Dynamic Analysis

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import requests
import tensorflow as tf
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler,normalize
from sklearn import preprocessing
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import accuracy_score, roc_auc_score, roc_curve
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2
from sklearn.cluster import KMeans
from sklearn.decomposition import PCA
# from sklearn_extra.cluster import KMedoids
from sklearn import preprocessing
from sklearn.metrics import classification_report
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.model_selection import StratifiedShuffleSplit
from sklearn.model_selection import GridSearchCV
from sklearn import preprocessing
from tensorflow.keras.optimizers import SGD,Adam
from sklearn.ensemble import BaggingClassifier
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import StackingClassifier,VotingClassifier
from sklearn.ensemble import GradientBoostingClassifier
from xgboost import XGBClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
```

Loading the data

```
df=pd.read_csv("/content/android_traffic.csv",delimiter=";")
df.head()
```

	name	tcp_packets	dist_port_tcp	external_ips	vulume_bytes	udp_packets	tcp
0	AntiVirus	36	6	3	3911	0	
1	AntiVirus	117	0	9	23514	0	

```
df.shape
```

```
(7845, 17)
```

Checking and removing NULL values

```
df.isna().sum()
```

```
name                      0
tcp_packets               0
dist_port_tcp             0
external_ips              0
vulume_bytes              0
udp_packets               0
tcp_urg_packet            0
source_app_packets        0
remote_app_packets        0
source_app_bytes          0
remote_app_bytes          0
duracion                7845
avg_local_pkt_rate      7845
avg_remote_pkt_rate     7845
source_app_packets.1       0
dns_query_times            0
type                       0
dtype: int64
```

Dropping Columns with NULL values

```
df.drop(columns=["duracion","avg_local_pkt_rate","avg_remote_pkt_rate"],inplace=True)
df.isna().sum()
```

```
name                    0
tcp_packets             0
dist_port_tcp           0
external_ips            0
vulume_bytes            0
udp_packets             0
tcp_urg_packet          0
source_app_packets      0
remote_app_packets      0
source_app_bytes        0
remote_app_bytes        0
source_app_packets.1    0
dns_query_times         0
type                    0
dtype: int64
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7845 entries, 0 to 7844
Data columns (total 14 columns):
 #   Column                Non-Null Count  Dtype 
---  ------                --------------  ----- 
 0   name                  7845 non-null   object
 1   tcp_packets           7845 non-null   int64 
 2   dist_port_tcp         7845 non-null   int64 
 3   external_ips          7845 non-null   int64 
 4   vulume_bytes          7845 non-null   int64 
 5   udp_packets           7845 non-null   int64 
 6   tcp_urg_packet        7845 non-null   int64 
 7   source_app_packets    7845 non-null   int64 
 8   remote_app_packets    7845 non-null   int64 
 9   source_app_bytes      7845 non-null   int64 
 10  remote_app_bytes      7845 non-null   int64 
 11  source_app_packets.1  7845 non-null   int64 
 12  dns_query_times       7845 non-null   int64 
 13  type                  7845 non-null   object
dtypes: int64(12), object(2)
memory usage: 858.2+ KB
```

Categories of Apps

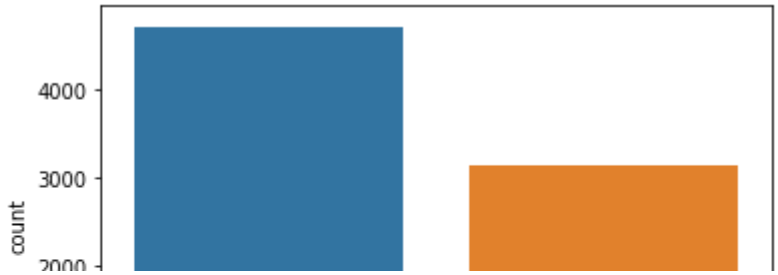
```
df["name"].value_counts()
```

```
Reading             774
Plankton            483
DroidKungFu         427
AntiVirus           396
NewsAndMagazines    360
                   ...
Spy.ImLog             1
Acnetdoor             1
SpyMob                1
Mobsquz               1
YcChar                1
Name: name, Length: 114, dtype: int64
```

Distribution of Target variable

```
sns.countplot(df["type"])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7f867d2e4950>
```



Labelling Categorical Text Values

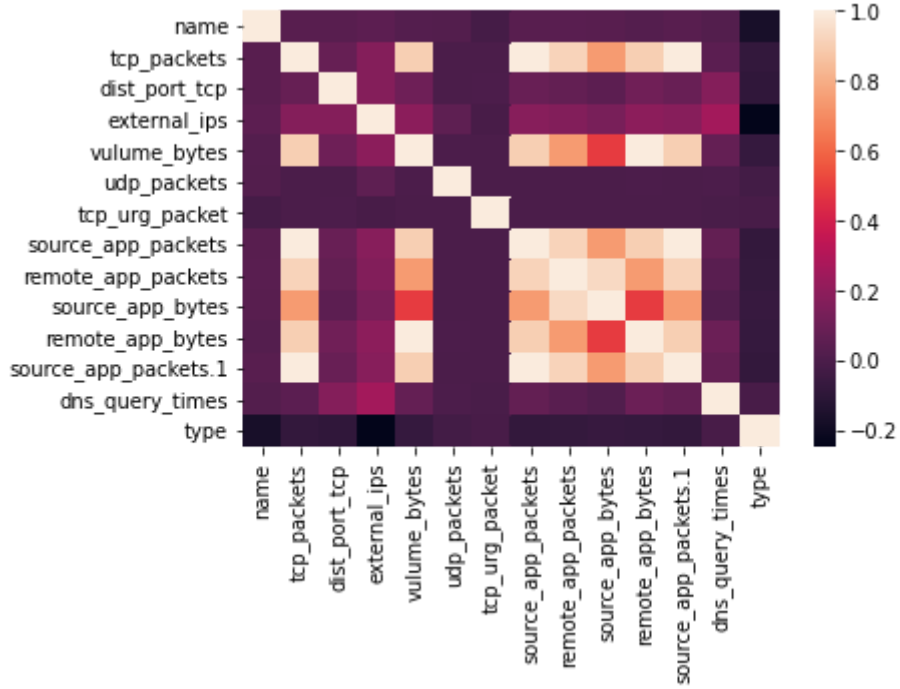
```
encoder=preprocessing.LabelEncoder()

df["name"]=encoder.fit_transform(df["name"])
df["type"]=encoder.fit_transform(df["type"])
```

Data Analysis

```
sns.heatmap(df.corr())
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f866da28a50>
```




```
cor=df.corr()
cor["type"]
```

```
name                 -0.168660
tcp_packets          -0.078641
dist_port_tcp        -0.086772
external_ips         -0.247536
vulume_bytes         -0.067534
udp_packets          -0.026907
tcp_urg_packet       -0.013049
```

```
source_app_packets     -0.078329
remote_app_packets     -0.074458
source_app_bytes       -0.063112
remote_app_bytes       -0.067577
source_app_packets.1   -0.078329
dns_query_times        -0.009106
type                    1.000000
Name: type, dtype: float64
```

```
df.drop(columns="dns_query_times",inplace=True)
```

```
cor=df.corr()
cor["type"]
```

```
name                   -0.168660
tcp_packets            -0.078641
dist_port_tcp          -0.086772
external_ips           -0.247536
vulume_bytes           -0.067534
udp_packets            -0.026907
tcp_urg_packet         -0.013049
source_app_packets     -0.078329
remote_app_packets     -0.074458
source_app_bytes       -0.063112
remote_app_bytes       -0.067577
source_app_packets.1   -0.078329
type                    1.000000
Name: type, dtype: float64
```

```
df.describe()
```

	name	tcp_packets	dist_port_tcp	external_ips	vulume_bytes	udp_pack
count	7845.000000	7845.000000	7845.000000	7845.000000	7.845000e+03	7845.000
mean	46.981262	147.578713	7.738177	2.748502	1.654375e+04	0.056
std	28.985376	777.920084	51.654222	2.923005	8.225650e+04	1.394
min	0.000000	0.000000	0.000000	0.000000	0.000000e+00	0.000
25%	19.000000	6.000000	0.000000	1.000000	8.880000e+02	0.000
50%	47.000000	25.000000	0.000000	2.000000	3.509000e+03	0.000
75%	72.000000	93.000000	0.000000	4.000000	1.218900e+04	0.000
max	113.000000	37143.000000	2167.000000	43.000000	4.226790e+06	65.000

```
# scaler=StandardScaler()

x=df.iloc[:][:-1]
y=df.iloc[:-1,-1]
x = normalize(x)
```

```
x= pd.DataFrame(x)
# x=scaler.fit_transform(x)
xtrain,xtest,ytrain,ytest=train_test_split(x,y)
```

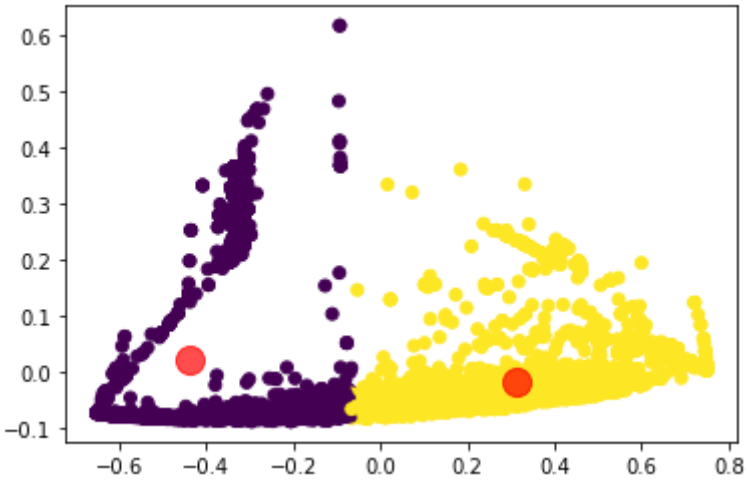
```
pca = PCA(n_components = 2)
X_train_principal = pca.fit_transform(xtrain)
X_train = pd.DataFrame(X_train_principal)
X_train.columns = ['P1', 'P2']
xtrain=pd.DataFrame(X_train)

kmeans = KMeans(n_clusters=2, random_state=40)
label = kmeans.fit_predict(xtrain)
centers=kmeans.cluster_centers_
centers
```

```
array([[-0.43634194,  0.02225338],
       [ 0.31053022, -0.015837  ]])
```

```
plt.scatter(xtrain["P1"],xtrain["P2"],c=label)
plt.scatter(centers[:, 0], centers[:, 1], c='red', s=200, alpha=0.7)
```

```
<matplotlib.collections.PathCollection at 0x7f866d904990>
```




```
pca.fit_transform(df.iloc[:][:-1])
```

```
array([[-197994.37476751,   -9849.70970646],
       [-175707.41032303,   17416.10187028],
       [ -38151.19660505,   12572.27328103],
       ...,
       [-203218.36533268,  -14853.27266992],
       [-203063.05245772,  -15238.27397576],
       [-203063.05245772,  -15238.27397576]])
```

```
# scaler=StandardScaler()
x=df.iloc[:][:-1]
y=df.iloc[:-1,-1]
x = normalize(x)
x= pd.DataFrame(x)
# x=scaler.fit_transform(x)
xtrain,xtest,ytrain,ytest=train_test_split(x,y)
```

```
len(y)
```

```
7844
```

SVM

```
clf = SVC(kernel="linear",verbose=True)
clf.fit(xtrain,ytrain)
```

```
[LibSVM]SVC(kernel='linear', verbose=True)
```

```
print(classification_report(ytest,clf.predict(xtest)))
svc=clf.score(x,y)
```

```
                   precision    recall  f1-score   support

               0       0.70      0.95      0.81      1176
               1       0.85      0.38      0.52       785

        accuracy                           0.72      1961
       macro avg       0.77      0.67      0.66      1961
    weighted avg       0.76      0.72      0.69      1961
```

RANDOM FORREST CLASSIFIER

```
rtclf=RandomForestClassifier()
rtclf.fit(xtrain,ytrain)
```

```
RandomForestClassifier()
```

```
print(classification_report(ytest,rtclf.predict(xtest)))
rt=rtclf.score(x,y)
```

```
                   precision    recall  f1-score   support

               0       1.00      1.00      1.00      1176
               1       1.00      1.00      1.00       785

        accuracy                           1.00      1961
       macro avg       1.00      1.00      1.00      1961
    weighted avg       1.00      1.00      1.00      1961
```

DEEP NEURAL NETWORK CLASSIFIER

```
```

```
xtrain[0]
```

```
6044    0.004358
3666    0.004931
6583    0.071473
5422    0.003495
6859    0.002728
          ...
1832    0.000202
5849    0.016714
5660    0.002551
5299    0.000999
275     0.003893
Name: 0, Length: 5883, dtype: float64
```

```
ddmodel=tf.keras.Sequential()
ddmodel.add(tf.keras.layers.Dense(512,input_shape=(13,)))
ddmodel.add(tf.keras.layers.Dense(256, activation=('relu') ))
ddmodel.add(tf.keras.layers.Dense(128, activation=('relu') ))
ddmodel.add(tf.keras.layers.Dense(64,  activation=('relu') ))
ddmodel.add(tf.keras.layers.Dense(32,  activation=('relu') ))
ddmodel.add(tf.keras.layers.Dense(16,  activation=('relu') ))
ddmodel.add(tf.keras.layers.Dense(1,  activation=('sigmoid') ))

ddmodel.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_14 (Dense)            (None, 512)               7168

 dense_15 (Dense)            (None, 256)               131328

 dense_16 (Dense)            (None, 128)               32896

 dense_17 (Dense)            (None, 64)                8256

 dense_18 (Dense)            (None, 32)                2080

 dense_19 (Dense)            (None, 16)                528

 dense_20 (Dense)            (None, 1)                 17

=================================================================
Total params: 182,273
Trainable params: 182,273
Non-trainable params: 0
_________________________________________________________________
```

```
ddmodel.compile(optimizer=tf.keras.optimizers.SGD(learning_rate=0.01),loss="binary_crossen

ddmodel.fit(xtrain,ytrain,epochs=20)
```

```
Epoch 1/20
184/184 [==============================] - 1s 4ms/step - loss: 0.6650 - accuracy: 0.5
```

```
Epoch 2/20
184/184 [==============================] - 1s 4ms/step - loss: 0.6439 - accuracy: 0.6
Epoch 3/20
184/184 [==============================] - 1s 4ms/step - loss: 0.6255 - accuracy: 0.6
Epoch 4/20
184/184 [==============================] - 1s 4ms/step - loss: 0.6043 - accuracy: 0.6
Epoch 5/20
184/184 [==============================] - 1s 4ms/step - loss: 0.5860 - accuracy: 0.7
Epoch 6/20
184/184 [==============================] - 1s 4ms/step - loss: 0.5732 - accuracy: 0.7
Epoch 7/20
184/184 [==============================] - 1s 4ms/step - loss: 0.5634 - accuracy: 0.7
Epoch 8/20
184/184 [==============================] - 1s 4ms/step - loss: 0.5618 - accuracy: 0.7
Epoch 9/20
184/184 [==============================] - 1s 4ms/step - loss: 0.5570 - accuracy: 0.7
Epoch 10/20
184/184 [==============================] - 1s 4ms/step - loss: 0.5547 - accuracy: 0.7
Epoch 11/20
184/184 [==============================] - 1s 4ms/step - loss: 0.5533 - accuracy: 0.7
Epoch 12/20
184/184 [==============================] - 1s 4ms/step - loss: 0.5532 - accuracy: 0.7
Epoch 13/20
184/184 [==============================] - 1s 4ms/step - loss: 0.5515 - accuracy: 0.7
Epoch 14/20
184/184 [==============================] - 1s 4ms/step - loss: 0.5531 - accuracy: 0.7
Epoch 15/20
184/184 [==============================] - 1s 4ms/step - loss: 0.5515 - accuracy: 0.7
Epoch 16/20
184/184 [==============================] - 1s 4ms/step - loss: 0.5527 - accuracy: 0.7
Epoch 17/20
184/184 [==============================] - 1s 4ms/step - loss: 0.5501 - accuracy: 0.7
Epoch 18/20
184/184 [==============================] - 1s 4ms/step - loss: 0.5497 - accuracy: 0.7
Epoch 19/20
184/184 [==============================] - 1s 4ms/step - loss: 0.5507 - accuracy: 0.7
Epoch 20/20
184/184 [==============================] - 1s 4ms/step - loss: 0.5516 - accuracy: 0.7
<keras.callbacks.History at 0x7f866d5579d0>
```

```
print(classification_report(np.round(ddmodel.predict(xtest)),ytest))
# dm=ddmodel.score(x,y)
```

```
                precision    recall  f1-score   support

           0.0       0.93      0.72      0.81      1507
           1.0       0.47      0.81      0.59       454

    accuracy                           0.74      1961
   macro avg       0.70      0.76      0.70      1961
weighted avg       0.82      0.74      0.76      1961
```

```
ddm=ddmodel.evaluate(xtest,ytest)[1]
```

```
62/62 [==============================] - 0s 2ms/step - loss: 0.5502 - accuracy: 0.741
```

Naive Bayes

```
gnb = GaussianNB()
gnb.fit(xtrain,ytrain)
```

```
GaussianNB()
```

```
print(classification_report(ytest,gnb.predict(xtest)))
gn=gnb.score(x,y)
```

```
                  precision    recall  f1-score   support

               0       0.91      0.93      0.92      1176
               1       0.90      0.86      0.88       785

        accuracy                           0.90      1961
       macro avg       0.90      0.90      0.90      1961
    weighted avg       0.90      0.90      0.90      1961
```

KNN Classifier

```
knn=neigh = KNeighborsClassifier(n_neighbors=3)
knn.fit(xtrain,ytrain)
```

```
KNeighborsClassifier(n_neighbors=3)
```

```
print(classification_report(ytest,knn.predict(xtest)))
kn=knn.score(xtest,ytest)
```

```
                  precision    recall  f1-score   support

               0       0.87      0.85      0.86      1176
               1       0.79      0.81      0.80       785

        accuracy                           0.84      1961
       macro avg       0.83      0.83      0.83      1961
    weighted avg       0.84      0.84      0.84      1961
```

```
from matplotlib.pyplot import figure

figure(figsize=(8, 6), dpi=80)
scores=[ddm,svc,rt,gn,kn]
names=["DNN","SVM","Random Forrest","Naive Bayes","KNN"]
plt.bar(names,scores)
plt.ylabel("Accuracy")
```

```
plt.title("Algorithm comparisions")
plt.tick_params(axis="x",which="major",labelsize=10)
```