A report on

**CREDIT SCORING SYSTEM**

Submitted for the fulfilment of Event 2 of the course

**LINEAR ALGEBRA(EC510)**

|  |  |
| --- | --- |
| **NAME** | **USN** |
| BHAGYANIDHI A PATIL | 01JST18EC016 |

**INDEX**

**CONTENTS PAGE NO.**

1. INTRODUCTION 3
2. LITERATURE REVIEW 4
3. METHOD 7
4. CODE 9
5. RESULTS 25
6. ADVANTAGES 27
7. DISADVANTAGES 28
8. APPLICATIONS 28

**CREDIT SCORING SYSTEM**

**1.INTRODUCTION**

A credit scoring model is the statistical tool widely used by lenders to access the credit worthiness of their potential and existing customers. **Credit score,** is a numeric value that represents the creditworthiness of an individual. All credit lending institutions like banks have complex credit models that use the information contained in the application like salary, credit commitments and past loan performances to determine a credit score of an application or an existing customer. The model outputs a score that represents how likely the lender will be repaid on time if they give a person a loan or a credit card.

A credit scorecard is one of such credit models, it is one of the most common credit models due to the fact it is relatively easy to interpret for customers and that it has been around for the last few decades, hence the development process is standard and widely understood.

However, it is worth to note that the range of score may be different from institution to institution, and the cut-off point for reject applications with lower score would vary from lender to lender and may even differ in the same lender but for different products.

In this model we will be using various demographic attributes and past repayment behaviour of an individual that can be utilized to predict his or her probability of default. In this model we have taken the dataset which contain 5962rows and 14columns.

These are the explanation of what each column represents:

1. BAD: 1 = applicant defaulted on loan or seriously delinquent; 0 = applicant paid loan
2. LOAN: Amount of the loan request
3. MORTDUE: Amount due on existing mortgage
4. VALUE: Value of current property REASON
5. DebtCon = debt consolidation;
6. Home Imp = home improvement
7. JOB: Occupational categories
8. YOJ: Years at present job
9. DEROG: Number of major derogatory reports
10. DELINQ: Number of delinquent credit lines
11. CLAGE: Age of oldest credit line in months
12. NINQ: Number of recent credit inquiries
13. CLNO: Number of credit lines
14. DEBTINC: Debt-to-income ratio

The binary variable BAD will be the target variable in our credit scoring model, while other variables will be used as predictors.

**2. LITERATURE REVIEW**

The need for controlling and effectively managing credit risk has led financial institutions to excel in improving techniques designed for this purpose, resulting in the development of various quantitative models by financial institutions and consulting companies. Hence, the growing number of academic studies about credit scoring shows a variety of classification methods applied to discriminate good and bad borrowers.

3. Three credit scoring models

3.1 Neural network

Neural networks have a strong ability to deal with

complicated problems by simulating the human brain, it

can be used to simulate the non-linear relationship in

complicated data. The feed-forward networks are the most

widely used architecture because they offer good

generalization abilities and are readily to implement. The

network architecture used in the paper is consists of three

layers of neurons connected by weights. The input of each

neuron is the weighted sum of the network inputs, and the

output of the neuron is a sigmoidal function value based

on its inputs. Given a finite number of pattern pairs

consisting of an input pattern

j

x and a target output

pattern

j

y , this network is trained by supervised

learning. Generally, the backpropagation algorithm,

which is the most popular learning algorithm, is adopted

to perform steepest descent on the total mean squared

error (

MSE

)

2

1

)

~

(

2

1

jj

n

t

yyMSE −=

3.1 Neural network

Neural networks have a strong ability to deal with

complicated problems by simulating the human brain, it

can be used to simulate the non-linear relationship in

complicated data. The feed-forward networks are the most

widely used architecture because they offer good

generalization abilities and are readily to implement. The

network architecture used in the paper is consists of three

layers of neurons connected by weights. The input of each

neuron is the weighted sum of the network inputs, and the

output of the neuron is a sigmoidal function value based

on its inputs. Given a finite number of pattern pairs

consisting of an input pattern

j

x and a target output

pattern

j

y , this network is trained by supervised

learning. Generally, the backpropagation algorithm,

which is the most popular learning algorithm, is adopted

to perform steepest descent on the total mean squared

error (

MSE

)

2

1

)

~

(

2

1

jj

n

t

yyMSE −=

There are different credit score models, which emphasize varying factors.

## **FICO Scoring Model**

The FICO scoring model is considered the most reliable because it has the best track record. It has been around since 1989 and there have been numerous revisions over the last three decades to take into account the changing factors that determine an accurate credit score.

The “classic” FICO scoring model gives consumers a number between 300 and 850. A score under 600 is considered poor. A score above 740 is considered excellent. In between is considered average to above average.

The latest scoring model is FICO 9 and it debuted in 2014. The major difference in the FICO 9 model is that it puts less weight on unpaid medical bills.

Why the change? The thinking behind FICO 9 indicated that unpaid medical debt was not necessarily an indicator of financial health. An individual could be waiting on insurance payments before paying the debt or they might not even know that a bill was sent to collections. In some cases, this factor could cause the credit score to rise by as much as 25 points.

FICO compares it to a consumer upgrading a computer operating system every time a new version of Windows is released. You may be satisfied with Windows XP or you may have upgraded to Windows 8 or 10.

The same thing happens with businesses and lenders who use the FICO score. Some lenders are still using FICO 5. Some have upgraded to FICO 9. The only way to know the FICO score meaning is to ask the lender you are dealing with.

## **Vantage Score Model**

The Vantage Score model was introduced in 2006 when the three major credit reporting bureaus — Experian, Equifax and TransUnion – decided to offer FICO some competition in the credit score business.

The Vantage Score model looks at familiar data — things like paying on time, keeping credit card balances low, avoiding new credit obligations, bank accounts and other assets — to calculate its score.

**High Weight: Payment History (40%) —** Whether you pay on time is the top predictor of risk. Remember that late payments are a negative that can appear on your credit report for seven years.

**Extreme Weight: Age and Type of Credit (21%) —** What’s the mix between your length of credit history and your account. Are you paying on a 30-year mortgage at the same time you’re paying on a 5-year car loan and monthly credit card bills? If you can handle all that – with on time payments! – you will do very well in this category.

**Extreme Weight: Credit Utilization (20%) —**It’s dividing your balances by your available credit. It’s recommended to keep your utilization under 30%.

**Medium Weight: Total Balances (11%) —** It’s your total debt (both current and delinquent. Similar to credit utilization, by lowering your debt, it gives you a higher chance of increasing your credit score.

**Low Weight: Recent Behaviour (5%) —** It deals with newly opened accounts and the number of hard inquiries. A high number is not a good sign for your credit report.

**Extremely Low Weight: Available Credit (3%) —** It’s the amount of credit you have available to use. Keep in mind that the Vantage Score model is used by Credit Karma, a service that provides your free credit score and report, along with credit monitoring and advice.

Vantage Score saw some scoring changes in 2017, mostly in the area of “trended data.’’ A person who is paying down debt is now likely to be scored better than a person who is making minimum payments and slowly accumulating credit card debt.

## **Other Credit Scoring Models**

Outside of the conventional and well-known outlets, there are several other credit scoring models.

**Trans Risk —** It’s based on data from TransUnion and determines an individual’s risk on new accounts, instead of existing accounts. Because of that specialized nature, there’s not much information available about the Trans Risk score. Accordingly, it isn’t utilized by many lenders. It has been reported that an individual’s Trans Risk score has generally been drastically lower than their FICO score.

**Experian’s National Equivalency Score —** It assigns users a score of 0-1,000 — with the typical criteria of payment history, credit length, credit mix, credit utilization, total balances and the number of inquiries — but Experian has never publicized the score’s criteria or weight. The scoring seems counterintuitive for consumers accustomed to the FICO system. In Experian’s system a score of 100 means a 10% chance that at least one account will become delinquent in the next 24 months, while a score of 900 means a 90% chance of that. There is an alternative scoring method of 360 to 840 (840 is good, 360 is bad, making it more compatible with the FICO model.

**Credit Xpert Credit Score —** It was developed to help businesses approve new account candidates. It inspects credit reports for ways to raise its score quickly or detect false information. By improving those scores, that should lead to more loan approval for customers.

**CE Credit Score —** The creator of this scoring model (CE Analytics) was unhappy with the current model of customers paying for their credit score and companies hiding how their credit scores were revealed. This is a free service, available at Quizzle, and it’s meant to create a free, transparent and accurate credit score. There’s scoring from 330 to 830. The model’s sister company, Quicken Loans, uses it in credit determinations.

**Insurance Score —**Here’s a fun fact: Insurance companies use an insurance credit score to determine your risk as a customer. Insurance scores range from 200-997 — generally, a good score is 770 or higher, while 500 or lower is considered poor — but it varies in different types of insurance. It’s wise to monitor your insurance score because that’s how your premiums are determined.

**3.METHOD**

**BLOCK DIAGRAMS FOR OUR CREDIT SCORING MODEL:**

DATA EXPLORATION

DATA SET

[Home Equity loans]

=

,

where

n is the total number of pattern pairs.

Given an initial weights and threshold, each input

pattern passes this network and gets an output pattern.

Then the error between the output pattern and the target

pattern is determined by

MSE

, and adjustment to

weights and thresholds. The process is repeated with each

pattern pair assigned for training network until the train

error is within a prescribed tolerance level.

=

,

where

n is the total number of pattern pairs.

Given an initial weights and threshold, each input

pattern passes this network and gets an output pattern.

Then the error between the output pattern and the target

pattern is determined by

MSE

, and adjustment to

weights and thresholds. The process is repeated with each

pattern pair assigned for training network until the train

error is within a prescribed tolerance level.

More detailed descriptions of neural

TRAINING AND TEST SET

COMPARING DIFFERENT MODELS

MODELS

Decision Tree

Logistic Regression

1.Pearson corr\_fact

2. chi-square test

3. f\_classif

4. f regression

1.Pearson corr\_fact

2. chi-square test

3. f\_classif

4. f regression

The target variable usually takes a binary form, depending on the data, it can be 0 for performing customers and 1 to indicate defaulted customers. In our model we take column BAD as target variable. This target variable is the one whose values are to be modelled and predicted by other variables.

**4.CODE**:

**Code for data exploration:**

importnumpyasnp

import matplotlib.pyplot as plt

import pandas as pd

import itertools

from sklearn.metrics import confusion\_matrix

df = pd.read\_csv(r"C:\Users\Desktop\hmeq.csv")

df\_i = pd.read\_csv(r"C:\Users\Desktop\hmeq.csv") #location of dataset in your PC

df.head() #returns first 5 rows of dataset

df.shape #returns tuple which contain no of rows n col

df.info() #returns summary of dataframe

df.columns

print(df["BAD"].value\_counts())

df["BAD"].value\_counts().plot("barh") #to plot graph

print(df["REASON"].value\_counts())

print(df["JOB"].value\_counts())

#Plotting histogram for various columns

f["LOAN"].plot.hist(bins = 20,figsize=(15,7.5)) #no of histogram bins

df["DEBTINC"].plot.hist(bins = 20,figsize=(15,5))

df["CLAGE"].plot.hist(bins = 20,figsize=(15,7.5))

df["CLNO"].plot.hist(bins = 20,figsize=(15,5))

df["VALUE"].plot.hist(bins = 80,figsize=(15,7.5))

df["MORTDUE"].plot.hist(bins = 40,figsize=(15,7.5))

df["YOJ"].plot.hist(bins = 40,figsize=(15,7.5))

df["DEROG"].value\_counts() #reurns object containg count of unique values

df["DELINQ"].value\_counts()

df["NINQ"].value\_counts()

#To remove/fill the undefined values from dataset we use fillna function ....

df.isnull().sum() #shows number of undefined value in dataset.

f["REASON"].fillna(value = "DebtCon",inplace = True) #fillna fills out missing values in the given series object

df["JOB"].fillna(value = "Other",inplace = True)

df["DEROG"].fillna(value=0,inplace=True)

df["DELINQ"].fillna(value=0,inplace=True)

df.fillna(value=df.mean(),inplace=True)

df.isnull().sum() #now all undefined values will be filled by mean or median value

df.head()

**Code for decision tree :**

importing the required modules

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

from sklearn.tree import DecisionTreeClassifier

dectree\_basic = DecisionTreeClassifier()

dectree\_basic.max\_depth = 100

# Training the basic Decision Tree model with training set

dectree\_basic.fit(x\_basic\_tr,y\_tr)

# Predicting the output of the test cases using the algorithm created above

y\_pre = dectree\_basic.predict(x\_basic\_te)

# Validating the algorithm using various Performance metrics

a2 = accuracy\_score(y\_te,y\_pre)

f2 = f1\_score(y\_te, y\_pre, average="macro")

p2 = precision\_score(y\_te, y\_pre, average="macro")

r2 = recall\_score(y\_te, y\_pre, average="macro")

print("accuracy score : ",a2)

print("f1 score : ",f2)

print("precision score : ",p2)

print("recall score : ",r2)

#CONFUSION MATRIX (We have defined plot in Logregression part)....

Computing Confusion matrix for the above algorithm

cnf\_matrix = confusion\_matrix(y\_te, y\_pre)

np.set\_printoptions(precision=2)

# Plot non-normalized confusion matrix

plt.figure()

plot\_confusion\_matrix(cnf\_matrix, classes=["BAD"],

title='Confusion matrix,Decision Tree Algorithm')

plt.show()

......

df.loc[df["CLAGE"]>=600,"CLAGE"] = 600 #attribute access a grp of rows n columns by label

df.loc[df["VALUE"]>=400000,"VALUE"] = 400000

df.loc[df["MORTDUE"]>=300000,"MORTDUE"] = 300000

df.loc[df["DEBTINC"]>=100,"DEBTINC"] = 100

df["B\_DEROG"] = (df["DEROG"]>=1)\*1

df["B\_DELINQ"] = (df["DELINQ"]>=1)\*1

df["JOB"].unique()

df["REASON\_1"] = (df["REASON"] == "HomeImp")\*1

df["REASON\_2"] = (df["REASON"] != "HomeImp")\*1

df["JOB\_1"] = (df["JOB"]=="Other")\*1

df["JOB\_2"] = (df["JOB"]=="Office")\*1

df["JOB\_3"] = (df["JOB"]=="Sales")\*1

df["JOB\_4"] = (df["JOB"]=="Mgr")\*1

df["JOB\_5"] = (df["JOB"]=="ProfExe")\*1

df["JOB\_6"] = (df["JOB"]=="Self")\*1

df.drop(["JOB","REASON"],axis = 1,inplace = True)

df["YOJ"] = df["YOJ"].apply(lambda t : np.log(t+1))

df.head()

**Code for logistic regression:**

# importing the required modules

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

from sklearn.linear\_model import LogisticRegression

# removing the features BAD,JOB,REASON from the input features set

x\_basic = df.drop(columns=["BAD","JOB","REASON"])

y = df["BAD"]

# Spliting the data into test and train sets

x\_basic\_tr,x\_basic\_te,y\_tr,y\_te = train\_test\_split(x\_basic,y,test\_size =.33,random\_state=1)

logreg\_basic = LogisticRegression()

# Training the basic logistic regression model with training set

logreg\_basic.fit(x\_basic\_tr,y\_tr\_x

# Printing the coefficients

print("intercept ")

print(logreg\_basic.intercept\_)

print("")

print("coefficients ")

print(logreg\_basic.coef\_)

# Predicting the output of the test cases using the algorithm created above

y\_pre = logreg\_basic.predict(x\_basic\_te)

# Validating the algorithm using various Performance metrics

from sklearn.metrics import accuracy\_score, f1\_score, precision\_score, recall\_score

print("")

a1 = accuracy\_score(y\_te,y\_pre)

f1 = f1\_score(y\_te, y\_pre, average="macro")

p1 = precision\_score(y\_te, y\_pre, average="macro")

r1 = recall\_score(y\_te, y\_pre, average="macro")

print("accuracy score : ",a1)

print("f1 score : ",f1)

print("precision score : ",p1)

print("recall score : ",r1)

#PLOTING CONFUSION MATRIX.....

def plot\_confusion\_matrix(cm, classes,

normalize=False,

title='Confusion matrix',

cmap=plt.cm.Blues):

"""

This function prints and plots the confusion matrix.

Normalization can be applied by setting `normalize=True`.

"""

if normalize:

cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]

print("Normalized confusion matrix")

else:

print('Confusion matrix, without normalization')

print(cm)

plt.imshow(cm, interpolation='nearest', cmap=cmap)

plt.title(title)

plt.colorbar()

tick\_marks = np.arange(len(classes))

plt.xticks(tick\_marks, classes, rotation=45)

plt.yticks(tick\_marks, classes)

fmt = '.2f' if normalize else 'd'

thresh = cm.max() / 2.

for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):

plt.text(j, i, format(cm[i, j], fmt),

horizontalalignment="center",

color="white" if cm[i, j] > thresh else "black")

plt.tight\_layout()

plt.ylabel('True label')

plt.xlabel('Predicted label')

# Computing Confusion matrix for the above algorithm

cnf\_matrix = confusion\_matrix(y\_te, y\_pre)

np.set\_printoptions(precision=2)

# Plot non-normalized confusion matrix

plt.figure()

plot\_confusion\_matrix(cnf\_matrix, classes=["BAD"],

title='Confusion matrix - Logistic Regression Algorithm')

plt.show()

Code for Pearson coefficient:

df.corr(method='pearson')

feat1=["DEROG","DELINQ","CLAGE","NINQ","DEBTINC","YOJ","LOAN"]

#feat2=["DEROG","DELINQ","CLAGE","NINQ","DEBTINC","LOAN","JOB\_2","YOJ","JOB\_3","MORTDUE"]

x = df[feat1] #pearson corr method

y = df["BAD"]

x\_tr,x\_te,y\_tr,y\_te = train\_test\_split(x,y,test\_size = 0.33,random\_state=1)

logreg = LogisticRegression()

logreg.fit(x\_tr,y\_tr) #training the model

y\_pre = logreg.predict(x\_te)

a3 = accuracy\_score(y\_te,y\_pre)

f3 = f1\_score(y\_te, y\_pre, average="macro")

p3 = precision\_score(y\_te, y\_pre, average="macro")

r3 = recall\_score(y\_te, y\_pre, average="macro")

print("accuracy score : ",a3)

print("f1 score : ",f3)

print("precision score : ",p3)

print("recall score : ",r3)

#Computing Confusion matrix for the above algorithm

cnf\_matrix = confusion\_matrix(y\_te, y\_pre)

np.set\_printoptions(precision=2)

# Plot non-normalized confusion matrix

plt.figure()

plot\_confusion\_matrix(cnf\_matrix, classes=["BAD"],

title='Confusion matrix - Logistic Regression Algorithm with pearson corr\_f')

plt.show()

**# Decision Tree classifier using feat1**

clf\_tree=DecisionTreeClassifier()

clf\_tree.max\_depth = 100

clf\_tree.fit(x\_tr,y\_tr)

y\_pre = clf\_tree.predict(x\_te)

a4 = accuracy\_score(y\_te,y\_pre)

f4 = f1\_score(y\_te, y\_pre, average="macro")

p4 = precision\_score(y\_te, y\_pre, average="macro")

r4 = recall\_score(y\_te, y\_pre, average="macro")

print("accuracy score : ",a4)

print("f1 score : ",f4)

print("precision score : ",p4)

print("recall score : ",r4)

print("")

**# Computing Confusion matrix for the above algorithm**

cnf\_matrix = confusion\_matrix(y\_te, y\_pre)

np.set\_printoptions(precision=2)

# Plot non-normalized confusion matrix

plt.figure()

plot\_confusion\_matrix(cnf\_matrix, classes=["BAD"],

title='Confusion matrix - Decision Tree Algorithm using pearson corr\_f')

plt.show()

**Code for Chi-square test:**

# Finding the best 10 features using chi2 test

from sklearn.feature\_selection import chi2

from sklearn.feature\_selection import SelectKBest

df\_new = pd.DataFrame(SelectKBest(chi2, k=10).fit\_transform(df.drop(["BAD"],axis = 1),df["BAD"]))

df\_new.head(

# Running the logistic regression algorithm using the features selected from chi2 test

x = df\_new

y = df["BAD"]

x\_tr,x\_te,y\_tr,y\_te = train\_test\_split(x,y,test\_size = .33,random\_state=1)

logreg = LogisticRegression()

logreg.fit(x\_tr,y\_tr)

y\_pre = logreg.predict(x\_te)

y\_pre = logreg.predict(x\_te)

a5 = accuracy\_score(y\_te,y\_pre)

f5 = f1\_score(y\_te, y\_pre, average="macro")

p5 = precision\_score(y\_te, y\_pre, average="macro")

r5 = recall\_score(y\_te, y\_pre, average="macro")

print("accuracy score : ",a5)

print("f1 score : ",f5)

print("precision score : ",p5)

print("recall score : ",r5)

# Computing Confusion matrix for the above algorithm

cnf\_matrix = confusion\_matrix(y\_te, y\_pre)

np.set\_printoptions(precision=2)

# Plot non-normalized confusion matrix

plt.figure()

plot\_confusion\_matrix(cnf\_matrix, classes=["BAD"],

title='Confusion matrix - Logistic Regression Algorithm with chi2 test')

plt.show()

# Decision Tree classifier using features from chi2 test

clf\_tree=DecisionTreeClassifier()

clf\_tree.max\_depth = 100

clf\_tree.fit(x\_tr,y\_tr)

y\_pre = clf\_tree.predict(x\_te)

a6 = accuracy\_score(y\_te,y\_pre)

f6 = f1\_score(y\_te, y\_pre, average="macro")

p6 = precision\_score(y\_te, y\_pre, average="macro")

r6 = recall\_score(y\_te, y\_pre, average="macro")

print("accuracy score : ",a6)

print("f1 score : ",f6)

print("precision score : ",p6)

print("recall score : ",r6)

# Computing Confusion matrix for the above algorithm

cnf\_matrix = confusion\_matrix(y\_te, y\_pre)

np.set\_printoptions(precision=2)

# Plot non-normalized confusion matrix

plt.figure()

plot\_confusion\_matrix(cnf\_matrix, classes=["BAD"],

title='Confusion matrix - Decision Tree Algorithm using chi2 test for feature selection')

plt.show()

df.head()

**Code for f\_classif:**

from sklearn.feature\_selection import f\_classif

df\_new2 = pd.DataFrame(SelectKBest(f\_classif, k=10).fit\_transform(df.drop(["BAD"],axis=1),df["BAD"]))

df\_new2.head()

# Running the logistic regression algorithm using the features selected from f\_classif test

x = df\_new2

y = df["BAD"]

x\_tr,x\_te,y\_tr,y\_te = train\_test\_split(x,y,test\_size = .33,random\_state=1)

logreg = LogisticRegression()

logreg.fit(x\_tr,y\_tr)

y\_pre = logreg.predict(x\_te)

a7 = accuracy\_score(y\_te,y\_pre)

f7 = f1\_score(y\_te, y\_pre, average="macro")

p7 = precision\_score(y\_te, y\_pre, average="macro")

r7 = recall\_score(y\_te, y\_pre, average="macro")

print("accuracy score : ",a7)

print("f1 score : ",f7)

print("precision score : ",p7)

print("recall score : ",r7)

# Computing Confusion matrix for the above algorithm

cnf\_matrix = confusion\_matrix(y\_te, y\_pre)

np.set\_printoptions(precision=2)

# Plot non-normalized confusion matrix

plt.figure()

plot\_confusion\_matrix(cnf\_matrix, classes=["BAD"],

title='Confusion matrix - Logistic Regression Algorithm with f\_classif')

plt.show()

# Decision Tree classifier using features from f\_classif test

clf\_tree=DecisionTreeClassifier()

clf\_tree.max\_depth = 100

clf\_tree.fit(x\_tr,y\_tr)

y\_pre = clf\_tree.predict(x\_te)

a8 = accuracy\_score(y\_te,y\_pre)

f8 = f1\_score(y\_te, y\_pre, average="macro")

p8 = precision\_score(y\_te, y\_pre, average="macro")

r8 = recall\_score(y\_te, y\_pre, average="macro")

print("accuracy score : ",a8)

print("f1 score : ",f8)

print("precision score : ",p8)

print("recall score : ",r8)

# Computing Confusion matrix for the above algorithm

cnf\_matrix = confusion\_matrix(y\_te, y\_pre)

np.set\_printoptions(precision=2)

# Plot non-normalized confusion matrix

plt.figure()

plot\_confusion\_matrix(cnf\_matrix, classes=["BAD"],

title='Confusion matrix - Decision Tree Algorithm using f\_classif feature selector')

plt.show()

Code for f\_regression:

from sklearn import tree

import graphviz

dot\_dat = tree.export\_graphviz(clf\_tree, out\_file=None)

graph = graphviz.Source(dot\_dat)

graph

from sklearn.feature\_selection import f\_regression

df\_new3 = pd.DataFrame(SelectKBest(f\_regression, k=10).fit\_transform(df.drop(["BAD"],axis=1),df["BAD"]))

df\_new3.head()

# Running the logistic regression algorithm using the features selected from f\_regression test

x = df\_new3

y = df["BAD"]

x\_tr,x\_te,y\_tr,y\_te = train\_test\_split(x,y,test\_size = .33,random\_state=1)

logreg = LogisticRegression()

logreg.fit(x\_tr,y\_tr)

y\_pre2 = logreg.predict(x\_te)

a9 = accuracy\_score(y\_te,y\_pre2)

f9 = f1\_score(y\_te, y\_pre2, average="macro")

p9 = precision\_score(y\_te, y\_pre2, average="macro")

r9 = recall\_score(y\_te, y\_pre2, average="macro")

print("accuracy score : ",a9)

print("f1 score : ",f9)

print("precision score : ",p9)

print("recall score : ",r9)

# Computing Confusion matrix for the above algorithm

cnf\_matrix = confusion\_matrix(y\_te, y\_pre)

np.set\_printoptions(precision=2)

# Plot non-normalized confusion matrix

plt.figure()

plot\_confusion\_matrix(cnf\_matrix, classes=["BAD"],

title='Confusion matrix - Logistic Regression Algorithm with f\_regression')

plt.show()

# Decision Tree classifier using features from f\_regression test

clf\_tree=DecisionTreeClassifier()

clf\_tree.max\_depth = 100

clf\_tree.fit(x\_tr,y\_tr)

y\_pre = clf\_tree.predict(x\_te)

a10 = accuracy\_score(y\_te,y\_pre)

f10 = f1\_score(y\_te, y\_pre, average="macro")

p10= precision\_score(y\_te, y\_pre, average="macro")

r10 = recall\_score(y\_te, y\_pre, average="macro")

print("accuracy score : ",a10)

print("f1 score : ",f10)

print("precision score : ",p10)

print("recall score : ",r10)

# Computing Confusion matrix for the above algorithm

cnf\_matrix = confusion\_matrix(y\_te, y\_pre)

np.set\_printoptions(precision=2)

# Plot non-normalized confusion matrix

plt.figure()

plot\_confusion\_matrix(cnf\_matrix, classes=["BAD"],

title='Confusion matrix - Decision Tree Algorithm using f\_regression feature selector')

plt.show()

**Code for comparing methods:**

models = pd.DataFrame({

'Model': ['Logistic Regression', 'Decision Tree','Logistic Regression', 'Decision Tree','Logistic Regression', 'Decision Tree','Logistic Regression', 'Decision Tree','Logistic Regression', 'Decision Tree'],

'Feature Selection Method' : ['None','None','Pearson corr\_fact','Pearson corr\_fact','chi2 test','chi2 test','f\_classif','f\_classif','f\_regression','f\_regression'],

'Accuracy Score': [a1,a2,a3,a4,a5,a6,a7,a8,a9,a10],

'Recall Score' : [r1,r2,r3,r4,r5,r6,r7,r8,r9,r10],

'F1 Score' : [f1,f2,f3,f4,f5,f6,f7,f8,f9,f10],

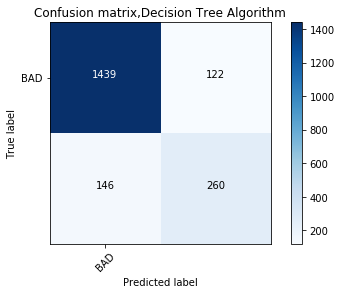
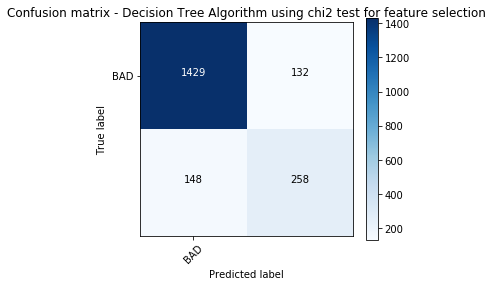
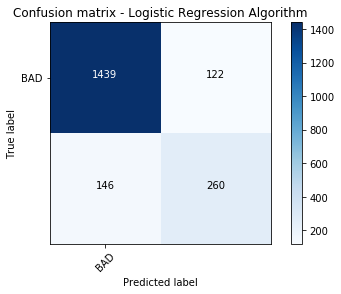
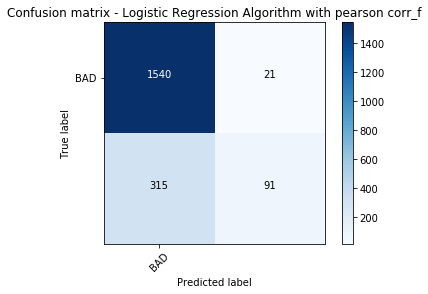
'Precision Score' : [p1,p2,p3,p4,p5,p6,p7,p8,p9,p10]

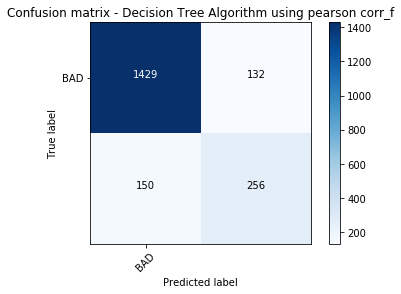
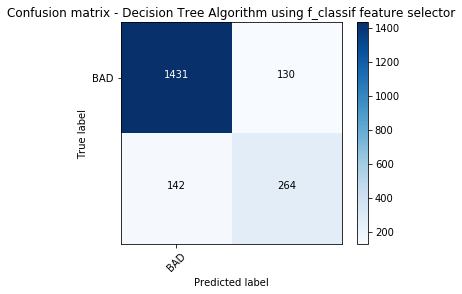
})

models

pd.pivot\_table(models,index = ["Feature Selection Method","Model"])

**5.RESULTS**

**  **

****

|  |  | **Accuracy Score** | **F1 Score** | **Precision Score** | **Recall Score** |
| --- | --- | --- | --- | --- | --- |
| **Feature Selection Method** | **Model** |  |  |  |  |
| **None** | **Decision Tree** | 0.863752 | 0.787355 | 0.794257 | 0.781120 |
| **Logistic Regression** | 0.792578 | 0.444550 | 0.521842 | 0.500271 |
| **Pearson corr\_fact** | **Decision Tree** | 0.856634 | 0.777514 | 0.782398 | 0.772990 |
| **Logistic Regression** | 0.829181 | 0.626495 | 0.821344 | 0.605343 |
| **chi2 test** | **Decision Tree** | 0.857651 | 0.779506 | 0.783845 | 0.775453 |
| **Logistic Regression** | 0.825623 | 0.612992 | 0.816271 | 0.595811 |
| **f\_classif** | **Decision Tree** | 0.861718 | 0.786605 | 0.789889 | 0.783483 |
| **Logistic Regression** | 0.826640 | 0.647622 | 0.770896 | 0.622877 |
| **f\_regression** | **Decision Tree** | 0.860702 | 0.787010 | 0.787537 | 0.786487 |
| **Logistic Regression** | 0.826640 | 0.647622 | 0.770896 | 0.622877 |

|  | **Model** | **Feature Selection Method** | **Accuracy Score** | **Recall Score** | **F1 Score** | **Precision Score** |
| --- | --- | --- | --- | --- | --- | --- |
| **0** | Logistic Regression | None | 0.792578 | 0.500271 | 0.444550 | 0.521842 |
| **1** | Decision Tree | None | 0.863752 | 0.781120 | 0.787355 | 0.794257 |
| **2** | Logistic Regression | Pearson corr\_fact | 0.829181 | 0.605343 | 0.626495 | 0.821344 |
| **3** | Decision Tree | Pearson corr\_fact | 0.856634 | 0.772990 | 0.777514 | 0.782398 |
| **4** | Logistic Regression | chi2 test | 0.825623 | 0.595811 | 0.612992 | 0.816271 |
| **5** | Decision Tree | chi2 test | 0.857651 | 0.775453 | 0.779506 | 0.783845 |
| **6** | Logistic Regression | f\_classif | 0.826640 | 0.622877 | 0.647622 | 0.770896 |
| **7** | Decision Tree | f\_classif | 0.861718 | 0.783483 | 0.786605 | 0.789889 |
| **8** | Logistic Regression | f\_regression | 0.826640 | 0.622877 | 0.647622 | 0.770896 |
| **9** | Decision Tree | f\_regression | 0.860702 | 0.786487 | 0.787010 | 0.787537 |

**6.ADVANTAGES**

* Faster decisions.
* More consistent decisions.
* Better decisions resulting in increased sales and profits.
* Increased productivity in the credit department.
* The ability to identify customers that can be offered higher credit limits at relatively low risk to the company.
* Flexibility built into the model so that the credit manager can alter certain parameters and increase or decrease the amount of credit risk the scoring model will consider acceptable, or unacceptable.

**7.DISADVANTAGES**

* A properly designed credit scoring system allows creditors to evaluate thousands of applications consistently, impartially and quickly. If this is true, then the opposite must also be true. A poorly designed credit scoring system can evaluate thousands of applicants and can make the wrong recommendation every time.
* Credit risk can never be measured precisely, and any model that says it can is wrong.
* Credit risk can change almost overnight. Example: The owner of a business dies and there is no one qualified to replace him.
* Credit managers should be able to override the credit score and its credit recommendation. However, doing so is difficult to justify if there is a serious payment problem, or worse a bad debt loss. For this reason, credit professionals are reticent to override the scoring model even when they believe the "recommendation" is wrong.
* Internally developed credit scoring models often lack sophistication and usually have not been subjected to critical analysis of the statistical significance of the factors used to develop a credit score and a credit recommendation...but
* Professionally designed, tested and validated credit scoring models can be expensive, and can be hard to customize. As a result, the credit scores these programs generate may not mimic the decision-making style and the risk tolerance of the companies that purchase them - and therefore they do not produce the desired results.
* Some professionally designed models only provide the credit manager with a numerical score. With this limited information, it can be quite difficult for the credit manager to explain a negative credit decision [based only on a numerical score] to an irate credit applicant, or to an active customer.

**8.APPLICATIONS**

* Finance firms
* Banks
* Any other money lenders