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

```
dataset = pd.read_csv('/content/drive/My Drive/kddcup99_csv.csv')
```

```
dataset.head(5)
```

	duration	protocol_type	service	flag	src_bytes	dst_bytes	land	wrong_fragmen
0	0	tcp	http	SF	181	5450	0	(
1	0	tcp	http	SF	239	486	0	(
2	0	tcp	http	SF	235	1337	0	(
3	0	tcp	http	SF	219	1337	0	(
4	0	tcp	http	SF	217	2032	0	(

selecting the feature from varaiance method

```
print(dataset.var()['src_bytes'])
print(dataset.var()['dst_bytes'])
print(dataset.var()['land'])
```

```
976576992036.6654
1091643891.0952706
4.453071700180602e-05
```

```
print(dataset.var()['wrong_fragment'])
print(dataset.var()['urgent'])
print(dataset.var()['hot'])
```

```
0.018172491590816044
3.03630038804738e-05
0.6116856844184144
```

```
print(dataset.var()['num_failed_logins'])
print(dataset.var()['logged_in'])
print(dataset.var()['lnum_compromised'])
print(dataset.var()['lroot_shell'])
print(dataset.var()['lsu_attempted'])
```

```
0.00024085837553570224
0.12626868283082365
3.2339838746190672
0.0001113193556638224
6.072508173886537e-05
```

```
print(dataset.var()['lnum_root'])
print(dataset.var()['lnum_file_creations'])
print(dataset.var()['lnum_shells'])
print(dataset.var()['lnum_access_files'])
#print(dataset.var()['lnum_outbound_cmds'])
#print(dataset.var()['is_host_login'])
print(dataset.var()['is_guest_login'])
```

```
4.051043258019034
0.009296040477405787
0.000121440870502456
0.001330916397839991
0.001384663728080717
```

```
print(dataset.var()['count'])
print(dataset.var()['srv_count'])
print(dataset.var()['serror_rate'])
print(dataset.var()['srv_serror_rate'])
print(dataset.var()['rerror_rate'])
print(dataset.var()['srv_rerror_rate'])
print(dataset.var()['same_srv_rate'])
print(dataset.var()['diff_srv_rate'])
print(dataset.var()['srv_diff_host_rate'])
print(dataset.var()['dst_host_count'])
print(dataset.var()['dst_host_srv_count'])
print(dataset.var()['dst_host_same_srv_rate'])
print(dataset.var()['dst_host_diff_srv_rate'])
```

```
45431.6891034859
60674.89003505529
0.1449456310521699
0.14517386836860224
0.0536495355281265
0.05389232330783698
0.15069129958002264
0.006757755850526054
0.020276896258574477
4191.863370949552
11244.525010698932
0.1687402028545784
0.011937575833318366
```

```
print(dataset.var()['dst_host_same_src_port_rate'])
print(dataset.var()['dst_host_srv_diff_host_rate'])
print(dataset.var()['dst_host_serror_rate'])
print(dataset.var()['dst_host_srv_serror_rate'])
print(dataset.var()['dst_host_rerror_rate'])
print(dataset.var()['dst_host_srv_rerror_rate'])
```

```
0.2316583307814032
0.0017751826094751953
0.14485133584224127
0.1450998566016115
```

remove redudunant feature

```
dataset['lsu_attempted'].value_counts()
dataset.drop('lsu_attempted', axis=1, inplace=True)
dataset['urgent'].value_counts()
dataset.drop('urgent', axis=1, inplace=True)
dataset['lnum_outbound_cmds'].value_counts()
dataset.drop('lnum_outbound_cmds', axis=1, inplace=True)
dataset['is_host_login'].value_counts()
dataset.drop('is_host_login', axis=1, inplace=True)
dataset['wrong_fragment'].value_counts()
dataset.drop('wrong_fragment', axis=1, inplace=True)
dataset['hot'].value_counts()
dataset.drop('hot', axis=1, inplace=True)
dataset['num_failed_logins'].value_counts()
dataset.drop('num_failed_logins', axis=1, inplace=True)
dataset['logged_in'].value_counts()
dataset.drop('logged_in', axis=1, inplace=True)
dataset['lroot_shell'].value_counts()
dataset.drop('lroot_shell', axis=1, inplace=True)
dataset['lnum_file_creations'].value_counts()
dataset.drop('lnum_file_creations', axis=1, inplace=True)
dataset['lnum_shells'].value_counts()
dataset.drop('lnum_shells', axis=1, inplace=True)
dataset['lnum_access_files'].value_counts()
dataset.drop('lnum_access_files', axis=1, inplace=True)
dataset['is_guest_login'].value_counts()
dataset.drop('is_guest_login', axis=1, inplace=True)
dataset['serror_rate'].value_counts()
dataset.drop('serror_rate', axis=1, inplace=True)
dataset['srv_serror_rate'].value_counts()
dataset.drop('srv_serror_rate', axis=1, inplace=True)
```

```
dataset['rerror_rate'].value_counts()
dataset.drop('rerror_rate', axis=1, inplace=True)
dataset['srv_rerror_rate'].value_counts()
dataset.drop('srv_rerror_rate', axis=1, inplace=True)
dataset['same_srv_rate'].value_counts()
dataset.drop('same_srv_rate', axis=1, inplace=True)
dataset['diff_srv_rate'].value_counts()
dataset.drop('diff_srv_rate', axis=1, inplace=True)
dataset['srv_diff_host_rate'].value_counts()
dataset.drop('srv_diff_host_rate', axis=1, inplace=True)
dataset['dst_host_same_srv_rate'].value_counts()
dataset.drop('dst_host_same_srv_rate', axis=1, inplace=True)
dataset['dst_host_diff_srv_rate'].value_counts()
dataset.drop('dst_host_diff_srv_rate', axis=1, inplace=True)
dataset['dst_host_same_src_port_rate'].value_counts()
```

```
dataset['dst_host_same_src_port_rate'].value_counts()
dataset.drop('dst_host_same_src_port_rate', axis=1, inplace=True)
dataset['dst_host_srv_diff_host_rate'].value_counts()
dataset.drop('dst_host_srv_diff_host_rate', axis=1, inplace=True)
dataset['dst_host_serror_rate'].value_counts()
dataset.drop('dst_host_serror_rate', axis=1, inplace=True)
dataset['dst_host_srv_serror_rate'].value_counts()
dataset.drop('dst_host_srv_serror_rate', axis=1, inplace=True)
dataset['dst_host_rerror_rate'].value_counts()
dataset.drop('dst_host_rerror_rate', axis=1, inplace=True)
dataset['dst_host_srv_rerror_rate'].value_counts()
dataset.drop('dst_host_srv_rerror_rate', axis=1, inplace=True)
```

```
dataset.head()
#dataset.describe()
```

```
   duration  protocol_type  service  flag  src_bytes  dst_bytes  land  lnum_compromi
0         0            tcp     http    SF        181       5450     0
1         0            tcp     http    SF        239        486     0
2         0            tcp     http    SF        235       1337     0
3         0            tcp     http    SF        219       1337     0
4         0            tcp     http    SF        217       2032     0
```

```
dataset['label'] = dataset['label'].replace(['back', 'buffer_overflow', 'ftp_write', 'gues
```

```
dataset.describe()
```

```
             duration     src_bytes     dst_bytes           land  lnum_compromised
count   494020.000000  4.940200e+05  4.940200e+05  494020.000000     494020.000000  49
mean        47.979400  3.025616e+03  8.685308e+02       0.000045          0.010212
std        707.747185  9.882191e+05  3.304003e+04       0.006673          1.798328
min          0.000000  0.000000e+00  0.000000e+00       0.000000          0.000000
25%          0.000000  4.500000e+01  0.000000e+00       0.000000          0.000000
50%          0.000000  5.200000e+02  0.000000e+00       0.000000          0.000000
75%          0.000000  1.032000e+03  0.000000e+00       0.000000          0.000000
max      58329.000000  6.933756e+08  5.155468e+06       1.000000        884.000000
```

```
x = dataset.iloc[:, :-1].values
#x
y = dataset.iloc[:, 13].values
```

```
#y
```

```
x
```

```
array([[0, 'tcp', 'http', ..., 8, 9, 9],
       [0, 'tcp', 'http', ..., 8, 19, 19],
       [0, 'tcp', 'http', ..., 8, 29, 29],
       ...,
       [0, 'tcp', 'http', ..., 18, 16, 255],
       [0, 'tcp', 'http', ..., 12, 26, 255],
       [0, 'tcp', 'http', ..., 35, 6, 255]], dtype=object)
```

```
x.shape
```

```
(494020, 13)
```

```
y.shape
```

```
(494020,)
```

```
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
from sklearn.compose import ColumnTransformer
labelencoder_x_1 = LabelEncoder()
labelencoder_x_2 = LabelEncoder()
labelencoder_x_3 = LabelEncoder()
x[:, 1] = labelencoder_x_1.fit_transform(x[:, 1])
x[:, 2] = labelencoder_x_2.fit_transform(x[:, 2])
x[:, 3] = labelencoder_x_3.fit_transform(x[:, 3])
```

```
x
```

```
array([[0, 1, 22, ..., 8, 9, 9],
       [0, 1, 22, ..., 8, 19, 19],
       [0, 1, 22, ..., 8, 29, 29],
       ...,
       [0, 1, 22, ..., 18, 16, 255],
       [0, 1, 22, ..., 12, 26, 255],
       [0, 1, 22, ..., 35, 6, 255]], dtype=object)
```

```
 from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.3, random_state =
```

```
#feature scaling
from sklearn.preprocessing import StandardScaler
sc_x = StandardScaler()
x_train = sc_x.fit_transform(x_train)
x_test = sc_x.transform(x_test)
```

```
from sklearn.naive_bayes import GaussianNB
classifier = GaussianNB()
classifier.fit(x_train, y_train)
```

```
GaussianNB(priors=None, var_smoothing=1e-09)
```

```
# Predicting the Test set results
y_pred = classifier.predict(x_test)
```

```
# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix
confusion_matrix(y_test, y_pred)
```

```
array([[117292,   1559],
       [  7624,  21731]])
```

```
from sklearn.model_selection import cross_val_score
accuracies = cross_val_score(estimator = classifier, X = x_train, y = y_train, cv = 5)
accuracies.mean()
accuracies.std()
```

```
0.01824981895534346
```

```
from sklearn import metrics
# Model Accuracy, how often is the classifier correct?
print("Accuracy nb:",metrics.accuracy_score(y_test, y_pred))
```

```
Accuracy nb: 0.9380389457916684
```

```
from sklearn import metrics
# Model Accuracy, how often is the classifier correct?
print(metrics.classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

      attack       0.94      0.99      0.96    118851
      normal       0.93      0.74      0.83     29355

    accuracy                           0.94    148206
   macro avg       0.94      0.86      0.89    148206
weighted avg       0.94      0.94      0.94    148206
```

```
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.metrics import accuracy_score
#for optimization i used AdaBoostClassifier

gbt = GradientBoostingClassifier()
#abc2=AdaBoostClassifier(n_estimators=10,base_estimator=gbt,learning_rate=0.01)
gbt1=gbt.fit(x_train,y_train)
predictions = gbt1.predict(x_test)
print("accuracy:",accuracy_score(y_test, predictions)*100)
```

```
     accuracy: 99.88192110980661
```

```
from sklearn.linear_model import SGDClassifier
sgb = SGDClassifier(loss="hinge", penalty="l1")
#abc3=AdaBoostClassifier(n_estimators=100,base_estimator=sgb,learning_rate=0.01)
sgb1=sgb.fit(x_train, y_train)
predictions = sgb1.predict(x_test)#
print("accuracy for SGD:",accuracy_score(y_test, predictions)*100)
```

```
accuracy for SGD: 98.5648354317639
```

```
from sklearn.ensemble import AdaBoostClassifier
abt = AdaBoostClassifier(n_estimators=100)

abt1=abt.fit(x_train, y_train)
predictions = abt1.predict(x_test)
print("accuracy:",accuracy_score(y_test, predictions)*100)
```

```
accuracy: 99.8623537508603
```

```
from sklearn.tree import DecisionTreeClassifier
clf=DecisionTreeClassifier(max_leaf_nodes=15,criterion='gini')
#abc5=AdaBoostClassifier(n_estimators=100,base_estimator=clf,learning_rate=0.01)
clf1=clf.fit(x_train,y_train)
predictions = clf1.predict(x_test)
print("accuracy for decision tree:",accuracy_score(y_test, predictions)*100)
```

```
accuracy for decision tree: 99.8319906076677
```

```
from sklearn.ensemble import RandomForestClassifier
clf2 = RandomForestClassifier(n_estimators=1000,max_leaf_nodes=15)
#abc6=AdaBoostClassifier(n_estimators=100,base_estimator=clf2,learning_rate=0.01)
clf5=clf2.fit(x_train,y_train)
predictions = clf5.predict(x_test)
print("accuracy for RFC:",accuracy_score(y_test, predictions)*100)
```

```
accuracy for RFC: 99.72740644778214
```

```
from sklearn.neural_network import MLPClassifier
mlp = MLPClassifier(hidden_layer_sizes=(100, 100, 12), alpha=1e-4, solver='sgd', random_st
model_3 = mlp.fit(x_train, y_train)
y_pred = model_3.predict(x_test)
acc = metrics.accuracy_score(y_test, y_pred)
print("This is Multi Layer Perceptron classifier \n\n")
print("Accuracy: {: .4f} %".format(acc*100))
```

```
This is Multi Layer Perceptron classifier


Accuracy:  99.8576 %
```