In [ ]:

```
# Importing basic Libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

In [ ]:

```
# Importing Other Libraries
import xgboost as xgb
from xgboost.sklearn import XGBClassifier,XGBRegressor
import catboost
from catboost import CatBoostClassifier
from sklearn.preprocessing import LabelEncoder , MinMaxScaler
from sklearn.cross_validation import KFold , cross_val_score
from sklearn.metrics import accuracy_score , roc_auc_score,confusion_matrix
from sklearn.grid_search import GridSearchCV
from sklearn import metrics
from sklearn.ensemble import RandomForestRegressor,RandomForestClassifier
from sklearn.neighbors import KNeighborsRegressor,KNeighborsClassifier
from sklearn.svm import SVR
from sklearn.linear_model import LogisticRegression,LinearRegression
from sklearn.ensemble import ExtraTreesClassifier
```

In [ ]:

```
# Getting the Train and Test Dataset
train_data = pd.read_csv('../../../dataset/data_supremacy/train_FG6BvLg.csv')
test_data = pd.read_csv('../../../dataset/data_supremacy/test_wovud0B.csv')
print("Train Shape : {}\nTest Shape : {}".format(train_data.shape,test_data.shape))
```

In [ ]:

```
#train_data = train_data.sample(frac=1)
train_data.head(15)
```

Splitting the City and extracting only the id of an city

In [ ]:

```
train_city = train_data.city.str.split("_")
test_city = test_data.city.str.split("_")
```

In [ ]:

```
train_data.head()
```

In [ ]:

```
i=0
for x in train_city:
    train_data.loc[i,'city'] = x[1]
    i=i+1
```

In [ ]:

```
train_data.head()
```

In [ ]:

```
i=0
for x in test_city:
    test_data.loc[i,'city'] = x[1]
    i=i+1
```

In [ ]:

```
test_data.head()
```

In [ ]:

```
train_data.gender.value_counts()
```

In [ ]:

```
train_data.isnull().sum()
```

Popping out the Target Variable from training data

In [ ]:

```
target = train_data.pop('target')
enrollee_id = test_data['enrollee_id']
```

In [ ]:

```
test_data.isnull().sum()
```

Combining both the dataset to perform preprocessing

In [ ]:

```
combined_data = train_data.append(test_data)
```

In [ ]:

```
combined_data.training_hours.describe()
```

In [ ]:

```
"""
bins = [0,20,40,60,100,150,250,350]
labels = [1,2,3,4,5,6,7]
combined_data['Training_hours'] = pd.cut(combined_data['training_hours'],bins=bins,labels=labels)
combined_data.drop('training_hours',axis=1,inplace=True)
combined_data.Training_hours = combined_data.Training_hours.astype('int')
"""
```

In [ ]:

```
combined_data.city= combined_data.city.astype('int')
```

Converting/ Replacing string type columns to Integer

In [ ]:

```
combined_data.isnull().sum()
```

In [ ]:

```
print(combined_data.gender.value_counts())
combined_data.gender.replace({'Male':2,'Female':0,'Other':1},inplace=True)
```

In [ ]:

```
print(combined_data.enrolled_university.value_counts())
combined_data.enrolled_university.replace({'no_enrollment':0,'Part time course':1,'Full time course':2},inpl
ace=True)
```

In [ ]:

```
print(combined_data.education_level.value_counts())
combined_data.education_level.replace({'Primary School':1,'High School':2,'Graduate':3,'Masters':4,'Phd':5},
inplace=True)
```

In [ ]:

```
print(combined_data.major_discipline.value_counts())
combined_data.major_discipline.replace({'No Major':1,'Other':2,'Arts':3,'Humanities':4,'Business Degree':5,'
STEM':8},inplace=True)
```

In [ ]:

```
print(combined_data.experience.value_counts())
combined_data.experience.replace({'>20':25,'<1':0},inplace=True)
```

In [ ]:

```
print(combined_data.company_size.value_counts())
combined_data.company_size.replace({'<10':1,'10/49':2,'50-99':3,'100-500':4,'500-999':5,'1000-4999':6,'5000-9999':7,'10000+':8},inplace=True)
```

In [ ]:

```
print(combined_data.company_type.value_counts())
combined_data.company_type.replace({'Other':1,'Early Stage Startup':2,'Funded Startup':3,'NGO':4,'Public Sector':5,'Pvt Ltd':6},inplace=True)
```

In [ ]:

```
print(combined_data.last_new_job.value_counts())
combined_data.last_new_job.replace({'never':0,'>4':5,'nan':1},inplace=True)
```

In [ ]:

```
print(combined_data.relevent_experience.value_counts())
combined_data.relevent_experience.replace({'No relevent experience':0,'Has relevent experience':1},inplace=True)
```

In [ ]:

```
enc = LabelEncoder()
#combined_data.city = enc.fit_transform(combined_data.city)

combined_data.education_level = enc.fit_transform(combined_data.education_level.astype(str))
combined_data.education_level.dtype
```

In [ ]:

```
combined_data.head()
```

In [ ]:

```
combined_data.dtypes
```

In [ ]:

```
combined_data.last_new_job.value_counts()
```

In [ ]:

```
# Filling nan values
combined_data.last_new_job.fillna(1,inplace=True)
```

In [ ]:

```
combined_data.last_new_job = combined_data.last_new_job.astype('int')
```

In [ ]:

```
combined_data.experience.value_counts(ascending=True)
```

In [ ]:

```
# Fill nan values
combined_data.experience.fillna(21,inplace=True)
```

In [ ]:

```
combined_data.experience = combined_data.experience.astype('int')
```

In [ ]:

```
combined_data.experience[:1000].hist()
```

In [ ]:

```
combined_data.isnull().sum()
```

In [ ]:

```
combined_data.head()
```

In [ ]:

```
combined_data.isnull().sum()
```

In [ ]:

```
combined_data.enrolled_university.value_counts()
```

Filling all the nan values remaining with zero

In [ ]:

```
combined_data.major_discipline.fillna(0,inplace=True)
```

In [ ]:

```
combined_data.company_type.fillna(0,inplace=True)
```

In [ ]:

```
combined_data.company_size.fillna(0,inplace=True)
```

In [ ]:

```
combined_data.enrolled_university.fillna(0,inplace=True)
```

In [ ]:

```
combined_data.education_level.fillna(0,inplace=True)
```

In [ ]:

```
combined_data.isnull().sum()
```

In [ ]:

```
combined_data.drop('enrollee_id',axis=1,inplace=True)
```

In [ ]:

```
combined_data.isnull().sum()
```

In [ ]:

```
"""
bins = [0.4,0.5,0.6,0.7,0.8,0.9,1]
labels = [0,1,2,3,4,5]
combined_data['CityDI'] = pd.cut(combined_data['city_development_index'],bins=bins,labels=labels)
combined_data.drop('city_development_index',axis=1,inplace=True)
combined_data.CityDI = combined_data.CityDI.astype('int')
"""
```

In [ ]:

```
combined_data.dtypes
```

In [ ]:

```
"""
values  =  combined_data.values
scalar = MinMaxScaler(feature_range=(0,5))
x_scaled = scalar.fit_transform(values)
combined_data = pd.DataFrame(x_scaled,columns=combined_data.columns)
"""
combined_data.head()
```

Getting the train and test data back for prediction

In [ ]:

```
train_data , test_data = combined_data[:18359] , combined_data[18359:]
```

In [ ]:

```
x_train , y_train , x_val , y_val = train_data[:15000] , target[:15000] , train_data[15000:] , target[15000:
]
```

In [ ]:

```
def model_fit(alg, dtrain, target,dtest):
    xgb_param = alg.get_xgb_params()
    xgtrain = xgb.DMatrix(dtrain.values, label=target)
    cvresult = xgb.cv(xgb_param, xgtrain, num_boost_round=alg.get_params()['n_estimators'], nfold=5,
        early_stopping_rounds=50)

    #alg.fit(dtrain,target,use_best_model=True,eval_set=(x_val,y_val))
    alg.fit(dtrain,target)
    print("Model Report")
    print("Accuracy is {}".format(alg.score(x_val,y_val)))

    feat_imp = pd.Series(alg.feature_importances_,dtrain.columns).sort_values(ascending=False)
    feat_imp.plot(kind='bar', title='Feature Importances')
    plt.ylabel('Feature Importance Score')

    y_pred = alg.predict_proba(dtest)[:,1]
    return y_pred
```

I have used XGBoost Classifier with the following values for each parameter for prediction

In [ ]:

```
#clf = CatBoostClassifier(iterations=200,depth=4,eval_metric='AUC',l2_leaf_reg=9,learning_rate=0.1)
clf = XGBClassifier(
 learning_rate =0.3,
 n_estimators=100,
 max_depth=3,
 min_child_weight=1000,
 gamma=0.7,
 subsample=0.45,
 colsample_bytree=0.4,
 objective= 'binary:logistic',
 nthread=1,
 scale_pos_weight=1,
 seed=27,
reg_alpha =0.7,
 random_state=200)
```

In [ ]:

```
#clf = CatBoostClassifier(iterations=500,depth=4,learning_rate=0.03,eval_metric='AUC',loss_function='Logloss
')
clf = XGBClassifier(max_depth=4,n_estimators=150,learning_rate=0.05,colsample_bylevel=0.45,subsample=0.7)
```

In [ ]:

```
y_pred = model_fit(clf,x_train,y_train,test_data)    #87019
```

In [ ]:

```
#k_fold = KFold(len(train_data), n_folds=8, shuffle=True, random_state=0)
#print(np.mean(cross_val_score(clf, train_data, target, cv=k_fold, n_jobs=1))) #8678
```

In [ ]:

```
submission = pd.DataFrame({'enrollee_id': enrollee_id , 'target': y_pred})
submission.to_csv('submission.csv',index=False)
```

End of the Solution