```
#1.CHOOSING A DATASET :  MFG10 Year Termination Data.csv
#-------------------------------------------------------
# CHOOSING A ALGORITHAM : Decision Tree,Random
Forest,LogisticRegression,NaiveBayes, KNeighbors

# project work flow
#====================
# 1. choose a data set (i choose NETWORK prediction classification
data)
# 2. import all the  necessary libraries
# 3. load the data set using pandas module
# 4. feature selection (x,y) and scaling data (standard scalar),split
the data
# 5. model creation by invoking algorithm
# 6. model training by fitting (x train & y train) data
# 7. model prediction (ypre)- 'using xtest'/
# 8. calculate perfomace accuracy using output metrics

#2.IMPORTING ALL NECESSARY LIBRARIES
#----------------------------------

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score, confusion_matrix,
classification_report
from sklearn.tree import DecisionTreeClassifier
from sklearn.naive_bayes import GaussianNB,MultinomialNB
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier


#3.LOADING THE DATASET USING PANDAS MODULE
#------------------------------------------

data=pd.read_csv(r"C:\Users\ELCOT\Desktop\ysqure technology\
MFG10YearTerminationData (1).csv")
data[:5]
```

```
   EmployeeID  age  length_of_service  store_name  STATUS_YEAR  \
0        1318   52                 17          35         2006
1        1318   53                 18          35         2007
2        1318   54                 19          35         2008
3        1318   55                 20          35         2009
4        1318   56                 21          35         2010
```

```
    recorddate_key orighiredate_key  city_name department_name
job_title  \
0  12/31/2006 0:00        8/28/1989  Vancouver       Executive
CEO
1  12/31/2007 0:00        8/28/1989  Vancouver       Executive
CEO
2  12/31/2008 0:00        8/28/1989  Vancouver       Executive
CEO
3  12/31/2009 0:00        8/28/1989  Vancouver       Executive
CEO
4  12/31/2010 0:00        8/28/1989  Vancouver       Executive
CEO

  birthdate_key gender_short gender_full termreason_desc
termtype_desc  \
0    01-03-1954            M        Male  Not Applicable  Not
Applicable
1    01-03-1954            M        Male  Not Applicable  Not
Applicable
2    01-03-1954            M        Male  Not Applicable  Not
Applicable
3    01-03-1954            M        Male  Not Applicable  Not
Applicable
4    01-03-1954            M        Male  Not Applicable  Not
Applicable

  terminationdate_key  STATUS  BUSINESS_UNIT
0          01-01-1900  ACTIVE              0
1          01-01-1900  ACTIVE              0
2          01-01-1900  ACTIVE              0
3          01-01-1900  ACTIVE              0
4          01-01-1900  ACTIVE              0
```

```
data.info()
print('')
data.shape
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 49653 entries, 0 to 49652
Data columns (total 18 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   EmployeeID           49653 non-null  int64
 1   age                  49653 non-null  int64
 2   length_of_service    49653 non-null  int64
 3   store_name           49653 non-null  int64
 4   STATUS_YEAR          49653 non-null  int64
 5   recorddate_key       49653 non-null  object
 6   orighiredate_key     49653 non-null  object
 7   city_name            49653 non-null  object
 8   department_name      49653 non-null  object
 9   job_title            49653 non-null  object
 10  birthdate_key        49653 non-null  object
 11  gender_short         49653 non-null  object
 12  gender_full          49653 non-null  object
 13  termreason_desc      49653 non-null  object
 14  termtype_desc        49653 non-null  object
 15  terminationdate_key  49653 non-null  object
 16  STATUS               49653 non-null  object
 17  BUSINESS_UNIT        49653 non-null  int64
dtypes: int64(6), object(12)
memory usage: 6.8+ MB


(49653, 18)
```

```
#4.FEATURE SELECTION (X,y) AND SCALING DATA (STANDARD SCALAR)
#-------------------------------------------------------------

X= data.iloc[:,0:5].values
y= data.iloc[:,17].values

print(X.shape)
print(y.shape)

X[:5]

y[:5]
```

```
(49653, 5)
(49653,)

array([0, 0, 0, 0, 0], dtype=int64)
```

```
#DATA SPLITTING
#--------------

Xtrain,Xtest,ytrain,ytest =
train_test_split(X,y,test_size=0.20,random_state=2)

print('TARINING INPUT SAMPLES COUNT ==>',Xtrain.shape)
print('TRAINING OUTPUT SAMPLES COUNT ==>',ytrain.shape)
print('TESTING INPUT SAMPLE COUNT ==>',Xtest.shape)
print('TESTING OUTPUT SAMPLE COUNT ==>',ytest.shape)
```

```
TARINING INPUT SAMPLES COUNT ==> (39722, 5)
TRAINING OUTPUT SAMPLES COUNT ==> (39722,)
TESTING INPUT SAMPLE COUNT ==> (9931, 5)
TESTING OUTPUT SAMPLE COUNT ==> (9931,)
```

```
#IMPLEMENTING THE ALGORITHM
```

TERMINATION PREDICTION USING DECISION TREE

```
#5.model creation by invoking the algorithm
#------------------------------------------

dt=
DecisionTreeClassifier(max_depth=3,criterion='gini',random_state=3)
```

```
#6.model training by fitting the X and y data(X_train and y_train)
#-----------------------------------------------------------------

dt.fit(Xtrain,ytrain)
```

```
DecisionTreeClassifier(max_depth=3, random_state=3)
```

```
#7.model prediction (ypre) -'using x_test'
#----------------------------------------

ypre = dt.predict(Xtest)
```

```
#8.calculate performance accuracy using output matrix
#----------------------------------------------------

accuracy_score(ytest,ypre)
```

```
0.9997986104118417
```

```
etp=
DecisionTreeClassifier(max_depth=3,criterion='entropy',random_state=1)
```

```
etp.fit(Xtrain,ytrain)
```

```
DecisionTreeClassifier(criterion='entropy', max_depth=3,
random_state=1)
```

```
ypre_ent= etp.predict(Xtest)
```

```
accuracy_score(ytest,ypre_ent)
```

```
0.9997986104118417
```

TERMINATION PREDICTION USING NAIVE BAYES CLASSIFIER

```
logreg = LogisticRegression()
```

```
logreg.fit(Xtrain,ytrain)
```

```
LogisticRegression()

ypre_log = logreg.predict(Xtest)

accuracy_score(ytest,ypre_log)

0.9915416372973517
```

TERMINATION PREDICTION USING NAIVE BAYES CLASSIFIER

```
gau = GaussianNB()

gau.fit(Xtrain,ytrain)

GaussianNB()

test_gpred=gau.predict(Xtest)

accuracy_score(ytest,test_gpred)

0.9997986104118417

d=MultinomialNB()

d.fit(Xtrain,ytrain)

MultinomialNB()

test_mulpred = d.predict(Xtest)

accuracy_score(ytest,test_mulpred)

0.8592286778773537
```

TERMINATION PREDICTION USING RANDAM FOREST CLASSIFIER

```
#5.model creation by invoking the algorithm
#------------------------------------------
clfr=RandomForestClassifier(n_estimators=10,criterion='entropy',random_state=0)

#6.model training by fitting the X and y data(X_train and y_train)
#-----------------------------------------------------------------

clfr.fit(Xtrain,ytrain)
```

```
RandomForestClassifier(criterion='entropy', n_estimators=10,
random_state=0)
```

```
#7.model prediction (ypre) -'using x_test'
#----------------------------------------

ypre = dt.predict(Xtest)
```

```
#8.calculate performance accuracy using output matrix
#----------------------------------------------------

accuracy_score(ytest,ypre)
```

```
0.9997986104118417
```

```
clfr1=RandomForestClassifier(n_estimators=10,criterion='gini',random_s
tate=0)
```

```
clfr1.fit(Xtrain,ytrain)
```

```
RandomForestClassifier(n_estimators=10, random_state=0)
```

```
ypre1 = dt.predict(Xtest)
```

```
accuracy_score(ytest,ypre1)
```

```
0.9997986104118417
```

TERMINATION PREDICTION USING KNEIGHBORS CLASSIFIER

```
#5.model creation by invoking the algorithm
#------------------------------------------
knn = KNeighborsClassifier(n_neighbors=7)
```

```
knn.fit(Xtrain,ytrain)
```

```
KNeighborsClassifier(n_neighbors=7)
```

```
knn_ypre= dt.predict(Xtest)
accuracy_score(ytest,knn_ypre)
```

```
0.9997986104118417
```

```
compare=pd.DataFrame({'actual
output':ytest,'gini_dt':ypre,'entro_dt':ypre_ent,'logreg':ypre_log ,'G
aussianNB':test_gpred,'MultinomialNB':test_mulpred,'Gini_RF':ypre1,'En
tropy_RF':ypre,'KNeighbors':knn_ypre})
```

```
compare
```

```
      actual output  gini_dt  entro_dt  logreg  GaussianNB
MultinomialNB  \
0                  1        1         1       1           1
1
1                  1        1         1       1           1
1
2                  1        1         1       1           1
1
3                  1        1         1       1           1
1
4                  1        1         1       1           1
1
...              ...      ...       ...     ...         ...
...
9926               1        1         1       1           1
1
9927               1        1         1       1           1
1
9928               1        1         1       1           1
1
9929               1        1         1       1           1
0
9930               1        1         1       1           1
1

      Gini_RF  Entropy_RF  KNeighbors
0           1           1           1
1           1           1           1
2           1           1           1
3           1           1           1
4           1           1           1
...       ...         ...         ...
9926        1           1           1
9927        1           1           1
9928        1           1           1
9929        1           1           1
9930        1           1           1

[9931 rows x 9 columns]
```

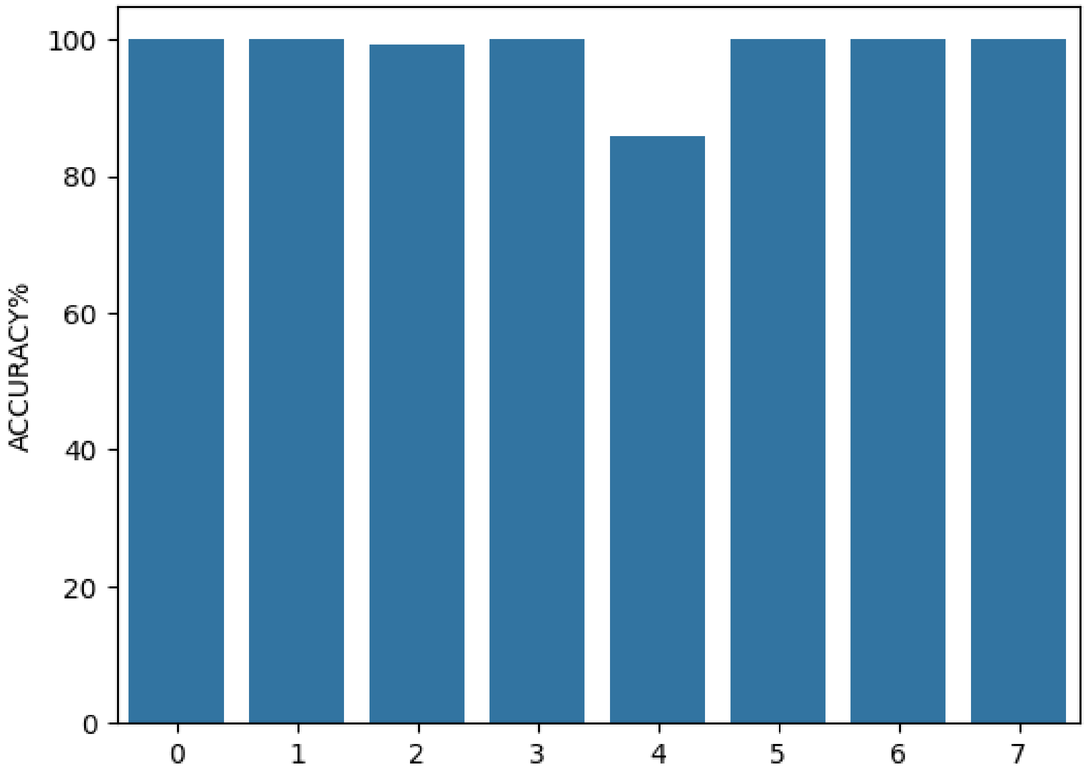
```
report=pd.DataFrame({'MODEL':['giniDT','entropyDT','logreg','guassNB','multinoNB','Gini_RF','Entropy_RF','KNeighbors'],'ACCURACY%':[accuracy_score(ytest,ypre)*100,accuracy_score(ytest,ypre_ent)*100,accuracy_score(ytest,ypre_log)*100,accuracy_score(ytest,test_gpred)*100,accuracy_score(ytest,test_mulpred)*100,accuracy_score(ytest,ypre1)*100,accuracy_score(ytest,ypre)*100,accuracy_score(ytest,knn_ypre)*100]})

report
```

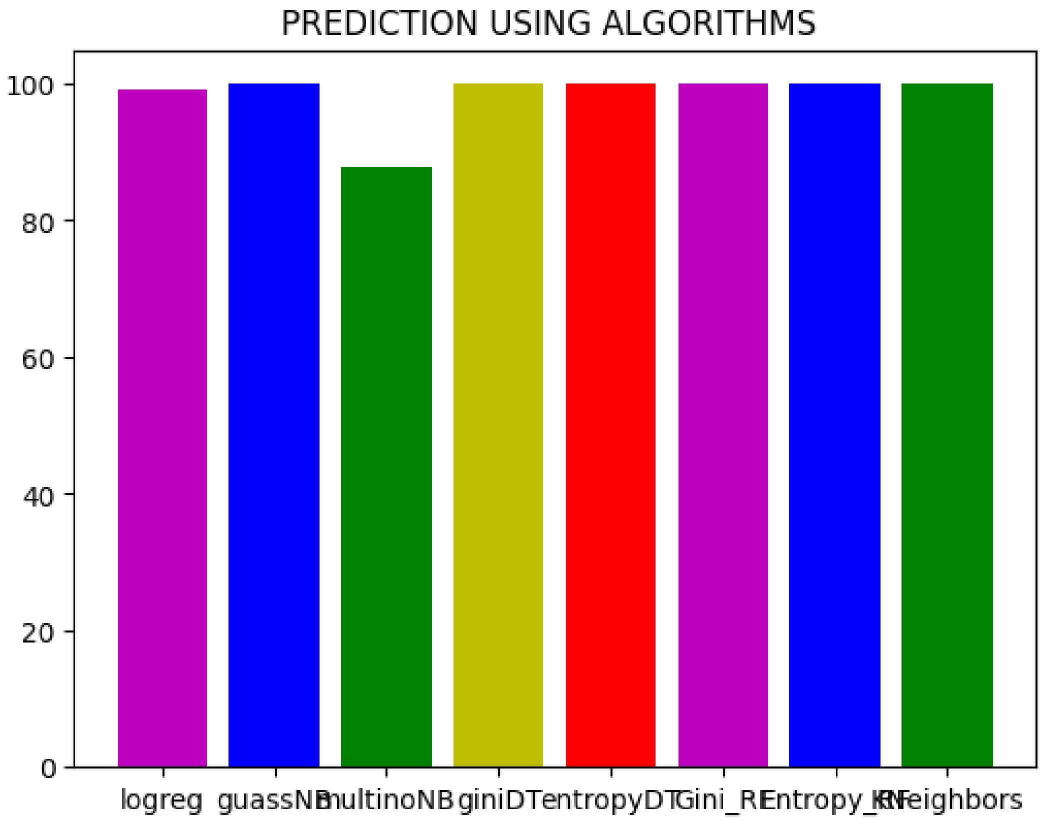
```
         MODEL  ACCURACY%
0       giniDT  99.979861
1    entropyDT  99.979861
2       logreg  99.154164
3      guassNB  99.979861
4    multinoNB  85.922868
5      Gini_RF  99.979861
6   Entropy_RF  99.979861
7   KNeighbors  99.979861
```

```
sns.barplot(report['ACCURACY%'])
```

```
<Axes: ylabel='ACCURACY%'>
```



```
count = [99.14,99.97,87.72,99.97,99.97,99.97,99.97,99.97]
color_code = ['m','b','g','y','r','m','b','g']
plt.bar(['logreg','guassNB','multinoNB','giniDT','entropyDT','Gini_RF'
,'Entropy_RF','KNeighbors'],count,color = color_code)
plt.title('PREDICTION USING ALGORITHMS ')
plt.show()
```



```
Conclusion: In this study, we investigated the predictive capability
of Decision Tree, Logistic Regression,NaiveBayes, KNeighbors
and Naive Bayes algorithms in assessing  MFG10 Year Termination on
dataset
criterion="entropy","ginDT","GussNB","logreg",'Gini_RF','Entropy_RF','
KNeighbors',
outperforms the other methods, achieving animpressive accuracy rating
of 99.97%.
```