Importing necessary libraries

In [3]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

Importing training data and testing data ¶

In [4]:
```
train= pd.read_csv('train_data.csv')
test= pd.read_csv('test_data.csv')
test_hidden= pd.read_csv('test_data_hidden.csv')
```

In [5]:
```
train.head()
```

Out[5]:

	Time	V1	V2	V3	V4	V5	V6	V7	V
0	38355.0	1.043949	0.318555	1.045810	2.805989	-0.561113	-0.367956	0.032736	-0.04233
1	22555.0	-1.665159	0.808440	1.805627	1.903416	-0.821627	0.934790	-0.824802	0.97589
2	2431.0	-0.324096	0.601836	0.865329	-2.138000	0.294663	-1.251553	1.072114	-0.33489
3	86773.0	-0.258270	1.217501	-0.585348	-0.875347	1.222481	-0.311027	1.073860	-0.16140
4	127202.0	2.142162	-0.494988	-1.936511	-0.818288	-0.025213	-1.027245	-0.151627	-0.30575

5 rows × 31 columns

Checking for missing values

In [50]:
```
train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 227845 entries, 0 to 227844
Data columns (total 31 columns):
Time      227845 non-null float64
V1        227845 non-null float64
V2        227845 non-null float64
V3        227845 non-null float64
V4        227845 non-null float64
V5        227845 non-null float64
V6        227845 non-null float64
V7        227845 non-null float64
V8        227845 non-null float64
V9        227845 non-null float64
V10       227845 non-null float64
V11       227845 non-null float64
V12       227845 non-null float64
V13       227845 non-null float64
V14       227845 non-null float64
V15       227845 non-null float64
V16       227845 non-null float64
V17       227845 non-null float64
V18       227845 non-null float64
V19       227845 non-null float64
V20       227845 non-null float64
V21       227845 non-null float64
V22       227845 non-null float64
V23       227845 non-null float64
V24       227845 non-null float64
V25       227845 non-null float64
V26       227845 non-null float64
V27       227845 non-null float64
V28       227845 non-null float64
Amount    227845 non-null float64
Class     227845 non-null int64
dtypes: float64(30), int64(1)
memory usage: 53.9 MB
```

No missing values as the number of records are same in all the columns

In [51]:
```
np.unique(train['Class'])
```

Out[51]:
```
array([0, 1])
```

In [52]:
```
np.where(np.isnan(train))
```

Out[52]:
```
(array([], dtype=int64), array([], dtype=int64))
```

In [53]:
```
test.head()
```

Out[53]:

	Time	V1	V2	V3	V4	V5	V6	V7	V
0	113050.0	0.114697	0.796303	-0.149553	-0.823011	0.878763	-0.553152	0.939259	-0.10850
1	26667.0	-0.039318	0.495784	-0.810884	0.546693	1.986257	4.386342	-1.344891	-1.74373
2	159519.0	2.275706	-1.531508	-1.021969	-1.602152	-1.220329	-0.462376	-1.196485	-0.14705
3	137545.0	1.940137	-0.357671	-1.210551	0.382523	0.050823	-0.171322	-0.109124	-0.00211
4	63369.0	1.081395	-0.502615	1.075887	-0.543359	-1.472946	-1.065484	-0.443231	-0.14337

5 rows × 30 columns

In [54]:
```
test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 56962 entries, 0 to 56961
Data columns (total 30 columns):
Time      56962 non-null float64
V1        56962 non-null float64
V2        56962 non-null float64
V3        56962 non-null float64
V4        56962 non-null float64
V5        56962 non-null float64
V6        56962 non-null float64
V7        56962 non-null float64
V8        56962 non-null float64
V9        56962 non-null float64
V10       56962 non-null float64
V11       56962 non-null float64
V12       56962 non-null float64
V13       56962 non-null float64
V14       56962 non-null float64
V15       56962 non-null float64
V16       56962 non-null float64
V17       56962 non-null float64
V18       56962 non-null float64
V19       56962 non-null float64
V20       56962 non-null float64
V21       56962 non-null float64
V22       56962 non-null float64
V23       56962 non-null float64
V24       56962 non-null float64
V25       56962 non-null float64
V26       56962 non-null float64
V27       56962 non-null float64
V28       56962 non-null float64
Amount    56962 non-null float64
dtypes: float64(30)
memory usage: 13.0 MB
```

In [55]:
```
np.where(np.isnan(test))
```

Out[55]:
```
(array([], dtype=int64), array([], dtype=int64))
```

Checking mean and standard deviation

In [56]:
```
train.describe()
```

Out[56]:

	Time	V1	V2	V3	V4	\
count	227845.000000	227845.000000	227845.000000	227845.000000	227845.000000	227845.0000(
mean	94752.853076	-0.003321	-0.001652	0.001066	-0.000374	0.0008i
std	47500.410602	1.963028	1.661178	1.516107	1.415061	1.3670i
min	0.000000	-56.407510	-72.715728	-32.965346	-5.683171	-42.1478(
25%	54182.000000	-0.922851	-0.598040	-0.889246	-0.848884	-0.6908'
50%	84607.000000	0.012663	0.066665	0.182170	-0.019309	-0.0552‹
75%	139340.000000	1.314821	0.804401	1.029449	0.744822	0.6108!
max	172792.000000	2.454930	22.057729	9.382558	16.875344	34.8016(

8 rows × 31 columns

Checking relationship between "Time" and "Class" columns

In [10]:
```
from scipy.stats import chi2_contingency
cont_table= pd.crosstab(train['Time'], train['Class'])
stat, p, dof, expected= chi2_contingency(cont_table)
if p <= 0.05:
    print("Alternate Hypothesis: Time and Class have some relationship")
else:
    print("Null Hypothesis: Time and Class are independent of each other")

print("Confidence level: {}%".format((1-p)*100))
```

```
Alternate Hypothesis: Time and Class have some relationship
Confidence level: 100.0%
```

Checking relationship between "Amount" and "Class" columns

In [11]:
```
from scipy.stats import chi2_contingency
cont_table= pd.crosstab(train['Amount'], train['Class'])
stat, p, dof, expected= chi2_contingency(cont_table)
if p <= 0.05:
    print("Alternate Hypothesis: Amount and Class have some relationship")
else:
    print("Null Hypothesis: Amount and Class are independent of each other")

print("Confidence level: {}%".format((1-p)*100))
```

```
Alternate Hypothesis: Amount and Class have some relationship
Confidence level: 100.0%
```

Checking for imbalanced dataset

In [13]:
```
train['Class'].value_counts()
```

Out[13]:
```
0    227451
1       394
Name: Class, dtype: int64
```

Label '0' is much greater than label '1'. Hence this is a problem of imbalanced dataset

Separating features and labels

In [13]:
```
train_features= train.drop("Class", axis=1).values
train_labels= train.iloc[:,30].values
```

In [14]:
```
test_features= test_hidden.drop("Class", axis=1).values
test_labels= test_hidden.iloc[:,30].values
```

In [15]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test= train_test_split(train_features,
                                                   train_labels,
                                                   test_size=0.2,
                                                   random_state= None)


print("X_train=", X_train.shape)
print("X_test=", X_test.shape)
print("y_train=", y_train.shape)
print("y_test=", y_test.shape)
```

```
X_train= (182276, 30)
X_test= (45569, 30)
y_train= (182276,)
y_test= (45569,)
```

Checking how imbalanced dataset performs

In [16]:
```
from sklearn.linear_model import LogisticRegression
lr= LogisticRegression()
lr.fit(X_train, y_train)
pred_lr= lr.predict(X_test)
```

```
/opt/anaconda3/lib/python3.7/site-packages/sklearn/linear_model/logistic.py:4
33: FutureWarning: Default solver will be changed to 'lbfgs' in 0.22. Specify
a solver to silence this warning.
  FutureWarning)
```

In [17]:
```
from sklearn.metrics import confusion_matrix, classification_report
print(classification_report(y_test, pred_lr ))

from sklearn.metrics import roc_curve, auc

fpr, tpr, thresholds= roc_curve(y_test, pred_lr)
auc_score= auc(fpr, tpr)
print("AUC score:", auc_score)
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     45480
           1       0.77      0.63      0.69        89

   micro avg       1.00      1.00      1.00     45569
   macro avg       0.88      0.81      0.85     45569
weighted avg       1.00      1.00      1.00     45569

AUC score: 0.8144198462344233
```

As we can see the AUC score is much low hence we can say that the model performs poor due to imbalanced dataset. Hence we need to solve this problem using upsampling

Upsampling using SMOTE

In [19]:
```
print("Before Oversampling, count of label 0 is {}".format(sum(y_train==0)))
print("Before Oversampling, count of label 1 is {}".format(sum(y_train==1)))


from imblearn.over_sampling import SMOTE
sm= SMOTE(random_state=1)
x_train_res, y_train_res= sm.fit_sample(X_train, y_train)
print("After Oversampling, count of label 0 is {}".format(sum(y_train_res==0
)))
print("After Oversampling, count of label 1 is {}".format(sum(y_train_res==1
)))
```

```
Before Oversampling, count of label 0 is 181971
Before Oversampling, count of label 1 is 305

Using TensorFlow backend.

After Oversampling, count of label 0 is 181971
After Oversampling, count of label 1 is 181971
```

The number of labels are now same. It is done to reduce the bias of the model

In [20]:
```
lr1= LogisticRegression()
lr1.fit(x_train_res, y_train_res)
pred_lr_sm= lr1.predict(X_test)
```

```
/opt/anaconda3/lib/python3.7/site-packages/sklearn/linear_model/logistic.py:4
33: FutureWarning: Default solver will be changed to 'lbfgs' in 0.22. Specify
a solver to silence this warning.
  FutureWarning)
```

In [21]:
```
from sklearn.metrics import roc_curve, auc
print(classification_report(y_test, pred_lr_sm))

fpr_1, tpr_1, thresholds_1= roc_curve(y_test, pred_lr_sm)
auc_score_1= auc(fpr_1, tpr_1)
print("AUC score:", auc_score_1)
```

```
              precision    recall  f1-score   support

           0       1.00      0.98      0.99     45480
           1       0.10      0.87      0.17        89

   micro avg       0.98      0.98      0.98     45569
   macro avg       0.55      0.92      0.58     45569
weighted avg       1.00      0.98      0.99     45569

AUC score: 0.9245477701026751
```

Model performs better after upsampling

Naive Bayes Classifier

In [26]:
```
from sklearn.naive_bayes import GaussianNB
gnb= GaussianNB()
gnb.fit(x_train_res, y_train_res)
pred_nb= gnb.predict(X_test)
```

In [27]:
```
from sklearn.metrics import f1_score
print(f1_score(y_test, pred_nb))
```

```
0.2966101694915254
```

In [34]:
```
pred_nb_test= gnb.predict(test_features)
print(f1_score(test_labels, pred_nb_test))
```

```
0.2404006677796327
```

Random Forest

In [36]:
```
from sklearn.ensemble import RandomForestClassifier
rf1= RandomForestClassifier()
rf1.fit(x_train_res, y_train_res)
pred_rf= rf1.predict(X_test)
```

```
/opt/anaconda3/lib/python3.7/site-packages/sklearn/ensemble/forest.py:246: Fu
tureWarning: The default value of n_estimators will change from 10 in version
0.20 to 100 in 0.22.
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)
```

In [37]:
```
print(f1_score(y_test, pred_rf))
```

```
0.880952380952381
```

In [38]:
```
pred_rf_test= rf1.predict(test_features)
print(f1_score(test_labels, pred_rf_test))
```

```
0.8324324324324324
```

In [26]:
```
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import KFold

scoring={'r2': 'r2',
        'MSE': 'neg_mean_squared_error'}

params={'n_estimators': [400, 500, 600],
        'max_depth': [2,4,6],
        'max_features': ['sqrt', 'log2'],
        'min_samples_split': [.05]
       }
kf= KFold(n_splits= 10, random_state= 20)
gs1= GridSearchCV(estimator= rf1,
                param_grid= params,
                refit='MSE',
                cv= kf,
                scoring= scoring,
                verbose= 2)
```

In [27]:
```
gs1.fit(x_train_res[:500], y_train_res[:500])
```

```
Fitting 10 folds for each of 18 candidates, totalling 180 fits
[CV] max_depth=2, max_features=sqrt, min_samples_split=0.05, n_estimators=400
```

```
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent worke
rs.
```

```
[CV]  max_depth=2, max_features=sqrt, min_samples_split=0.05, n_estimators=40
0, total=   0.2s
[CV] max_depth=2, max_features=sqrt, min_samples_split=0.05, n_estimators=400
```

```
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.3s remaining:    0.
0s
```

```
[CV]  max_depth=2, max_features=sqrt, min_samples_split=0.05, n_estimators=40
0, total=   0.2s
[CV] max_depth=2, max_features=sqrt, min_samples_split=0.05, n_estimators=400
[CV]  max_depth=2, max_features=sqrt, min_samples_split=0.05, n_estimators=40
0, total=   0.2s
[CV] max_depth=2, max_features=sqrt, min_samples_split=0.05, n_estimators=400
[CV]  max_depth=2, max_features=sqrt, min_samples_split=0.05, n_estimators=40
0, total=   0.2s
[CV] max_depth=2, max_features=sqrt, min_samples_split=0.05, n_estimators=400
[CV]  max_depth=2, max_features=sqrt, min_samples_split=0.05, n_estimators=40
0, total=   0.2s
[CV] max_depth=2, max_features=sqrt, min_samples_split=0.05, n_estimators=400
[CV]  max_depth=2, max_features=sqrt, min_samples_split=0.05, n_estimators=40
0, total=   0.2s
[CV] max_depth=2, max_features=sqrt, min_samples_split=0.05, n_estimators=400
[CV]  max_depth=2, max_features=sqrt, min_samples_split=0.05, n_estimators=40
0, total=   0.2s
[CV] max_depth=2, max_features=sqrt, min_samples_split=0.05, n_estimators=400
[CV]  max_depth=2, max_features=sqrt, min_samples_split=0.05, n_estimators=40
0, total=   0.2s
[CV] max_depth=2, max_features=sqrt, min_samples_split=0.05, n_estimators=400
[CV]  max_depth=2, max_features=sqrt, min_samples_split=0.05, n_estimators=40
0, total=   0.2s
[CV] max_depth=2, max_features=sqrt, min_samples_split=0.05, n_estimators=500
[CV]  max_depth=2, max_features=sqrt, min_samples_split=0.05, n_estimators=50
0, total=   0.3s
[CV] max_depth=2, max_features=sqrt, min_samples_split=0.05, n_estimators=500
[CV]  max_depth=2, max_features=sqrt, min_samples_split=0.05, n_estimators=50
0, total=   0.3s
[CV] max_depth=2, max_features=sqrt, min_samples_split=0.05, n_estimators=500
[CV]  max_depth=2, max_features=sqrt, min_samples_split=0.05, n_estimators=50
0, total=   0.3s
[CV] max_depth=2, max_features=sqrt, min_samples_split=0.05, n_estimators=500
[CV]  max_depth=2, max_features=sqrt, min_samples_split=0.05, n_estimators=50
0, total=   0.3s
[CV] max_depth=2, max_features=sqrt, min_samples_split=0.05, n_estimators=500
[CV]  max_depth=2, max_features=sqrt, min_samples_split=0.05, n_estimators=50
0, total=   0.3s
[CV] max_depth=2, max_features=sqrt, min_samples_split=0.05, n_estimators=500
[CV]  max_depth=2, max_features=sqrt, min_samples_split=0.05, n_estimators=50
0, total=   0.3s
[CV] max_depth=2, max_features=sqrt, min_samples_split=0.05, n_estimators=500
[CV]  max_depth=2, max_features=sqrt, min_samples_split=0.05, n_estimators=50
0, total=   0.3s
[CV] max_depth=2, max_features=sqrt, min_samples_split=0.05, n_estimators=500
[CV]  max_depth=2, max_features=sqrt, min_samples_split=0.05, n_estimators=50
0, total=   0.3s
[CV] max_depth=2, max_features=sqrt, min_samples_split=0.05, n_estimators=500
[CV]  max_depth=2, max_features=sqrt, min_samples_split=0.05, n_estimators=50
0, total=   0.3s
[CV] max_depth=2, max_features=sqrt, min_samples_split=0.05, n_estimators=500
[CV]  max_depth=2, max_features=sqrt, min_samples_split=0.05, n_estimators=50
0, total=   0.3s
[CV] max_depth=2, max_features=sqrt, min_samples_split=0.05, n_estimators=600
```

```
[CV]  max_depth=2, max_features=sqrt, min_samples_split=0.05, n_estimators=60
0, total=   0.3s
[CV] max_depth=2, max_features=sqrt, min_samples_split=0.05, n_estimators=600
[CV]  max_depth=2, max_features=sqrt, min_samples_split=0.05, n_estimators=60
0, total=   0.3s
[CV] max_depth=2, max_features=sqrt, min_samples_split=0.05, n_estimators=600
[CV]  max_depth=2, max_features=sqrt, min_samples_split=0.05, n_estimators=60
0, total=   0.3s
[CV] max_depth=2, max_features=sqrt, min_samples_split=0.05, n_estimators=600
[CV]  max_depth=2, max_features=sqrt, min_samples_split=0.05, n_estimators=60
0, total=   0.3s
[CV] max_depth=2, max_features=sqrt, min_samples_split=0.05, n_estimators=600
[CV]  max_depth=2, max_features=sqrt, min_samples_split=0.05, n_estimators=60
0, total=   0.3s
[CV] max_depth=2, max_features=sqrt, min_samples_split=0.05, n_estimators=600
[CV]  max_depth=2, max_features=sqrt, min_samples_split=0.05, n_estimators=60
0, total=   0.3s
[CV] max_depth=2, max_features=sqrt, min_samples_split=0.05, n_estimators=600
[CV]  max_depth=2, max_features=sqrt, min_samples_split=0.05, n_estimators=60
0, total=   0.3s
[CV] max_depth=2, max_features=sqrt, min_samples_split=0.05, n_estimators=600
[CV]  max_depth=2, max_features=sqrt, min_samples_split=0.05, n_estimators=60
0, total=   0.3s
[CV] max_depth=2, max_features=sqrt, min_samples_split=0.05, n_estimators=600
[CV]  max_depth=2, max_features=sqrt, min_samples_split=0.05, n_estimators=60
0, total=   0.3s
[CV] max_depth=2, max_features=log2, min_samples_split=0.05, n_estimators=400
[CV]  max_depth=2, max_features=log2, min_samples_split=0.05, n_estimators=40
0, total=   0.2s
[CV] max_depth=2, max_features=log2, min_samples_split=0.05, n_estimators=400
[CV]  max_depth=2, max_features=log2, min_samples_split=0.05, n_estimators=40
0, total=   0.2s
[CV] max_depth=2, max_features=log2, min_samples_split=0.05, n_estimators=400
[CV]  max_depth=2, max_features=log2, min_samples_split=0.05, n_estimators=40
0, total=   0.2s
[CV] max_depth=2, max_features=log2, min_samples_split=0.05, n_estimators=400
[CV]  max_depth=2, max_features=log2, min_samples_split=0.05, n_estimators=40
0, total=   0.2s
[CV] max_depth=2, max_features=log2, min_samples_split=0.05, n_estimators=400
[CV]  max_depth=2, max_features=log2, min_samples_split=0.05, n_estimators=40
0, total=   0.2s
[CV] max_depth=2, max_features=log2, min_samples_split=0.05, n_estimators=400
[CV]  max_depth=2, max_features=log2, min_samples_split=0.05, n_estimators=40
0, total=   0.2s
[CV] max_depth=2, max_features=log2, min_samples_split=0.05, n_estimators=400
[CV]  max_depth=2, max_features=log2, min_samples_split=0.05, n_estimators=40
0, total=   0.2s
[CV] max_depth=2, max_features=log2, min_samples_split=0.05, n_estimators=400
[CV]  max_depth=2, max_features=log2, min_samples_split=0.05, n_estimators=40
0, total=   0.2s
[CV] max_depth=2, max_features=log2, min_samples_split=0.05, n_estimators=400
[CV]  max_depth=2, max_features=log2, min_samples_split=0.05, n_estimators=40
0, total=   0.2s
[CV] max_depth=2, max_features=log2, min_samples_split=0.05, n_estimators=400
```

```
[CV]  max_depth=2, max_features=log2, min_samples_split=0.05, n_estimators=40
0, total=   0.2s
[CV] max_depth=2, max_features=log2, min_samples_split=0.05, n_estimators=500
[CV]  max_depth=2, max_features=log2, min_samples_split=0.05, n_estimators=50
0, total=   0.3s
[CV] max_depth=2, max_features=log2, min_samples_split=0.05, n_estimators=500
[CV]  max_depth=2, max_features=log2, min_samples_split=0.05, n_estimators=50
0, total=   0.3s
[CV] max_depth=2, max_features=log2, min_samples_split=0.05, n_estimators=500
[CV]  max_depth=2, max_features=log2, min_samples_split=0.05, n_estimators=50
0, total=   0.3s
[CV] max_depth=2, max_features=log2, min_samples_split=0.05, n_estimators=500
[CV]  max_depth=2, max_features=log2, min_samples_split=0.05, n_estimators=50
0, total=   0.3s
[CV] max_depth=2, max_features=log2, min_samples_split=0.05, n_estimators=500
[CV]  max_depth=2, max_features=log2, min_samples_split=0.05, n_estimators=50
0, total=   0.3s
[CV] max_depth=2, max_features=log2, min_samples_split=0.05, n_estimators=500
[CV]  max_depth=2, max_features=log2, min_samples_split=0.05, n_estimators=50
0, total=   0.3s
[CV] max_depth=2, max_features=log2, min_samples_split=0.05, n_estimators=500
[CV]  max_depth=2, max_features=log2, min_samples_split=0.05, n_estimators=50
0, total=   0.3s
[CV] max_depth=2, max_features=log2, min_samples_split=0.05, n_estimators=500
[CV]  max_depth=2, max_features=log2, min_samples_split=0.05, n_estimators=50
0, total=   0.3s
[CV] max_depth=2, max_features=log2, min_samples_split=0.05, n_estimators=500
[CV]  max_depth=2, max_features=log2, min_samples_split=0.05, n_estimators=50
0, total=   0.3s
[CV] max_depth=2, max_features=log2, min_samples_split=0.05, n_estimators=600
[CV]  max_depth=2, max_features=log2, min_samples_split=0.05, n_estimators=60
0, total=   0.3s
[CV] max_depth=2, max_features=log2, min_samples_split=0.05, n_estimators=600
[CV]  max_depth=2, max_features=log2, min_samples_split=0.05, n_estimators=60
0, total=   0.4s
[CV] max_depth=2, max_features=log2, min_samples_split=0.05, n_estimators=600
[CV]  max_depth=2, max_features=log2, min_samples_split=0.05, n_estimators=60
0, total=   0.3s
[CV] max_depth=2, max_features=log2, min_samples_split=0.05, n_estimators=600
[CV]  max_depth=2, max_features=log2, min_samples_split=0.05, n_estimators=60
0, total=   0.3s
[CV] max_depth=2, max_features=log2, min_samples_split=0.05, n_estimators=600
[CV]  max_depth=2, max_features=log2, min_samples_split=0.05, n_estimators=60
0, total=   0.3s
[CV] max_depth=2, max_features=log2, min_samples_split=0.05, n_estimators=600
[CV]  max_depth=2, max_features=log2, min_samples_split=0.05, n_estimators=60
0, total=   0.3s
[CV] max_depth=2, max_features=log2, min_samples_split=0.05, n_estimators=600
[CV]  max_depth=2, max_features=log2, min_samples_split=0.05, n_estimators=60
0, total=   0.3s
[CV] max_depth=2, max_features=log2, min_samples_split=0.05, n_estimators=600
[CV]  max_depth=2, max_features=log2, min_samples_split=0.05, n_estimators=60
0, total=   0.3s
[CV] max_depth=2, max_features=log2, min_samples_split=0.05, n_estimators=600
```

```
[CV]  max_depth=2, max_features=log2, min_samples_split=0.05, n_estimators=60
0, total=   0.3s
[CV] max_depth=2, max_features=log2, min_samples_split=0.05, n_estimators=600
[CV]  max_depth=2, max_features=log2, min_samples_split=0.05, n_estimators=60
0, total=   0.3s
[CV] max_depth=4, max_features=sqrt, min_samples_split=0.05, n_estimators=400
[CV]  max_depth=4, max_features=sqrt, min_samples_split=0.05, n_estimators=40
0, total=   0.2s
[CV] max_depth=4, max_features=sqrt, min_samples_split=0.05, n_estimators=400
[CV]  max_depth=4, max_features=sqrt, min_samples_split=0.05, n_estimators=40
0, total=   0.2s
[CV] max_depth=4, max_features=sqrt, min_samples_split=0.05, n_estimators=400
[CV]  max_depth=4, max_features=sqrt, min_samples_split=0.05, n_estimators=40
0, total=   0.2s
[CV] max_depth=4, max_features=sqrt, min_samples_split=0.05, n_estimators=400
[CV]  max_depth=4, max_features=sqrt, min_samples_split=0.05, n_estimators=40
0, total=   0.2s
[CV] max_depth=4, max_features=sqrt, min_samples_split=0.05, n_estimators=400
[CV]  max_depth=4, max_features=sqrt, min_samples_split=0.05, n_estimators=40
0, total=   0.2s
[CV] max_depth=4, max_features=sqrt, min_samples_split=0.05, n_estimators=400
[CV]  max_depth=4, max_features=sqrt, min_samples_split=0.05, n_estimators=40
0, total=   0.2s
[CV] max_depth=4, max_features=sqrt, min_samples_split=0.05, n_estimators=400
[CV]  max_depth=4, max_features=sqrt, min_samples_split=0.05, n_estimators=40
0, total=   0.2s
[CV] max_depth=4, max_features=sqrt, min_samples_split=0.05, n_estimators=400
[CV]  max_depth=4, max_features=sqrt, min_samples_split=0.05, n_estimators=40
0, total=   0.2s
[CV] max_depth=4, max_features=sqrt, min_samples_split=0.05, n_estimators=400
[CV]  max_depth=4, max_features=sqrt, min_samples_split=0.05, n_estimators=40
0, total=   0.2s
[CV] max_depth=4, max_features=sqrt, min_samples_split=0.05, n_estimators=400
[CV]  max_depth=4, max_features=sqrt, min_samples_split=0.05, n_estimators=40
0, total=   0.2s
[CV] max_depth=4, max_features=sqrt, min_samples_split=0.05, n_estimators=500
[CV]  max_depth=4, max_features=sqrt, min_samples_split=0.05, n_estimators=50
0, total=   0.3s
[CV] max_depth=4, max_features=sqrt, min_samples_split=0.05, n_estimators=500
[CV]  max_depth=4, max_features=sqrt, min_samples_split=0.05, n_estimators=50
0, total=   0.3s
[CV] max_depth=4, max_features=sqrt, min_samples_split=0.05, n_estimators=500
[CV]  max_depth=4, max_features=sqrt, min_samples_split=0.05, n_estimators=50
0, total=   0.3s
[CV] max_depth=4, max_features=sqrt, min_samples_split=0.05, n_estimators=500
[CV]  max_depth=4, max_features=sqrt, min_samples_split=0.05, n_estimators=50
0, total=   0.3s
[CV] max_depth=4, max_features=sqrt, min_samples_split=0.05, n_estimators=500
[CV]  max_depth=4, max_features=sqrt, min_samples_split=0.05, n_estimators=50
0, total=   0.3s
[CV] max_depth=4, max_features=sqrt, min_samples_split=0.05, n_estimators=500
[CV]  max_depth=4, max_features=sqrt, min_samples_split=0.05, n_estimators=50
0, total=   0.3s
[CV] max_depth=4, max_features=sqrt, min_samples_split=0.05, n_estimators=500
[CV]  max_depth=4, max_features=sqrt, min_samples_split=0.05, n_estimators=50
0, total=   0.3s
[CV] max_depth=4, max_features=sqrt, min_samples_split=0.05, n_estimators=500
```

```
[CV]  max_depth=4, max_features=sqrt, min_samples_split=0.05, n_estimators=50
0, total=   0.3s
[CV] max_depth=4, max_features=sqrt, min_samples_split=0.05, n_estimators=500
[CV]  max_depth=4, max_features=sqrt, min_samples_split=0.05, n_estimators=50
0, total=   0.3s
[CV] max_depth=4, max_features=sqrt, min_samples_split=0.05, n_estimators=500
[CV]  max_depth=4, max_features=sqrt, min_samples_split=0.05, n_estimators=50
0, total=   0.3s
[CV] max_depth=4, max_features=sqrt, min_samples_split=0.05, n_estimators=600
[CV]  max_depth=4, max_features=sqrt, min_samples_split=0.05, n_estimators=60
0, total=   0.3s
[CV] max_depth=4, max_features=sqrt, min_samples_split=0.05, n_estimators=600
[CV]  max_depth=4, max_features=sqrt, min_samples_split=0.05, n_estimators=60
0, total=   0.3s
[CV] max_depth=4, max_features=sqrt, min_samples_split=0.05, n_estimators=600
[CV]  max_depth=4, max_features=sqrt, min_samples_split=0.05, n_estimators=60
0, total=   0.3s
[CV] max_depth=4, max_features=sqrt, min_samples_split=0.05, n_estimators=600
[CV]  max_depth=4, max_features=sqrt, min_samples_split=0.05, n_estimators=60
0, total=   0.3s
[CV] max_depth=4, max_features=sqrt, min_samples_split=0.05, n_estimators=600
[CV]  max_depth=4, max_features=sqrt, min_samples_split=0.05, n_estimators=60
0, total=   0.4s
[CV] max_depth=4, max_features=sqrt, min_samples_split=0.05, n_estimators=600
[CV]  max_depth=4, max_features=sqrt, min_samples_split=0.05, n_estimators=60
0, total=   0.3s
[CV] max_depth=4, max_features=sqrt, min_samples_split=0.05, n_estimators=600
[CV]  max_depth=4, max_features=sqrt, min_samples_split=0.05, n_estimators=60
0, total=   0.3s
[CV] max_depth=4, max_features=sqrt, min_samples_split=0.05, n_estimators=600
[CV]  max_depth=4, max_features=sqrt, min_samples_split=0.05, n_estimators=60
0, total=   0.3s
[CV] max_depth=4, max_features=sqrt, min_samples_split=0.05, n_estimators=600
[CV]  max_depth=4, max_features=sqrt, min_samples_split=0.05, n_estimators=60
0, total=   0.3s
[CV] max_depth=4, max_features=sqrt, min_samples_split=0.05, n_estimators=600
[CV]  max_depth=4, max_features=sqrt, min_samples_split=0.05, n_estimators=60
0, total=   0.3s
[CV] max_depth=4, max_features=log2, min_samples_split=0.05, n_estimators=400
[CV]  max_depth=4, max_features=log2, min_samples_split=0.05, n_estimators=40
0, total=   0.2s
[CV] max_depth=4, max_features=log2, min_samples_split=0.05, n_estimators=400
[CV]  max_depth=4, max_features=log2, min_samples_split=0.05, n_estimators=40
0, total=   0.2s
[CV] max_depth=4, max_features=log2, min_samples_split=0.05, n_estimators=400
[CV]  max_depth=4, max_features=log2, min_samples_split=0.05, n_estimators=40
0, total=   0.2s
[CV] max_depth=4, max_features=log2, min_samples_split=0.05, n_estimators=400
[CV]  max_depth=4, max_features=log2, min_samples_split=0.05, n_estimators=40
0, total=   0.2s
[CV] max_depth=4, max_features=log2, min_samples_split=0.05, n_estimators=400
[CV]  max_depth=4, max_features=log2, min_samples_split=0.05, n_estimators=40
0, total=   0.2s
[CV] max_depth=4, max_features=log2, min_samples_split=0.05, n_estimators=400
[CV]  max_depth=4, max_features=log2, min_samples_split=0.05, n_estimators=40
0, total=   0.2s
[CV] max_depth=4, max_features=log2, min_samples_split=0.05, n_estimators=400
```

```
[CV]  max_depth=4, max_features=log2, min_samples_split=0.05, n_estimators=40
0, total=   0.2s
[CV] max_depth=4, max_features=log2, min_samples_split=0.05, n_estimators=400
[CV]  max_depth=4, max_features=log2, min_samples_split=0.05, n_estimators=40
0, total=   0.2s
[CV] max_depth=4, max_features=log2, min_samples_split=0.05, n_estimators=400
[CV]  max_depth=4, max_features=log2, min_samples_split=0.05, n_estimators=40
0, total=   0.2s
[CV] max_depth=4, max_features=log2, min_samples_split=0.05, n_estimators=400
[CV]  max_depth=4, max_features=log2, min_samples_split=0.05, n_estimators=40
0, total=   0.2s
[CV] max_depth=4, max_features=log2, min_samples_split=0.05, n_estimators=500
[CV]  max_depth=4, max_features=log2, min_samples_split=0.05, n_estimators=50
0, total=   0.3s
[CV] max_depth=4, max_features=log2, min_samples_split=0.05, n_estimators=500
[CV]  max_depth=4, max_features=log2, min_samples_split=0.05, n_estimators=50
0, total=   0.3s
[CV] max_depth=4, max_features=log2, min_samples_split=0.05, n_estimators=500
[CV]  max_depth=4, max_features=log2, min_samples_split=0.05, n_estimators=50
0, total=   0.3s
[CV] max_depth=4, max_features=log2, min_samples_split=0.05, n_estimators=500
[CV]  max_depth=4, max_features=log2, min_samples_split=0.05, n_estimators=50
0, total=   0.3s
[CV] max_depth=4, max_features=log2, min_samples_split=0.05, n_estimators=500
[CV]  max_depth=4, max_features=log2, min_samples_split=0.05, n_estimators=50
0, total=   0.3s
[CV] max_depth=4, max_features=log2, min_samples_split=0.05, n_estimators=500
[CV]  max_depth=4, max_features=log2, min_samples_split=0.05, n_estimators=50
0, total=   0.3s
[CV] max_depth=4, max_features=log2, min_samples_split=0.05, n_estimators=500
[CV]  max_depth=4, max_features=log2, min_samples_split=0.05, n_estimators=50
0, total=   0.3s
[CV] max_depth=4, max_features=log2, min_samples_split=0.05, n_estimators=500
[CV]  max_depth=4, max_features=log2, min_samples_split=0.05, n_estimators=50
0, total=   0.3s
[CV] max_depth=4, max_features=log2, min_samples_split=0.05, n_estimators=500
[CV]  max_depth=4, max_features=log2, min_samples_split=0.05, n_estimators=50
0, total=   0.3s
[CV] max_depth=4, max_features=log2, min_samples_split=0.05, n_estimators=500
[CV]  max_depth=4, max_features=log2, min_samples_split=0.05, n_estimators=50
0, total=   0.3s
[CV] max_depth=4, max_features=log2, min_samples_split=0.05, n_estimators=600
[CV]  max_depth=4, max_features=log2, min_samples_split=0.05, n_estimators=60
0, total=   0.3s
[CV] max_depth=4, max_features=log2, min_samples_split=0.05, n_estimators=600
[CV]  max_depth=4, max_features=log2, min_samples_split=0.05, n_estimators=60
0, total=   0.3s
[CV] max_depth=4, max_features=log2, min_samples_split=0.05, n_estimators=600
[CV]  max_depth=4, max_features=log2, min_samples_split=0.05, n_estimators=60
0, total=   0.3s
[CV] max_depth=4, max_features=log2, min_samples_split=0.05, n_estimators=600
[CV]  max_depth=4, max_features=log2, min_samples_split=0.05, n_estimators=60
0, total=   0.3s
[CV] max_depth=4, max_features=log2, min_samples_split=0.05, n_estimators=600
[CV]  max_depth=4, max_features=log2, min_samples_split=0.05, n_estimators=60
0, total=   0.3s
[CV] max_depth=4, max_features=log2, min_samples_split=0.05, n_estimators=600
```

```
[CV]  max_depth=4, max_features=log2, min_samples_split=0.05, n_estimators=60
0, total=   0.3s
[CV] max_depth=4, max_features=log2, min_samples_split=0.05, n_estimators=600
[CV]  max_depth=4, max_features=log2, min_samples_split=0.05, n_estimators=60
0, total=   0.3s
[CV] max_depth=4, max_features=log2, min_samples_split=0.05, n_estimators=600
[CV]  max_depth=4, max_features=log2, min_samples_split=0.05, n_estimators=60
0, total=   0.3s
[CV] max_depth=4, max_features=log2, min_samples_split=0.05, n_estimators=600
[CV]  max_depth=4, max_features=log2, min_samples_split=0.05, n_estimators=60
0, total=   0.3s
[CV] max_depth=4, max_features=log2, min_samples_split=0.05, n_estimators=600
[CV]  max_depth=4, max_features=log2, min_samples_split=0.05, n_estimators=60
0, total=   0.3s
[CV] max_depth=6, max_features=sqrt, min_samples_split=0.05, n_estimators=400
[CV]  max_depth=6, max_features=sqrt, min_samples_split=0.05, n_estimators=40
0, total=   0.2s
[CV] max_depth=6, max_features=sqrt, min_samples_split=0.05, n_estimators=400
[CV]  max_depth=6, max_features=sqrt, min_samples_split=0.05, n_estimators=40
0, total=   0.2s
[CV] max_depth=6, max_features=sqrt, min_samples_split=0.05, n_estimators=400
[CV]  max_depth=6, max_features=sqrt, min_samples_split=0.05, n_estimators=40
0, total=   0.2s
[CV] max_depth=6, max_features=sqrt, min_samples_split=0.05, n_estimators=400
[CV]  max_depth=6, max_features=sqrt, min_samples_split=0.05, n_estimators=40
0, total=   0.2s
[CV] max_depth=6, max_features=sqrt, min_samples_split=0.05, n_estimators=400
[CV]  max_depth=6, max_features=sqrt, min_samples_split=0.05, n_estimators=40
0, total=   0.2s
[CV] max_depth=6, max_features=sqrt, min_samples_split=0.05, n_estimators=400
[CV]  max_depth=6, max_features=sqrt, min_samples_split=0.05, n_estimators=40
0, total=   0.2s
[CV] max_depth=6, max_features=sqrt, min_samples_split=0.05, n_estimators=400
[CV]  max_depth=6, max_features=sqrt, min_samples_split=0.05, n_estimators=40
0, total=   0.2s
[CV] max_depth=6, max_features=sqrt, min_samples_split=0.05, n_estimators=400
[CV]  max_depth=6, max_features=sqrt, min_samples_split=0.05, n_estimators=40
0, total=   0.2s
[CV] max_depth=6, max_features=sqrt, min_samples_split=0.05, n_estimators=400
[CV]  max_depth=6, max_features=sqrt, min_samples_split=0.05, n_estimators=40
0, total=   0.2s
[CV] max_depth=6, max_features=sqrt, min_samples_split=0.05, n_estimators=400
[CV]  max_depth=6, max_features=sqrt, min_samples_split=0.05, n_estimators=40
0, total=   0.2s
[CV] max_depth=6, max_features=sqrt, min_samples_split=0.05, n_estimators=500
[CV]  max_depth=6, max_features=sqrt, min_samples_split=0.05, n_estimators=50
0, total=   0.3s
[CV] max_depth=6, max_features=sqrt, min_samples_split=0.05, n_estimators=500
[CV]  max_depth=6, max_features=sqrt, min_samples_split=0.05, n_estimators=50
0, total=   0.3s
[CV] max_depth=6, max_features=sqrt, min_samples_split=0.05, n_estimators=500
[CV]  max_depth=6, max_features=sqrt, min_samples_split=0.05, n_estimators=50
0, total=   0.3s
[CV] max_depth=6, max_features=sqrt, min_samples_split=0.05, n_estimators=500
[CV]  max_depth=6, max_features=sqrt, min_samples_split=0.05, n_estimators=50
0, total=   0.3s
[CV] max_depth=6, max_features=sqrt, min_samples_split=0.05, n_estimators=500
```

```
[CV]  max_depth=6, max_features=sqrt, min_samples_split=0.05, n_estimators=50
0, total=   0.3s
[CV] max_depth=6, max_features=sqrt, min_samples_split=0.05, n_estimators=500
[CV]  max_depth=6, max_features=sqrt, min_samples_split=0.05, n_estimators=50
0, total=   0.3s
[CV] max_depth=6, max_features=sqrt, min_samples_split=0.05, n_estimators=500
[CV]  max_depth=6, max_features=sqrt, min_samples_split=0.05, n_estimators=50
0, total=   0.3s
[CV] max_depth=6, max_features=sqrt, min_samples_split=0.05, n_estimators=500
[CV]  max_depth=6, max_features=sqrt, min_samples_split=0.05, n_estimators=50
0, total=   0.3s
[CV] max_depth=6, max_features=sqrt, min_samples_split=0.05, n_estimators=500
[CV]  max_depth=6, max_features=sqrt, min_samples_split=0.05, n_estimators=50
0, total=   0.3s
[CV] max_depth=6, max_features=sqrt, min_samples_split=0.05, n_estimators=500
[CV]  max_depth=6, max_features=sqrt, min_samples_split=0.05, n_estimators=50
0, total=   0.3s
[CV] max_depth=6, max_features=sqrt, min_samples_split=0.05, n_estimators=600
[CV]  max_depth=6, max_features=sqrt, min_samples_split=0.05, n_estimators=60
0, total=   0.3s
[CV] max_depth=6, max_features=sqrt, min_samples_split=0.05, n_estimators=600
[CV]  max_depth=6, max_features=sqrt, min_samples_split=0.05, n_estimators=60
0, total=   0.3s
[CV] max_depth=6, max_features=sqrt, min_samples_split=0.05, n_estimators=600
[CV]  max_depth=6, max_features=sqrt, min_samples_split=0.05, n_estimators=60
0, total=   0.3s
[CV] max_depth=6, max_features=sqrt, min_samples_split=0.05, n_estimators=600
[CV]  max_depth=6, max_features=sqrt, min_samples_split=0.05, n_estimators=60
0, total=   0.3s
[CV] max_depth=6, max_features=sqrt, min_samples_split=0.05, n_estimators=600
[CV]  max_depth=6, max_features=sqrt, min_samples_split=0.05, n_estimators=60
0, total=   0.3s
[CV] max_depth=6, max_features=sqrt, min_samples_split=0.05, n_estimators=600
[CV]  max_depth=6, max_features=sqrt, min_samples_split=0.05, n_estimators=60
0, total=   0.3s
[CV] max_depth=6, max_features=sqrt, min_samples_split=0.05, n_estimators=600
[CV]  max_depth=6, max_features=sqrt, min_samples_split=0.05, n_estimators=60
0, total=   0.3s
[CV] max_depth=6, max_features=sqrt, min_samples_split=0.05, n_estimators=600
[CV]  max_depth=6, max_features=sqrt, min_samples_split=0.05, n_estimators=60
0, total=   0.3s
[CV] max_depth=6, max_features=sqrt, min_samples_split=0.05, n_estimators=600
[CV]  max_depth=6, max_features=sqrt, min_samples_split=0.05, n_estimators=60
0, total=   0.3s
[CV] max_depth=6, max_features=sqrt, min_samples_split=0.05, n_estimators=600
[CV]  max_depth=6, max_features=sqrt, min_samples_split=0.05, n_estimators=60
0, total=   0.3s
[CV] max_depth=6, max_features=log2, min_samples_split=0.05, n_estimators=400
[CV]  max_depth=6, max_features=log2, min_samples_split=0.05, n_estimators=40
0, total=   0.2s
[CV] max_depth=6, max_features=log2, min_samples_split=0.05, n_estimators=400
[CV]  max_depth=6, max_features=log2, min_samples_split=0.05, n_estimators=40
0, total=   0.2s
[CV] max_depth=6, max_features=log2, min_samples_split=0.05, n_estimators=400
[CV]  max_depth=6, max_features=log2, min_samples_split=0.05, n_estimators=40
0, total=   0.2s
[CV] max_depth=6, max_features=log2, min_samples_split=0.05, n_estimators=400
```

```
[CV]  max_depth=6, max_features=log2, min_samples_split=0.05, n_estimators=40
0, total=   0.2s
[CV] max_depth=6, max_features=log2, min_samples_split=0.05, n_estimators=400
[CV]  max_depth=6, max_features=log2, min_samples_split=0.05, n_estimators=40
0, total=   0.2s
[CV] max_depth=6, max_features=log2, min_samples_split=0.05, n_estimators=400
[CV]  max_depth=6, max_features=log2, min_samples_split=0.05, n_estimators=40
0, total=   0.2s
[CV] max_depth=6, max_features=log2, min_samples_split=0.05, n_estimators=400
[CV]  max_depth=6, max_features=log2, min_samples_split=0.05, n_estimators=40
0, total=   0.2s
[CV] max_depth=6, max_features=log2, min_samples_split=0.05, n_estimators=400
[CV]  max_depth=6, max_features=log2, min_samples_split=0.05, n_estimators=40
0, total=   0.2s
[CV] max_depth=6, max_features=log2, min_samples_split=0.05, n_estimators=400
[CV]  max_depth=6, max_features=log2, min_samples_split=0.05, n_estimators=40
0, total=   0.2s
[CV] max_depth=6, max_features=log2, min_samples_split=0.05, n_estimators=400
[CV]  max_depth=6, max_features=log2, min_samples_split=0.05, n_estimators=40
0, total=   0.2s
[CV] max_depth=6, max_features=log2, min_samples_split=0.05, n_estimators=500
[CV]  max_depth=6, max_features=log2, min_samples_split=0.05, n_estimators=50
0, total=   0.3s
[CV] max_depth=6, max_features=log2, min_samples_split=0.05, n_estimators=500
[CV]  max_depth=6, max_features=log2, min_samples_split=0.05, n_estimators=50
0, total=   0.3s
[CV] max_depth=6, max_features=log2, min_samples_split=0.05, n_estimators=500
[CV]  max_depth=6, max_features=log2, min_samples_split=0.05, n_estimators=50
0, total=   0.3s
[CV] max_depth=6, max_features=log2, min_samples_split=0.05, n_estimators=500
[CV]  max_depth=6, max_features=log2, min_samples_split=0.05, n_estimators=50
0, total=   0.3s
[CV] max_depth=6, max_features=log2, min_samples_split=0.05, n_estimators=500
[CV]  max_depth=6, max_features=log2, min_samples_split=0.05, n_estimators=50
0, total=   0.3s
[CV] max_depth=6, max_features=log2, min_samples_split=0.05, n_estimators=500
[CV]  max_depth=6, max_features=log2, min_samples_split=0.05, n_estimators=50
0, total=   0.3s
[CV] max_depth=6, max_features=log2, min_samples_split=0.05, n_estimators=500
[CV]  max_depth=6, max_features=log2, min_samples_split=0.05, n_estimators=50
0, total=   0.3s
[CV] max_depth=6, max_features=log2, min_samples_split=0.05, n_estimators=500
[CV]  max_depth=6, max_features=log2, min_samples_split=0.05, n_estimators=50
0, total=   0.3s
[CV] max_depth=6, max_features=log2, min_samples_split=0.05, n_estimators=500
[CV]  max_depth=6, max_features=log2, min_samples_split=0.05, n_estimators=50
0, total=   0.3s
[CV] max_depth=6, max_features=log2, min_samples_split=0.05, n_estimators=500
[CV]  max_depth=6, max_features=log2, min_samples_split=0.05, n_estimators=50
0, total=   0.3s
[CV] max_depth=6, max_features=log2, min_samples_split=0.05, n_estimators=600
[CV]  max_depth=6, max_features=log2, min_samples_split=0.05, n_estimators=60
0, total=   0.3s
[CV] max_depth=6, max_features=log2, min_samples_split=0.05, n_estimators=600
[CV]  max_depth=6, max_features=log2, min_samples_split=0.05, n_estimators=60
0, total=   0.3s
[CV] max_depth=6, max_features=log2, min_samples_split=0.05, n_estimators=600
```

```
[CV]  max_depth=6, max_features=log2, min_samples_split=0.05, n_estimators=60
0, total=   0.3s
[CV] max_depth=6, max_features=log2, min_samples_split=0.05, n_estimators=600
[CV]  max_depth=6, max_features=log2, min_samples_split=0.05, n_estimators=60
0, total=   0.3s
[CV] max_depth=6, max_features=log2, min_samples_split=0.05, n_estimators=600
[CV]  max_depth=6, max_features=log2, min_samples_split=0.05, n_estimators=60
0, total=   0.3s
[CV] max_depth=6, max_features=log2, min_samples_split=0.05, n_estimators=600
[CV]  max_depth=6, max_features=log2, min_samples_split=0.05, n_estimators=60
0, total=   0.3s
[CV] max_depth=6, max_features=log2, min_samples_split=0.05, n_estimators=600
[CV]  max_depth=6, max_features=log2, min_samples_split=0.05, n_estimators=60
0, total=   0.3s
[CV] max_depth=6, max_features=log2, min_samples_split=0.05, n_estimators=600
[CV]  max_depth=6, max_features=log2, min_samples_split=0.05, n_estimators=60
0, total=   0.3s
[CV] max_depth=6, max_features=log2, min_samples_split=0.05, n_estimators=600
[CV]  max_depth=6, max_features=log2, min_samples_split=0.05, n_estimators=60
0, total=   0.3s
[CV] max_depth=6, max_features=log2, min_samples_split=0.05, n_estimators=600
[CV]  max_depth=6, max_features=log2, min_samples_split=0.05, n_estimators=60
0, total=   0.3s
```

```
[Parallel(n_jobs=1)]: Done 180 out of 180 | elapsed:   55.0s finished
```

Out[27]:

```
GridSearchCV(cv=KFold(n_splits=10, random_state=20, shuffle=False),
       error_score='raise-deprecating',
       estimator=RandomForestClassifier(bootstrap=True, class_weight=None, cr
iterion='gini',
            max_depth=None, max_features='auto', max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, n_estimators=10, n_jobs=None,
            oob_score=False, random_state=None, verbose=0,
            warm_start=False),
       fit_params=None, iid='warn', n_jobs=None,
       param_grid={'n_estimators': [400, 500, 600], 'max_depth': [2, 4, 6],
'max_features': ['sqrt', 'log2'], 'min_samples_split': [0.05]},
       pre_dispatch='2*n_jobs', refit='MSE', return_train_score='warn',
       scoring={'r2': 'r2', 'MSE': 'neg_mean_squared_error'}, verbose=2)
```

In [28]:

```
gs1.best_params_
```

Out[28]:

```
{'max_depth': 2,
 'max_features': 'sqrt',
 'min_samples_split': 0.05,
 'n_estimators': 400}
```

In [29]:
```
rf2= RandomForestClassifier(max_depth= 2,
                             max_features= 'sqrt',
                             min_samples_split= 0.05,
                            n_estimators=500
                            )
rf2.fit(x_train_res, y_train_res)
pred_rf_gs= rf2.predict(X_test)
```

In [30]:
```
print(f1_score(y_test, pred_rf_gs))
```

```
0.4949152542372882
```

Ada boost

In [40]:
```
from sklearn.ensemble import AdaBoostClassifier
ada= AdaBoostClassifier()
ada.fit(x_train_res, y_train_res)
predictions_ada= ada.predict(X_test)
```

In [41]:
```
print(f1_score(y_test, predictions_ada))
```

```
0.2851851851851852
```

In [42]:
```
pred_ada_test= ada.predict(test_features)
print(f1_score(test_labels, pred_ada_test))
```

```
0.23680456490727533
```

Gradient Boosting

In [43]:
```
from sklearn.ensemble import GradientBoostingClassifier
gb= GradientBoostingClassifier()
gb.fit(x_train_res, y_train_res)
predictions_gb= gb.predict(X_test)
```

In [44]:
```
print(f1_score(y_test, predictions_gb))
```

```
0.45161290322580655
```

In [45]:
```
pred_gb_test= gb.predict(test_features)
print(f1_score(test_labels, pred_gb_test))
```

```
0.36888888888888893
```

Artificial Neural Network

In [31]:
```
import sys


class NeuralNetMLP(object):

    def __init__(self, n_hidden=30,
                 l2=0., epochs=100, eta=0.001,
                 shuffle=True, minibatch_size=1, seed=None):

        self.random = np.random.RandomState(seed)
        self.n_hidden = n_hidden
        self.l2 = l2
        self.epochs = epochs
        self.eta = eta
        self.shuffle = shuffle
        self.minibatch_size = minibatch_size


    def onehot(self, y, n_classes):
        """Encode labels into one-hot representation"""

        onehot = np.zeros((n_classes, y.shape[0]))
        for idx, val in enumerate(y.astype(int)):
            onehot[val, idx] = 1.
        return onehot.T

    def sigmoid(self, z):
        """Compute sigmoid function"""
        return 1. / (1. + np.exp(-z))

    def forward(self, X):
        """Compute forward propagation"""

        # Sum product of hidden layer
        z_h = np.dot(X, self.w_h) + self.b_h

        # activation of hidden layer
        a_h = self.sigmoid(z_h)

        # net input of output layer
        z_out = np.dot(a_h, self.w_out) + self.b_out

        # activation of output layer
        a_out = self.sigmoid(z_out)

        return z_h, a_h, z_out, a_out

    def compute_cost(self, y_enc, output):
        """Compute cost function """

        L2_term = (self.l2 *
                   (np.sum(self.w_h ** 2.) +
                    np.sum(self.w_out ** 2.)))

        term1 = -y_enc * (np.log(output))
        term2 = (1. - y_enc) * np.log(1. - output)
```

```
        cost = np.sum(term1 - term2) + L2_term
        return cost

    def predict(self, X):
        """Predict class labels"""

        z_h, a_h, z_out, a_out = self.forward(X)
        y_pred = np.argmax(z_out, axis=1)
        return y_pred

    def fit(self, X_train, y_train, X_valid, y_valid):
        """ Learn weights from training data"""

        n_output = np.unique(y_train).shape[0]  # number of class labels
        n_features = X_train.shape[1]

        ########################
        # Weight initialization
        ########################

        # weights for input  (hidden)
        self.b_h = np.zeros(self.n_hidden)
        self.w_h = self.random.normal(loc=0.0, scale=0.1,
                                      size=(n_features, self.n_hidden))

        # weights for hidden  (output)
        self.b_out = np.zeros(n_output)
        self.w_out = self.random.normal(loc=0.0, scale=0.1,
                                        size=(self.n_hidden, n_output))

        epoch_strlen = len(str(self.epochs))
        self.eval_ = {'cost': [], 'train_acc': [], 'valid_acc': []}

        y_train_enc = self.onehot(y_train, n_output)

        # iterate over training epochs
        for i in range(self.epochs):

            # iterate over minibatches
            indices = np.arange(X_train.shape[0])

            if self.shuffle:
                self.random.shuffle(indices)

            for start_idx in range(0, indices.shape[0] - self.minibatch_size +
                                   1, self.minibatch_size):
                batch_idx = indices[start_idx:start_idx + self.minibatch_size]

                # forward propagation
                z_h, a_h, z_out, a_out = self.forward(X_train[batch_idx])

                ##################
                # Backpropagation
                ##################


                sigma_out = a_out - y_train_enc[batch_idx]
```

```
                sigmoid_derivative_h = a_h * (1. - a_h)


                sigma_h = (np.dot(sigma_out, self.w_out.T) *
                           sigmoid_derivative_h)


                grad_w_h = np.dot(X_train[batch_idx].T, sigma_h)
                grad_b_h = np.sum(sigma_h, axis=0)


                grad_w_out = np.dot(a_h.T, sigma_out)
                grad_b_out = np.sum(sigma_out, axis=0)

                # Regularization and weight updates
                delta_w_h = (grad_w_h + self.l2*self.w_h)
                delta_b_h = grad_b_h # bias is not regularized
                self.w_h -= self.eta * delta_w_h
                self.b_h -= self.eta * delta_b_h

                delta_w_out = (grad_w_out + self.l2*self.w_out)
                delta_b_out = grad_b_out  # bias is not regularized
                self.w_out -= self.eta * delta_w_out
                self.b_out -= self.eta * delta_b_out

            #############
            # Evaluation
            #############

            # Evaluation after each epoch during training
            z_h, a_h, z_out, a_out = self.forward(X_train)

            cost = self.compute_cost(y_enc=y_train_enc,
                                     output=a_out)

            y_train_pred = self.predict(X_train)
            y_valid_pred = self.predict(X_valid)

            train_acc = ((np.sum(y_train == y_train_pred)).astype(np.float) /
                         X_train.shape[0])
            valid_acc = ((np.sum(y_valid == y_valid_pred)).astype(np.float) /
                         X_valid.shape[0])

            sys.stderr.write('\r%0*d/%d | Cost: %.2f '
                             '| Train/Valid Acc.: %.2f%%/%.2f%% ' %
                             (epoch_strlen, i+1, self.epochs, cost,
                              train_acc*100, valid_acc*100))
            sys.stderr.flush()

            self.eval_['cost'].append(cost)
            self.eval_['train_acc'].append(train_acc)
            self.eval_['valid_acc'].append(valid_acc)

        return self
```

In [32]:
```
n_epochs= 100
nn = NeuralNetMLP(n_hidden=100,
                  l2=0.01,
                  epochs=n_epochs,
                  eta=0.0005,
                  minibatch_size=100,
                  shuffle=True,
                  seed=1)

nn.fit(X_train=x_train_res[:55000],
       y_train=y_train_res[:55000],
       X_valid=test_features[55000:],
       y_valid=test_labels[55000:])
```

```
/opt/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:29: RuntimeW
arning: overflow encountered in exp
100/100 | Cost: 1360.93 | Train/Valid Acc.: 99.83%/99.90%
```

Out[32]:
```
<__main__.NeuralNetMLP at 0x7f51799d4940>
```

Neural Network gave the best results of all the above models

Checking Anomalies in the dataset

In [32]:
```
def anomaly(t):
    q1, q3= np.percentile(t, [25, 75])
    IQR= q3-q1
    LR= q1- (1.5*IQR)
    UR= q3+ (1.5*IQR)
    return LR, UR
```

In [34]:
```
lower_range, upper_range= anomaly(train)
lower_range, upper_range
```

Out[34]:
```
(-1.773619854710322, 1.9939057538689218)
```

In [36]:
```
train[(train['Class'] > upper_range) & (train['Class'] < lower_range)]
```

Out[36]:

Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V23	V24	V25	V26	V27	V28	A

0 rows × 31 columns

So we can use -1.773 as the lower threshold value and 1.993 as upper threshold value for the dataset

In [ ]: