FRAUD AND ANOMALY DETECTION

ASSIGNMENT 1

QUESTIONS(Ques)

ANSWERS(Ans)

Ques 1. Identify the nature of attributes and data types you have in your data set.

Ans 1. The given data set NSL-KDD has multiple files. I have used 2 files.

(i) For training model file used is: KDDTrain+.txt
(ii) For testing model file used is: KDDTest+.txt

Each file has 43 features. Training file has 125973 instances and Test file has 22544 instances. There are 3 types of attributes i.e. Nominal, Binary and Continuous.

S.no.	Attribute type	Attributes
1	Nominal	Protocol_type, Service, Flag
2	Binary	Land, logged_in, root_shell, su_attempted, is_host_login, is_guest_login
3	Continuous	Duration, src_bytes, dst_bytes, wrong_fragment, urgent, hot, num_failed_logins, num_compromised, num_root, num_file_creations, num_shells, num_access_files, num_outbound_cmds, count, srv_count, serror_rate, srv_serror_rate, rerror_rate, srv_rerror_rate, same_srv_rate, diff_srv_rate, srv_diff_host_rate, dst_host_count, dst_host_srv_count, dst_host_same_srv_rate, dst_host_diff_srv_rate, dst_host_same_src_port_rate, dst_host_srv_diff_host_rate, dst_host_serror_rate, dst_host_srv_serror_rate, dst_host_rerror_rate, dst_host_srv_rerror_rate

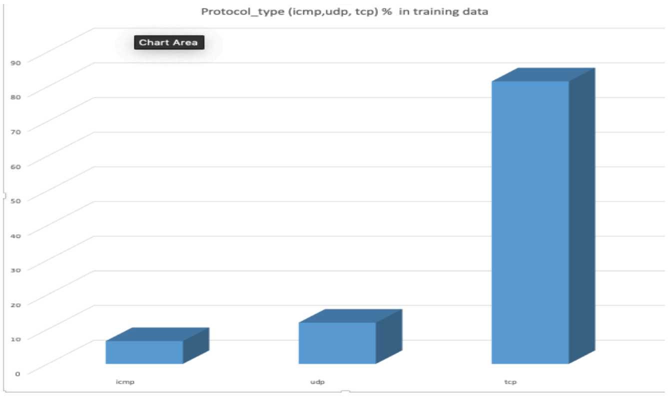
Data types of each attribute is shown in below table:

```
duration                          int64
protocol_type                    object
service                          object
flag                             object
src_bytes                         int64
dst_bytes                         int64
land                              int64
wrong_fragment                    int64
urgent                            int64
hot                               int64
num_failed_logins                 int64
logged_in                         int64
num_compromised                   int64
root_shell                        int64
su_attempted                      int64
num_root                          int64
num_file_creations                int64
num_shells                        int64
num_access_files                  int64
num_outbound_cmds                 int64
is_host_login                     int64
is_guest_login                    int64
count                             int64
srv_count                         int64
serror_rate                     float64
srv_serror_rate                 float64
rerror_rate                     float64
srv_rerror_rate                 float64
same_srv_rate                   float64
diff_srv_rate                   float64
srv_diff_host_rate              float64
dst_host_count                    int64
dst_host_srv_count                int64
dst_host_same_srv_rate          float64
dst_host_diff_srv_rate          float64
dst_host_same_src_port_rate     float64
dst_host_srv_diff_host_rate     float64
dst_host_serror_rate            float64
dst_host_srv_serror_rate        float64
dst_host_rerror_rate            float64
dst_host_srv_rerror_rate        float64
label                            object
complex                           int64
dtype: object
```

Ques 2 Any unique feature or pattern you identifies by visually looking at the data set.

Ans 2. After Data visualization multiple patterns came into notice. There are few features that I figured has great impact on the target feature. Here Target feature I considered is 'class'. It is a binary class that has two values 0 and 1. Here, 0 means 'normal' and 1 means 'abnormal' data . Target feature is derived from 'label' feature determining 39 types of attack and normal values.

• Protocol_types (icmp, udp,tcp) : Below graph determines the %of data instances present in training data that has following values. This concludes,



- 81.516674 % data instances are of protocol_type 'tcp'.

- 11.901757% - udp

- 6.581569% - icmp

•Protocol_type impacts Target feature class (normal and abnormal) as:

	protocol_type	normalORanomaly	typeCount
0	icmp	anomaly	6982
1	icmp	normal	1309
2	tcp	anomaly	49089
3	tcp	normal	53600
4	udp	anomaly	2559
5	udp	normal	12434

It means there is 48.5% chances that if protocol_type is udp then data is normal. If protocol_type is icmp then 53.3% chances that data is anomaly.

Ques 3. Comment on the data quality problem in your data set. Is there any noise, outliers, missing values, duplicate or wrong data?

Ans 3. As stated this dataset :

- It does not include redundant records in the train set, so the classifiers will not be biased towards more frequent records.
- There is no duplicate records in the proposed test sets; therefore, the performance of the learners are not biased by the methods which have better detection rates on the frequent records.

- The number of selected records from each difficulty level group is inversely proportional to the percentage of records in the original KDD data set. As a result, the classification rates of distinct machine learning methods vary in a wider range, which makes it more efficient to have an accurate evaluation of different learning techniques.
- The number of records in the train and test sets are reasonable, which makes it affordable to run the experiments on the complete set without the need to randomly select a small portion. Consequently, evaluation results of different research works will be consistent and comparable.

While working on this dataset, I found that in feature 'label' that determines the types of attack, there are some attacks that doesn't fall in either of the category i.e. DoS, R2L, U2R or Probe.

Outliers detection:

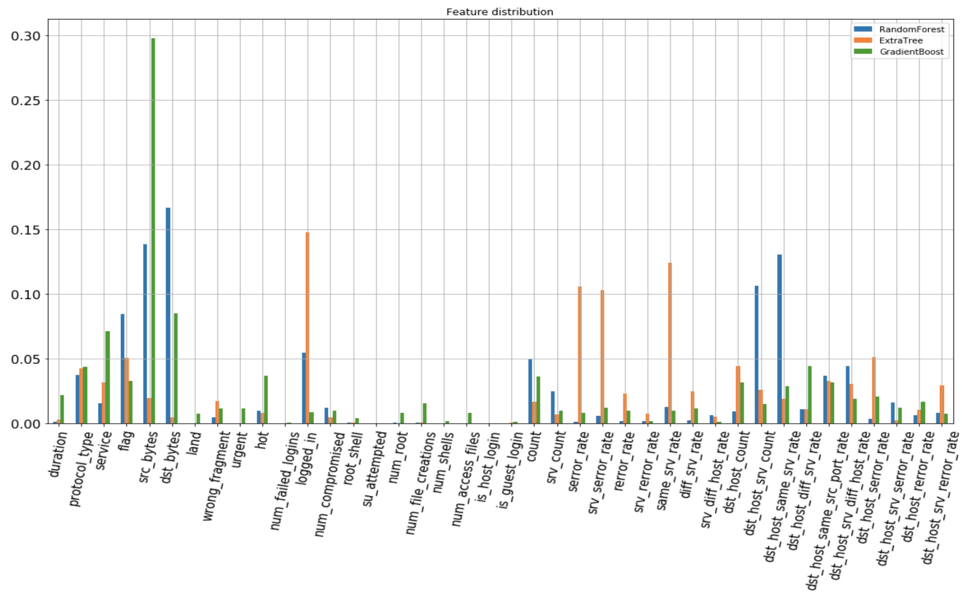
• I experimented on feature 'duration' and there are number of instances that are many outliers detected. To detect the outliers, I used Z-score to identify the outliers.

	duration	protocol_type	...	duration_zscore	is_outlier
43	9052	2	...	3.363735	True
115	25950	1	...	9.846659	True
165	9015	1	...	3.349540	True
253	9235	1	...	3.433943	True
289	36613	1	...	13.937523	True
396	31401	1	...	11.937937	True
527	10455	2	...	3.901997	True
533	10326	2	...	3.852506	True
591	41285	1	...	15.729937	True
605	13488	1	...	5.065608	True
776	17399	1	...	6.566065	True
785	21263	1	...	8.048490	True
976	9908	1	...	3.692140	True
1038	35682	1	...	13.580344	True
1047	15435	2	...	5.812575	True
1097	8486	2	...	3.146589	True
1141	37815	1	...	14.398671	True
1178	9114	1	...	3.387521	True
1235	41802	1	...	15.928285	True
1388	40703	1	...	15.506653	True
1460	8625	1	...	3.199916	True
1490	8556	2	...	3.173445	True
1494	37874	1	...	14.421306	True
1795	41111	1	...	15.663182	True
1944	9431	1	...	3.509139	True
1996	37749	1	...	14.373350	True
2029	39667	1	...	15.109191	True
2119	25641	1	...	9.728111	True
2136	38776	1	...	14.767359	True
2273	37688	1	...	14.349947	True

To build model and perform mathematical computation, I converted nominal input features into integer type by performing "**One-hot encoding**". It is performed on 'protocol_type', 'flag', and ' service' features.

After this, I calculated **standard deviation** of features and it showed that feature "num_outbound_cmds" has std_div = 0. So I drop this feature from the training and test data.

After this, I did Feature selection since the dimensionality of this dataset is high. To include features that has higher impact on target feature, I performed **Ensemble Feature selection techniques** i.e. RandomForestClassifier, GradientBoostingClassifier and ExtrTreeClassifier. After performing these techniques, I picked top 15 features from each category (non-duplicate) that results into input 25 features. So dimensionality of dataset got reduced from 43 to 25 input features.



Ques 4. Which anomaly detection technique(s) you will apply and why?

Ans 4. There are certain characteristics of this data set i.e.

- High dimensionality: Since the input features are large in this data set. So the algorithm to detect anomaly should be selected that can easily compute high dimensionality dataset.
- Outliers: Need to select the algorithm that can easily detect outliers.

I experimented with 3 algorithms:

1. **Decision Tree Classifier:** Decision trees have several advantages compared to other classification methods, which make them more suitable for outlier detection. In particular they have an easily interpretable structure and they are also less susceptible to the curse of dimensionality.

   - Inexpensive to construct
   - Extremely fast to classify unknown records
   - Robust to noise (especially when methods to avoid overfitting are employed)
   - Can easily handle redundant or irrelevant attributes (unless the attributes are interacting)

2. **Random Forest Classifier:** The main features of the random forests algorithm are listed as follows: It runs efficiently on large data sets with many features. It can give the estimates of what features are important. It has no nominal data problem and does not over- fit. It can handle unbalanced data sets.

3. **Gradient Boosting Classifier**: GBT build trees one at a time, where each new tree helps to correct errors made by previously trained tree. Researchers says application of GBM is *anomaly detection* in supervised learning settings where data is often highly unbalanced such as DNA sequences, credit card transactions or cyber security. But GBM sometimes more sensitive to overfitting if the data is noisy But in our case data is not much noisy so I used GBM.

Ques 6. Unique assumptions regarding the nature of anomalies made by the techniques in that category?

Ans 6. In each technique if there is type of attack apart from DoS, R2L, U2R and probe then, the output is "NaN". In order to solve this issue, I categorized all the types of attacks as 'abnormal' and non-attacks as 'normal'.

Ques 7. If you are applying classification-based algorithms, how did you split your records?

Ans 7. I applied Decision tree, Random forest and Gradient descent. For training the model number of instances I took was 125973 instances as present in KDDTrain+.txt file whereas for testing the model I took 22544 instances as present in KDDTest+.txt files. Input features and target features are split as shown below:

```
##########################################################################################
#splitting input feature and target feature of TRAIN data
X_train['class'].isna().sum() #checking if any null values
X_train = X_train.dropna() #dropping instances that have null values
train_data_X = X_train.iloc[:,0:len(X_train.columns)-2] #input features
train_data_Y = X_train.loc[:,'class']
train_data_Y = train_data_Y.astype('int')  #target feature 'class'

#splitting input feature and target feature of TEST data

X_test['class'].isna().sum() #checking if any null values
X_test = X_test.dropna() #dropping instances that have null values
test_data_X = X_test.iloc[:,0:len(X_test.columns)-2] #input features
test_data_Y = X_test.loc[:,'class']
test_data_Y = test_data_Y.astype('int')  #target feature 'class'
```

Ques 8. How did you measure the accuracy of anomaly detection algorithms?

Ans 8. Accuracy measure metric I selected is accuracy score.

```
conclusion = pd.DataFrame({'models': ["LOGISTIC REGRESSION","DECISION TREE CLASSIFIER","RANDOMFOREST CLASSIFIER","GRADIENT BOOSTING CLASSIFIER"],
                           'accuracies': [accuracy_score(test_data_Y, predicted),accuracy_score(test_data_Y,pred_Deci_tree),
                                          accuracy_score(test_data_Y, pred_Random_forest),accuracy_score(test_data_Y,pred_Gradient_boost)]})

print(conclusion)

###############################################################################################################3
```

The accuracy score of each model is listed below:

```
In [605]: print(conclusion)
                         models  accuracies
0           LOGISTIC REGRESSION    0.725309
1      DECISION TREE CLASSIFIER    0.850190
2       RANDOMFOREST CLASSIFIER    0.809039
3  GRADIENT BOOSTING CLASSIFIER    0.825500
```

Ques 9. Represent the list of attacks handled by your anomaly detection algorithm.

Ans 9. List of attacks handled by anomaly detection algorithms are:

- Dos
- R2L
- U2R
- Probe

abnormal

Ques 10. Which evaluation metrics did you used to evaluate the performance of your anomaly detection algorithm?

Ans 10. To evaluate the performance of anomaly detection algorithm, I performed Confusion matrix metrics and to do hyper-parameter optimization, I perform Cross fold validation.

Results are shown below:

```
Confusion matrix score:

1. Decision Tree
 [[9425  286]
 [2863 8446]]

2. Random Forest
 [[9449  262]
 [3752 7557]]

3. Decision Tree
 [[9426  285]
 [3383 7926]]
```

Accuracy score before Cross- validation and After cross-validation:

• Before cross- validation:

```
In [617]: print(conclusion)
                        models  accuracies
0          LOGISTIC REGRESSION    0.725309
1     DECISION TREE CLASSIFIER    0.850190
2      RANDOMFOREST CLASSIFIER    0.809039
3  GRADIENT BOOSTING CLASSIFIER   0.825500
```

• After cross-validation:

```
                         Accuracy Score
Logistic Regression            0.885435
Decision Tree                  0.998496
Random Forest                  0.998848
GRDIENT BOOSTING               0.998264
```

CODE

```
import pandas as pd
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.ensemble import ExtraTreesClassifier
from sklearn import preprocessing
from sklearn.preprocessing import OneHotEncoder
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix,accuracy_score
from sklearn.model_selection import cross_val_score
from scipy.stats import zscore



col_names = ["duration","protocol_type","service","flag","src_bytes",
    "dst_bytes","land","wrong_fragment","urgent","hot","num_failed_logins",
    "logged_in","num_compromised","root_shell","su_attempted","num_root",
    "num_file_creations","num_shells","num_access_files","num_outbound_cmds",
    "is_host_login","is_guest_login","count","srv_count","serror_rate",
    "srv_serror_rate","rerror_rate","srv_rerror_rate","same_srv_rate",
    "diff_srv_rate","srv_diff_host_rate","dst_host_count","dst_host_srv_count",
    "dst_host_same_srv_rate","dst_host_diff_srv_rate","dst_host_same_src_port_rate",
    "dst_host_srv_diff_host_rate","dst_host_serror_rate","dst_host_srv_serror_rate",
    "dst_host_rerror_rate","dst_host_srv_rerror_rate","label","complex"]

train_data = pd.read_csv('/Users/poojatyagi/Desktop/Fraud and anomaly/NSL-KDD/KDDTrain+.txt', header= None,
names = col_names)
test_data = pd.read_csv('/Users/poojatyagi/Desktop/Fraud and anomaly/NSL-KDD/KDDTest+.txt', header= None, names
= col_names)


train_data.dtypes
test_data.shape
####################################################################################################
######
#Pre-processing dataset
#removing complex column as our target feature is types of attack from traina nd test data
X_train = train_data.drop(['complex'], axis=1)
X_test = test_data.drop(['complex'], axis=1)

# DoS,Probe, U2R, R2L attackes as category abnormal... 0- normal and 1 - abnormal -- TRAIN DATA
X_train.loc[(X_train.label == 'back')
 | (X_train.label == 'land')
 | (X_train.label == 'neptune')
 | (X_train.label == 'pod')
 | (X_train.label == 'smurf')
 | (X_train.label == 'teardrop')
 | (X_train.label == 'apache2')
 | (X_train.label == 'udpstorm')
 | (X_train.label == 'processtable')
 | (X_train.label == 'worm')
 | (X_train.label == 'satan')
 | (X_train.label == 'ipsweep')
 | (X_train.label == 'nmap')
 | (X_train.label == 'portsweep')
 | (X_train.label == 'mscan')
```

```
 | (X_train.label == 'saint')
 | (X_train.label == 'ftp_write')
 | (X_train.label == 'imap')
 | (X_train.label == 'phf')
 | (X_train.label == 'multihop')
 | (X_train.label == 'warezmaster')
 | (X_train.label == 'warezclient ')
 | (X_train.label == 'spy')
 | (X_train.label == 'xlock')
 | (X_train.label == 'xsnoop')
 | (X_train.label == 'snmpguess')
 | (X_train.label == 'snmpgetattack')
 | (X_train.label == 'httptunnel')
 | (X_train.label == 'sendmail')
 | (X_train.label == 'named')
 | (X_train.label == 'buffer_overflow')
 | (X_train.label == 'loadmodule')
 | (X_train.label == 'rootkit')
 | (X_train.label == 'perl')
 | (X_train.label == 'sqlattack')
 | (X_train.label == 'xterm')
 | (X_train.label == 'ps'), 'class']= 1

#NORMAL
X_train.loc[(X_train.label == 'normal'), 'class']= 0

#DoS,Probe, U2R, R2L attackes as category abnormal... 0- normal and 1 - abnormal-- TEST DATA

#DoS
X_test.loc[(X_test.label == 'back')
 | (X_test.label == 'land')
 | (X_test.label == 'neptune')
 | (X_test.label == 'pod')
 | (X_test.label == 'smurf')
 | (X_test.label == 'teardrop')
 | (X_test.label == 'apache2')
 | (X_test.label == 'udpstorm')
 | (X_test.label == 'processtable')
 | (X_test.label == 'worm')
 | (X_test.label == 'satan')
 | (X_test.label == 'ipsweep')
 | (X_test.label == 'nmap')
 | (X_test.label == 'portsweep')
 | (X_test.label == 'mscan')
 | (X_test.label == 'saint')
 | (X_test.label == 'ftp_write')
 | (X_test.label == 'imap')
 | (X_test.label == 'phf')
 | (X_test.label == 'multihop')
 | (X_test.label == 'warezmaster')
 | (X_test.label == 'warezclient ')
 | (X_test.label == 'spy')
 | (X_test.label == 'xlock')
 | (X_test.label == 'xsnoop')
 | (X_test.label == 'snmpguess')
 | (X_test.label == 'snmpgetattack')
 | (X_test.label == 'httptunnel')
 | (X_test.label == 'sendmail')
```

```
 | (X_test.label == 'named')
 | (X_test.label == 'buffer_overflow')
 | (X_test.label == 'loadmodule')
 | (X_test.label == 'rootkit')
 | (X_test.label == 'perl')
 | (X_test.label == 'sqlattack')
 | (X_test.label == 'xterm')
 | (X_test.label == 'ps'), 'class']= 1

#NORMAL
X_test.loc[(X_test.label == 'normal'), 'class']= 0

##################################################################################################
###########
#splitting input feature and target feature of TRAIN data
X_train['class'].isna().sum() #checking if any null values
X_train = X_train.dropna() #dropping instances that have null values
train_data_X = X_train.iloc[:,0:len(X_train.columns)-2] #input features
train_data_Y = X_train.loc[:,'class']
train_data_Y = train_data_Y.astype('int')  #target feature 'class'

#splitting input feature and target feature of TEST data

X_test['class'].isna().sum() #checking if any null values
X_test = X_test.dropna() #dropping instances that have null values
test_data_X = X_test.iloc[:,0:len(X_test.columns)-2] #input features
test_data_Y = X_test.loc[:,'class']
test_data_Y = test_data_Y.astype('int')  #target feature 'class'

##################################################################################################
###########
#ONE-HOT ENCODING for 3 nominal features ['protocol_type','flag','service']
le = preprocessing.LabelEncoder()
enc = OneHotEncoder()
lb = preprocessing.LabelBinarizer()

train_data_X['protocol_type'] = le.fit_transform(train_data_X['protocol_type'])
test_data_X['protocol_type'] = le.fit_transform(test_data_X['protocol_type'])

train_data_X['flag'] = le.fit_transform(train_data_X['flag'])
test_data_X['flag'] = le.fit_transform(test_data_X['flag'])

train_data_X['service'] = le.fit_transform(train_data_X['service'])
test_data_X['service'] = le.fit_transform(test_data_X['service'])


##################################################################################################
####
#DATA VISUALIZATION

# Calculating standard devistion of features excluding continous features
std_list = ['protocol_type', 'service', 'flag','root_shell', 'land', 'logged_in', 'su_attempted', 'is_host_login', 'is_guest_login']
std_train = train_data_X.drop(std_list, axis=1)

#drop n smallest std features
stdtrain = std_train.std(axis=0)
std_X_train = stdtrain.to_frame()
std_X_train.nsmallest(10, columns=0).head(10)
```

```
#num_outbound_cmds has 0 std so will drop that feature
train_data_X = train_data_X.drop(['num_outbound_cmds'], axis=1)
test_data_X = test_data_X.drop(['num_outbound_cmds'], axis=1)

# Making list of 10 features with lowest standard deviation
stdrop_list = ['urgent', 'num_shells', 'root_shell',
       'num_failed_logins', 'num_access_files', 'dst_host_srv_diff_host_rate',
       'diff_srv_rate', 'dst_host_diff_srv_rate', 'wrong_fragment']

X_test_stdrop = test_data_X.drop(stdrop_list, axis=1)
X_train_stdrop = train_data_X.drop(stdrop_list, axis=1)
df_train_stdrop = pd.concat([X_train_stdrop, train_data_Y], axis=1)

#####################################################################################################
###
#OUTLIERS DETECTION
#Feature 'Duration'

traiNData_OD = train_data_X.copy()
traiNData_OD["duration_zscore"] = zscore(traiNData_OD["duration"])
traiNData_OD["is_outlier"] = traiNData_OD["duration_zscore"].apply(lambda x: x <= -3 or x >= 3)

traiNData_OD[traiNData_OD["is_outlier"]]


#####################################################################################################
####
#FEATURE SELECTION
#Ensemble feature selection (Random forst classifier and Gradient Boosting classifier)
RF = RandomForestClassifier(n_estimators=10, criterion='entropy', max_features='auto', bootstrap=True)
GB = GradientBoostingClassifier(loss='deviance', learning_rate=0.1, n_estimators=200, max_features='auto')
ET = ExtraTreesClassifier(n_estimators=10, criterion='gini', max_features='auto', bootstrap=False)

y_train = train_data_Y.loc[:].ravel()
x_train = train_data_X.values
x_test = test_data_X.values


RF.fit(train_data_X, train_data_Y)
RF_feature = RF.feature_importances_
RF_feature
rf_score = RF.score(test_data_X, test_data_Y)
print("RandomForestClassifier score is:", rf_score)


GB.fit(train_data_X, train_data_Y)
GB_feature = GB.feature_importances_
GB_feature
gb_score = GB.score(test_data_X, test_data_Y)
print("GradientBoostingClassifier score is:", gb_score)

ET.fit(train_data_X, train_data_Y)
ET_feature = ET.feature_importances_
ET_feature
et_score = ET.score(test_data_X, test_data_Y)
print("ExtraTreeClassifier score is:", et_score)
```

```
#Representing how features affect each other via ensembling
cols = train_data_X.columns.values
features_effect = pd.DataFrame({'features': cols,
                        'RandomForest' : RF_feature,
                        'ExtraTree' : ET_feature,
                        'GradientBoost' : GB_feature
                       })


#Plotting graph showing the individual features impact
graph = features_effect.plot.bar(figsize = (18, 10), title = 'Feature distribution', grid=True, legend=True, fontsize = 15,
                        xticks=features_effect.index)
graph.set_xticklabels(features_effect.features, rotation = 80)



#Now selecting the top 15 features from all the ensembling outputs
random_feat = features_effect.nlargest(15, 'RandomForest')
grad_feat = features_effect.nlargest(15, 'GradientBoost')
extra_feat = features_effect.nlargest(15, 'ExtraTree')

#removing the duplicates features if any
final_features = pd.concat([extra_feat, grad_feat, random_feat])
final_features = final_features.drop_duplicates() # delete duplicate feature
selected_features = final_features['features'].values.tolist()

#Preparing the dataset that includes only selected features
X_train_ens = train_data_X[selected_features]
X_test_ens = test_data_X[selected_features]



###################################################### APPLYING ALGORITHMS ON ENSEMBLED
FEATURED DATASET ###################################
# [1] Applying DECISION TREE classification
Deci_tree = DecisionTreeClassifier()
Deci_tree.fit(X_train_ens,train_data_Y)
pred_Deci_tree = Deci_tree.predict(X_test_ens)

# [2] RANDOM FOREST CLASSIDFIER
RF.fit(X_train_ens, train_data_Y)
pred_Random_forest = RF.predict(X_test_ens)

# [3] Gradient Boosting classifier
GB.fit(X_train_ens, train_data_Y)
pred_Gradient_boost = GB.predict(X_test_ens)

# [4] Logistic Regression
model = LogisticRegression()
model.fit(X_train_ens, train_data_Y)
predicted = model.predict(X_test_ens)
matrix = confusion_matrix(test_data_Y, predicted)

#Calculating accuracy score of all models
conclusion = pd.DataFrame({'models': ["LOGISTIC REGRESSION","DECISION TREE
CLASSIFIER","RANDOMFOREST CLASSIFIER","GRADIENT BOOSTING CLASSIFIER"],
                        'accuracies': [accuracy_score(test_data_Y, predicted),accuracy_score(test_data_Y,pred_Deci_tree),
```

```
                                    accuracy_score(test_data_Y,
pred_Random_forest),accuracy_score(test_data_Y,pred_Gradient_boost)]})

print("Accuracy score:")
print(conclusion)

#### Confusion_matrix_metric
matrix1 = confusion_matrix(test_data_Y, predicted)
matrix2 = confusion_matrix(test_data_Y,pred_Deci_tree)
matrix3 = confusion_matrix(test_data_Y, pred_Random_forest)
matrix4 = confusion_matrix(test_data_Y,pred_Gradient_boost)

print("Confusion matrix score: \n")
print("1. Decision Tree \n", matrix2,"\n")
print("2. Random Forest \n", matrix3,"\n")
print("3. Decision Tree \n", matrix4,"\n")

####################################################################################################
###########3
#CROSS-VALIDATION SCORE

""" 2. PERFORMING CROSS VALIDATION FOR:
 1.RANDOM FOREST
 2.LOGISTIC REGRESSION
 3.DECISION TREE
 4.GRADIENT BOOSTING CLASSIFIER"""

#1. Random forest
rfc_eval= cross_val_score(estimator = RF, X = X_train_ens, y = train_data_Y, cv = 10)
rfc_eval.mean()

#2. logistic regression
logic_reg_eval= cross_val_score(estimator = model, X = X_train_ens, y = train_data_Y, cv = 10)
logic_reg_eval.mean()

#3. Decision Tree
Deci_tree_eval= cross_val_score(estimator = Deci_tree, X = X_train_ens, y = train_data_Y, cv = 10)
Deci_tree_eval.mean()

#4. SGDC
gb_eval= cross_val_score(estimator = GB, X = X_train_ens, y = train_data_Y, cv = 10)
gb_eval.mean()

#DISPLAYING THE IMPROVED TABLE OF ACCURACY OF ALL MODELS AFTER HYPER PARAMETER
OPTIMIZATION
Summary_table = pd.DataFrame([logic_reg_eval.mean(),Deci_tree_eval.mean(),rfc_eval.mean(),gb_eval.mean()],
                  index=['Logistic Regression', 'Decision Tree', 'Random Forest', 'GRDIENT BOOSTING'],
columns=['Accuracy Score'])


print("Summary Table for Section 1.2")
Summary_table
###########################################
END##############################################################
```