```
from google.colab import drive
drive.mount('/content/drive')
```

```
     Mounted at /content/drive
```

```
import os
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import warnings
warnings.filterwarnings("ignore")
import tensorflow as tf
from sklearn import model_selection
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
from __future__ import absolute_import, division, print_function, unicode_literals
from IPython.display import clear_output
from six.moves import urllib

import tensorflow.compat.v2.feature_column as fc

import tensorflow as tf
os.getcwd()
os.listdir('.')
print(os.getcwd())
print(os.chdir('/content/drive/MyDrive/Proj_colab'))
```

```
     /content/drive/My Drive/Proj_colab
     None
```

```
# %% Read the original data and drop the columns
originalD = pd.read_csv('data/Original_data.csv', low_memory=False)
#originalD = pd.read_csv('data/Original_data.csv', low_memory=False)
original_F = originalD.drop(['birthyr','faminc','employ','marstat','child18','pid3','pid7'
original_F
```

```
             id  gender  race  educ  region
0     371823339       1     1     2       2
1     398212310       1     1     2       3
2     392933925       1     1     1       1
3     372445135       1     1     2       2
4     392602384       1     1     2       3
...         ...     ...   ...   ...     ...
4995  287972460       2     6     2       2
4996  137306469       2     6     2       3
```

```
# %% Read the breached  data and drop the columns
breachD = pd.read_csv('data/breached_data.csv', low_memory=False)
breach_F = breachD.drop(['Title','Domain','Name','BreachDate','AddedDate','ModifiedDate','
breach_F.loc[:,'Breached'] ='1'
breach_F["Breached"] = breach_F["Breached"].astype(object).astype(int)
breach_F
```

```
              id  Breached
0      135664815         1
1      355286483         1
2      355286483         1
3      355286483         1
4      339141795         1
...          ...       ...
14974  131884325         1
14975  131884325         1
14976  131884325         1
14977  131884325         1
14978  131884325         1

14979 rows × 2 columns
```

```
breach_F1 = breach_F.drop_duplicates(subset =["id"] )
breach_F1["Breached"].replace({1: 0},inplace = True)
#df["column1"].replace({"a": "x", "b": "y"}, inplace=True)
#breach_F = breach_
#breach_F.loc[:,'Breached'] ='1'
breach_F1
```

	id	Breached
0	135664815	0
1	355286483	0
4	339141795	0
5	341961164	0
6	374206867	0
...	...	...
14960	137327203	0
14963	334328189	0
14967	151192859	0
14973	152094711	0
14974	131884325	0

```
df3 = pd.merge(breach_F, breach_F1, how='outer')
df3
```

	id	Breached
0	135664815	1
1	355286483	1
2	355286483	1
3	355286483	1
4	339141795	1
...	...	...
19116	137327203	0
19117	334328189	0
19118	151192859	0
19119	152094711	0
19120	131884325	0

19121 rows × 2 columns

```
# %% Merge the two files
fin_dat = pd.merge(original_F, df3, on='id', how='inner')
print("Number of rows in the final dataset: ", fin_dat.shape[0])
fin_dat.head(5)
```

```
Number of rows in the final dataset:  19121
```

	id	gender	race	educ	region	Breached
0	371823339	1	1	2	2	1
1	371823339	1	1	2	2	1
2	371823339	1	1	2	2	0
3	392933925	1	1	1	1	1

```
#input
#x=fin_dat.drop('Breached',axis=1)
#y= fin_dat.Breached

X = fin_dat.iloc[:, :-1].values
y = fin_dat.iloc[:, -1].values
#splitting
#x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state =
#printing shapes of testing and training sets :
print("shape of original dataset :", fin_dat.shape)
print("shape of input - training set", X_train.shape)
print("shape of output - training set", y_train.shape)
print("shape of input - testing set", X_test.shape)
print("shape of output - testing set", y_test.shape)
```

```
shape of original dataset : (19121, 6)
shape of input - training set (15296, 5)
shape of output - training set (15296,)
shape of input - testing set (3825, 5)
shape of output - testing set (3825,)
```

```
# Naive Bayes
classifier = GaussianNB()
classifier.fit(X_train, y_train)

y_pred = classifier.predict(X_test)

# Summary of the predictions made by the classifier
print(classification_report(y_test, y_pred))
print(confusion_matrix(y_test, y_pred))
# Accuracy score
from sklearn.metrics import accuracy_score
print('accuracy is',accuracy_score(y_pred,y_test))
```

```
              precision    recall  f1-score   support

           0       0.00      0.00      0.00       806
           1       0.79      1.00      0.88      3019

    accuracy                           0.79      3825
   macro avg       0.39      0.50      0.44      3825
weighted avg       0.62      0.79      0.70      3825

[[   0  806]
```

```
 [   0 3019]]
accuracy is 0.7892810457516339
```