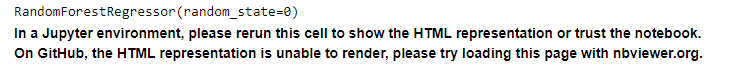
#### MODEL TRAINING

#### from sklearn.ensemble import RandomForestRegressor

#### regressor = RandomForestRegressor(n\_estimators=100,random\_state=0)

#### regressor.fit(x\_train,y\_train)

## output:



#### predicted=regressor.predict(x\_test)

#### predicted

## output:



EVALUATING THE MODEL

#### from sklearn.metrics import confusion\_matrix,accuracy\_score

#### train\_accuracy=regressor.score(x\_train,y\_train)

#### print('train\_accuracy:',train\_accuracy)

#### R\_test\_accuracy=regressor.score(x\_test,y\_test)

#### print('test\_accuracy:',R\_test\_accuracy)

## output:



#### COMPARISON BETWEEN LINEAR ANDRANDOM USING BARGRAPH

#### import matplotlib.pyplot as plt

#### linear\_regression\_accuracy =0.9999999825628604

#### random\_forest\_accuracy =0.992675587067571

#### accuracy\_scores = [linear\_regression\_accuracy, random\_forest\_accuracy]

#### model\_names = ['Linear Regression', 'Random Forest Regression']

#### plt.bar(model\_names, accuracy\_scores)

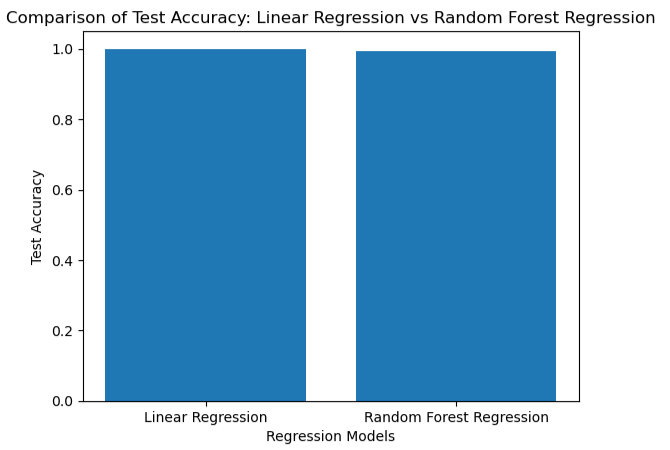
#### plt.xlabel('Regression Models')

#### plt.ylabel('Test Accuracy')

#### plt.title('Comparison of Test Accuracy: Linear Regression vs Random Forest Regression')

#### plt.show()

## output:



#### RESULT:

#### In this problem,x-axis represents the Regression modeles and y-axis represents the

#### Test accuracy.It iscomarasion between Linear Regression and Forest Regressor

#### Linear Regression accuracy:0.99

#### Random Forest Regressor Accuracy:0.92

#### In this model Linear regression has given best accuracy in the pictorial representations of graph is shown below

ANN MODEL

PREPROCESSING THE X\_TRAIN&X\_TEST

from sklearn.preprocessing import Standard

#### scaler=StandardScaler()

#### x\_train=scaler.fit\_transform(x\_train)

#### x\_test=scaler.fit\_transform(x\_test)

#### TRAIINING THE MODEL USING ANN MODEL

#### from sklearn.neural\_network import MLPRegressor

#### ann\_model=MLPRegressor(hidden\_layer\_sizes=(128,64,32),activation='relu',solver='lbfgs')

#### ann\_model.fit(x\_train,y\_train)

## output:

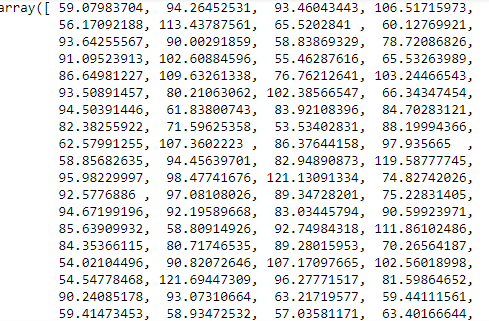


#### PREDICTION

#### y\_pred=ann\_model.predict(x\_test)

#### y\_pred

## output:



#### EVALUATING THE MODEL

#### train\_accuracy=ann\_model.score(x\_train,y\_train)

#### print('train\_accuracy(R\_squared):',train\_accuracy)

#### test\_accuracy=ann\_model.score(x\_test,y\_test)

#### print('test\_accuracy(R\_squared):',test\_accuracy)

#### print('Mean Absolute Error:',metrics.mean\_absolute\_error(y\_test,y\_pred))

#### print('Mean Squared Error:',metrics.mean\_squared\_error(y\_test,y\_pred))

#### print('Root Mean Squared Error:',math.sqrt(metrics.mean\_squared\_error(y\_test,y\_pred)))





COMPRESSION BETWEEN LINEAR REGRESSION RANDOMFOREST REGRESSION AND ANN

#### import matplotlib.pyplot as plt

#### linear\_regression\_accuracy =0.9999999825628604

#### random\_forest\_accuracy =0.9999999825628604

#### ANN=0.999754780100124

#### accuracy\_scores = [linear\_regression\_accuracy, random\_forest\_accuracy,ANN]

#### model\_names = ['Linear Regression', 'Random Forest Regression','ANN']

#### plt.bar(model\_names, accuracy\_scores)

#### plt.xlabel('Regression Models')

#### plt.ylabel('Test Accuracy')

#### plt.title('Comparison of Test Accuracy: Linear Regression vs Random Forest Regression vs ANN')

#### plt.show()

## output:

