/////////////////////////////////: README ://////////////////////////////////

PLATFORM: WINDOWS10>>ANACONDA-----SPYDER+JUPYTER NOTEBOOKS(PYTHON3.6)

INTEL I7 6TH GEN, 8GB DDR4, NVIDIA 960M

KAGGLE SCORE:

.375423(gbr)

.343664(rfr)

HIGHEST KAGGLE SCORE FOR THIS PROBLEM: .3924

RUNTIME:185 secs(GRADIENT BOOSTED ALGORITHM)

240 secs(RANDOM FOREST ALGORITHM)

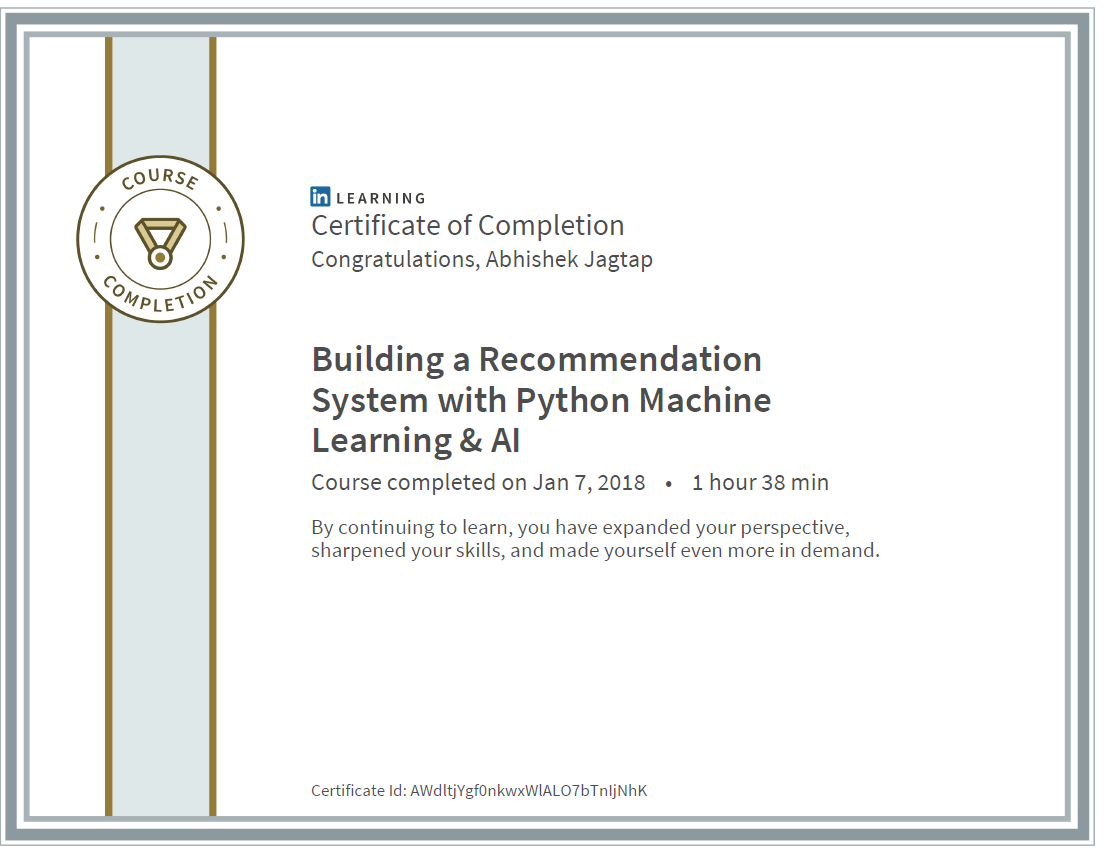
Feature selection: Selection of features as per their importance is also included in the files

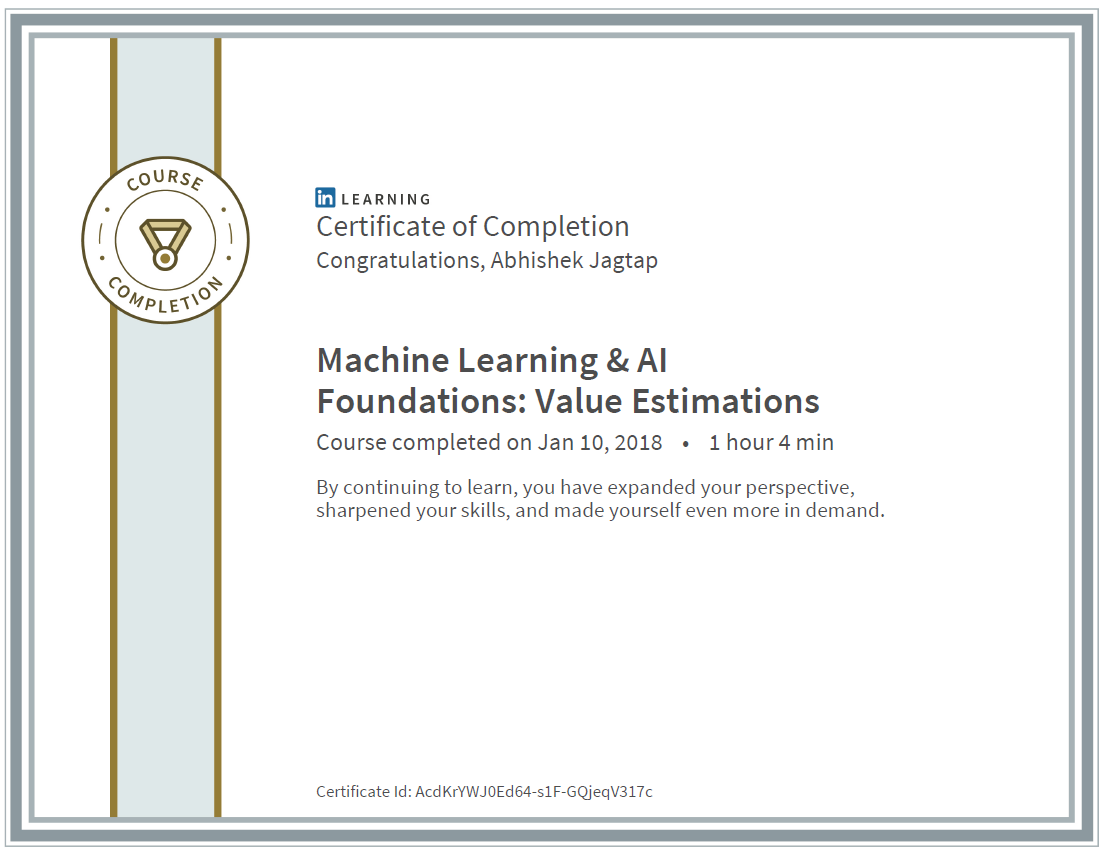
Code and its explanation is as below

LINK FOR LEARNING: <http://pbpython.com/categorical-encoding.html>

<http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html>

<http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html>

Courses done during hackathon period for better understanding:  




**USING RANDOM FOREST ALGORITHM**

# Liberty Mutual Group: Property Inspection Prediction

import pandas as pd

import numpy as np

from sklearn import preprocessing

from sklearn.ensemble import RandomForestRegressor

from sklearn.grid\_search import GridSearchCV

from sklearn.externals import joblib

#load train and test

df = pd.read\_csv('train.csv', index\_col=0)

tf = pd.read\_csv('test.csv', index\_col=0)

df.head()

tf.head()

tf.drop('T2\_V10', axis=1, inplace=True)

tf.drop('T2\_V7', axis=1, inplace=True)

tf.drop('T1\_V13', axis=1, inplace=True)

tf.drop('T1\_V10', axis=1, inplace=True)

tf.shape

labels = df.Hazard

df.drop('Hazard', axis=1, inplace=True)

df.drop('T2\_V10', axis=1, inplace=True)

df.drop('T2\_V7', axis=1, inplace=True)

df.drop('T1\_V13', axis=1, inplace=True)

df.drop('T1\_V10', axis=1, inplace=True)

df.shape

df.head()

columns = df.columns

test\_ind = tf.index

print(columns)

print(test\_ind)

df = np.array(df)

tf = np.array(tf)

print(df)

print(tf)

# label encode the categorical variables

for i in range(df.shape[1]):

if type(df[1,i]) is str:

lbl = preprocessing.LabelEncoder()

lbl.fit(list(df[:,i]) + list(tf[:,i]))

df[:,i] = lbl.transform(df[:,i])

tf[:,i] = lbl.transform(tf[:,i])

df = df.astype(float)

tf = tf.astype(float)

print(df)

print(tf)

param\_grid = {'n\_estimators': [100]}

model = GridSearchCV(RandomForestRegressor(), param\_grid)

model = model.fit(df,labels)

print("Best parameters found by grid search:")

print(model.best\_params\_)

print("Best CV score:")

print(model.best\_score\_)

preds = model.predict(tf)

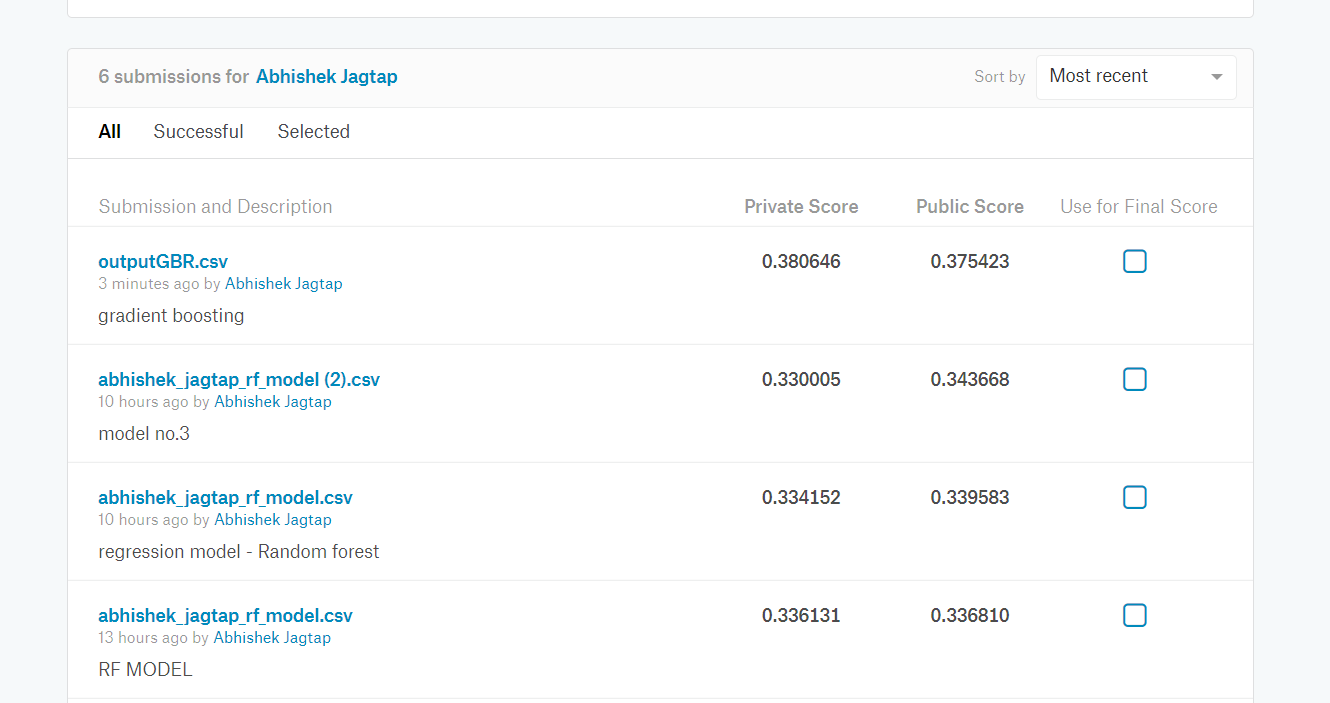
joblib.dump(model, 'Random\_forest\_regressor\_model.pkl')

#generate solution

preds = pd.DataFrame({"Id": test\_ind, "Hazard": preds})

preds = preds.set\_index('Id')

preds.to\_csv('abhishek\_jagtap\_rf\_model.csv')



**USING GRADIENT BOOSTING ALGORITHM**

#author - abhishek jagtap

#liberty mutual group property inspection prediction

#model used- gradient boosting regressor

#total runtime observed- 185secs on intel 6th generation 6500HQ edition 2.4GHZ CPU

#GPU - 960M # 8GB DDR4 RAM

#import necessary libraries

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn import ensemble

from sklearn.metrics import mean\_absolute\_error

from sklearn.externals import joblib

# Load the data set

df = pd.read\_csv("train.csv")

tf= pd.read\_csv("test.csv")

# This is to remove the unnecessary feature and creating the problem of overfitting

# This removal is done by checking the importance of each feature using feature selection file

df.drop('T2\_V10', axis=1, inplace=True)

df.drop('T2\_V7', axis=1, inplace=True)

df.drop('T1\_V13', axis=1, inplace=True)

df.drop('T1\_V10', axis=1, inplace=True)

# Alternative to remove the fields from the data set that we don't want to include in our model

#del df['T2\_V10']

#del df['T2\_V7']

#del df['T1\_V13']

#del df['T1\_V10']

#same thing for test data

tf.drop('T2\_V10', axis=1, inplace=True)

tf.drop('T2\_V7', axis=1, inplace=True)

tf.drop('T1\_V13', axis=1, inplace=True)

tf.drop('T1\_V10', axis=1, inplace=True)

labels = df.Hazard

df.drop('Hazard', axis=1, inplace=True)

columns = df.columns

# Replace categorical data with one-hot encoded data

features\_df = pd.get\_dummies(df, columns=['T1\_V4','T1\_V5','T1\_V6','T1\_V7','T1\_V8',

'T1\_V9','T1\_V11','T1\_V12','T1\_V15','T1\_V16',

'T1\_V17','T2\_V3','T2\_V4','T2\_V5','T2\_V11','T2\_V12','T2\_V13'])

features\_tf = pd.get\_dummies(tf, columns=['T1\_V4','T1\_V5','T1\_V6','T1\_V7','T1\_V8',

'T1\_V9','T1\_V11','T1\_V12','T1\_V15','T1\_V16',

'T1\_V17','T2\_V3','T2\_V4','T2\_V5','T2\_V11','T2\_V12','T2\_V13'])

features\_df = features\_df.astype(float)

features\_tf = features\_tf.astype(float)

test\_ind=features\_tf.index

#print the dimensions of the data after performing one-hot encoding

print("\n \n The dimensions of the train data are:")

print(features\_df.shape)

print("\n \n The dimensions of the test data are:")

print(features\_tf.shape)

# Split the data set in a training set (70%) and a test set (30%)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(features\_df, labels, test\_size=0.3, random\_state=0)

#n\_estimator=1000

# Fit regression model

model = ensemble.GradientBoostingRegressor(

n\_estimators=100,

learning\_rate=0.1,

max\_depth=6,

min\_samples\_leaf=9,

max\_features=0.1,

loss='huber',

random\_state=0

)

model.fit(X\_train, y\_train)

# Save the trained model to a file so we can use it in other programs for updating purpose

joblib.dump(model, 'trained\_house\_classifier\_model.pkl')

# Find the error rate on the training set

m1 = mean\_absolute\_error(y\_train, model.predict(X\_train))

print("Training Set Mean Absolute Error: %.4f" % m1)

# Find the error rate on the test set

m2 = mean\_absolute\_error(y\_test, model.predict(X\_test))

print("Test Set Mean Absolute Error: %.4f" % m2)

#abhi=joblib.load('trained\_house\_classifier\_model.pkl')

model.fit(features\_df,labels)

yp=model.predict(features\_tf)

#predict the values and save it to csv file

preds = pd.DataFrame({"Id": test\_ind, "Hazard": yp})

preds = preds.set\_index('Id')

preds.to\_csv('Abhishek\_jagtap\_SVM.csv')

