Importing Libraries

In [130]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import xgboost as xgb
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV
from imblearn.under_sampling import RandomUnderSampler
from imblearn.over_sampling import RandomOverSampler
```

Reading DataSets

In [131]:

```
Train = pd.read_csv('train.csv')
Test = pd.read_csv('test.csv')
```

Analysing DataSet

In [132]:

```
Train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 87554 entries, 0 to 87553
Columns: 188 entries, 1 to 188
dtypes: float64(188)
memory usage: 125.6 MB
```

In [133]:

```
Test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21892 entries, 0 to 21891
Columns: 188 entries, 1 to 188
dtypes: float64(188)
memory usage: 31.4 MB
```

In [134]:

```
Unique = []
Count = []
for i in range(87554):
    if Train.iloc[i, 187] not in Unique:
        Unique.append(Train.iloc[i, 187])
        Count.append(0)
for i in range(len(Unique)):
    Count[i] = list(Train.iloc[:, 187]).count(Unique[i])
Count
```

Out[134]:

```
[72471, 2223, 5788, 641, 6431]
```

Since one of the class in very much majority compared to others, lets undersample the data.

UnderSampling the training Data

In [135]:

```
x_train = Train.iloc[:,:187]
y_train = Train.iloc[:, 187]
Sampler = RandomUnderSampler(random_state = 0, replacement = False)
x_train, y_train = Sampler.fit_resample(x_train, y_train)
```

In [136]:

```
x_test = Test.iloc[:, :187]
y_test = Test.iloc[:, 187]
```

Applying XGBoost Classifier

In [137]:

```
XG = xgb.XGBClassifier(objective ='reg:squarederror', colsample_bytree = 0.5, learning_rate = 0.1,
                       max_depth = 9, alpha = 10, n_estimators = 200)
```

In [138]:

```
XG.fit(x_train, y_train)
```

Out[138]:

```
XGBClassifier(alpha=10, base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=0.5, gamma=0, gpu_id=-1,
              importance_type='gain', interaction_constraints='',
              learning_rate=0.1, max_delta_step=0, max_depth=9,
              min_child_weight=1, missing=nan, monotone_constraints='()',
              n_estimators=200, n_jobs=0, num_parallel_tree=1,
              objective='multi:softprob', random_state=0, reg_alpha=10,
              reg_lambda=1, scale_pos_weight=None, subsample=1,
              tree_method='exact', validate_parameters=1, verbosity=None)
```

In [139]:

```
XG.score(x_test,y_test)
```

Out[139]:

```
0.8320847798282478
```

XGBoost had an accuracy of 83.2 %

Applying K-Nearest Neighbors Classifier

In [140]:

```
KNN = KNeighborsClassifier(n_neighbors = 9)
```

In [141]:

```
KNN.fit(x_train, y_train)
```

Out[141]:

```
KNeighborsClassifier(n_neighbors=9)
```

In [142]:

```
y_pred = KNN.predict(x_test)
```

In [143]:

```
accuracy_score(y_test, y_pred)
```

Out[143]:

```
0.7333729216152018
```

KNN had an accuracy of 73.3 %

Applying Random Forest Classifier using Grid Search

In [144]:

```
rf = RandomForestClassifier()
```

In [154]:

```
param_grid = {
    'max_depth': [90],
    'max_features': ['auto'],
    'n_estimators': [1300, 1500],
}
```

In [155]:

```
grid_s = GridSearchCV(estimator = rf, param_grid = param_grid, cv = 4)
```

In [156]:

```
grid_s.fit(x_train, y_train)
```

Out[156]:

```
GridSearchCV(cv=4, estimator=RandomForestClassifier(),
             param_grid={'max_depth': [90], 'max_features': ['auto'],
                         'n_estimators': [1300, 2000]})
```

In [157]:

```
grid_s.best_params_
```

Out[157]:

```
{'max_depth': 90, 'max_features': 'auto', 'n_estimators': 1300}
```

In [174]:

```
rf2 = RandomForestRegressor(max_depth =  90, max_features = 'auto', n_estimators = 1600)
rf.fit(x_train, y_train)
```

Out[174]:

```
RandomForestClassifier()
```

In [175]:

```
rf.score(x_test, y_test)
```

Out[175]:

0.8664809062671296

Random Forest had highest accuracy of 86.60 %

In [ ]: