PROJECT 2

Choosing a Model for Predicting on Unseen Data

Problem Statement: Write a python program to perform K-Fold Cross Validation method on a given dataset, compare the J_Test and J_train mean values and select the model that has least error and predict the time taken for the year 2022 winter Olympics for women's 100m race.

Method of approach: Initially imported all the necessary packages that are needed for the execution of the program. Then I created a pandas dataframe to read the dataset as csv and added names to the rows and columns as years and time. To exclude the first line from the datatset, that is the number of lines and features associated with them, I created a variable, no_of_rows and no_of_columns with the df.iat[] function. Once this is done, I assigned the initial dataframe with the vales from the dataset that excluded the first line. Then I initialized two variables x,y and assigned the row values and column values respectively.

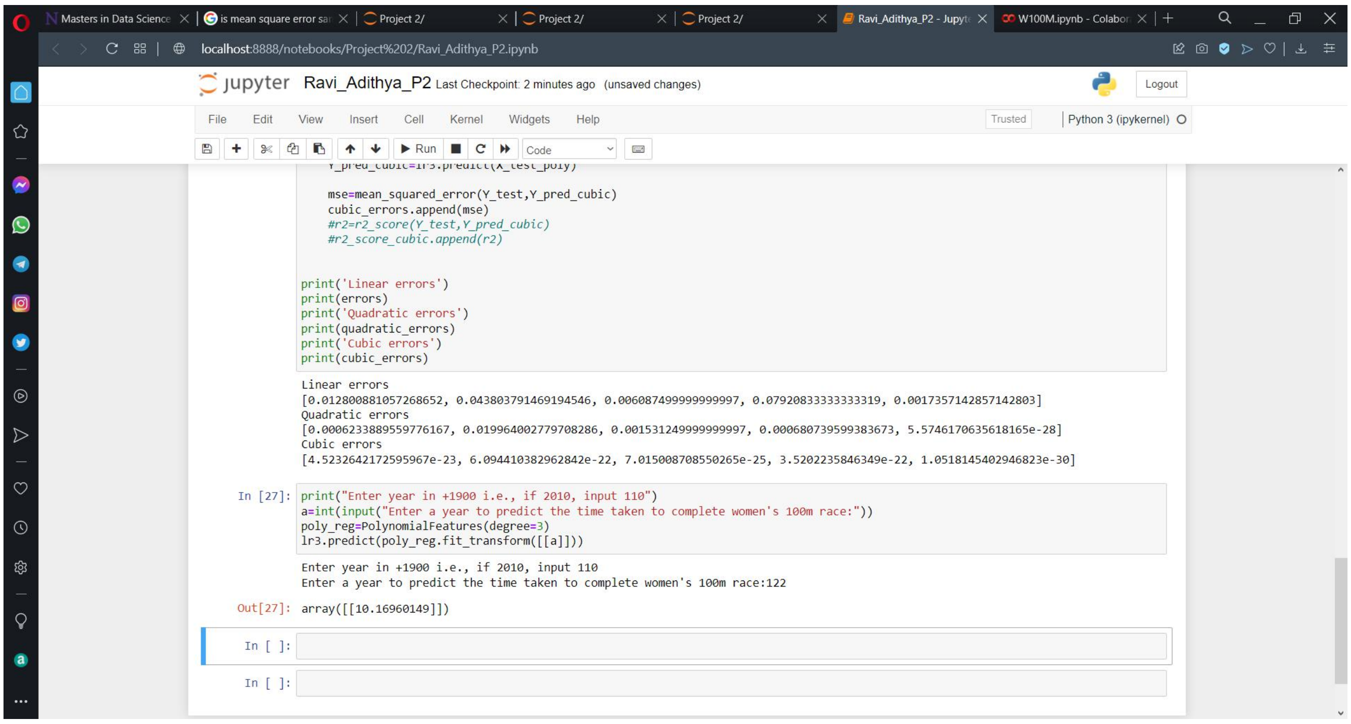
To do the K-Fold Cross Validation method from scratch without importing any inbuilt python packages from Sci-Kit Learn, I defined a function called cv_split(), that accepts the dataset and the number of folds as arguments. Firstly I created an empty array, dataset_split and another dataframe df_copy and assign the dataset value to it, because it is safer to work on a copy of the dataset rather making direct changes with the given dataset. Next, I created another variable - fold_size that calculated the number of elements the dataset is split into, so to do that taking the value of rows, diving it with the fold size (in this case K=5) and convert it into integer to get a proper split. Followed by that, I wrote a for loop to save each loop and declared an empty list. Within the for loop I wrote a while loop to add element s into the empty list. The values are randomized to begin with and then its appended to the list. Since we have got only 19 elements, that is its odd, I wrote an if statement that if df_copy.empty then break. Once all the looping process is done, the values are appended to the initially declared dataset_split array.

As they are normal arrays, I converted them to numpy array and then converted each of the fold as a pandas dataframe to a variable named 'K'.

For the main program, I imported all the required packages from sklearn and declared all necessary variables - error,quad_error,cubic_error. To split the dataset to test and training set, I wrote a for loop with count variable 'i' that ranges up to 5. Then created an empty pandas dataframe train_set and assigned the value of 'K[i]' to the test_set, then created a nested for loop with count variable 'j' and within that created an if statement with the condition 'j!=i', else there will be redundancy of the data. Followed by that I create an object for linear and polynomial regressions and printed their respective values. From the values obtained the mean train and mean test value is calculated. As per the problem statement I will have to plot a graph comparing the mean test and train set, which I failed to do.

Theoretically comparing the mean test and train set, the cubic model has the least error and will have to opt that model for future prediction to predict the time taken to complete the women's 100m race in winter Olympics 2022.

Output and Screenshot:



```
    Y_pred_cubic=lr3.predict(X_test_poly)

    mse=mean_squared_error(Y_test,Y_pred_cubic)
    cubic_errors.append(mse)
    #r2=r2_score(Y_test,Y_pred_cubic)
    #r2_score_cubic.append(r2)


print('Linear errors')
print(errors)
print('Quadratic errors')
print(quadratic_errors)
print('Cubic errors')
print(cubic_errors)
```

```
Linear errors
[0.012800881057268652, 0.043803791469194546, 0.006087499999999997, 0.07920833333333319, 0.0017357142857142803]
Quadratic errors
[0.0006233889559776167, 0.019964002779708286, 0.001531249999999997, 0.000680739599383673, 5.5746170635618165e-28]
Cubic errors
[4.5232642172595967e-23, 6.094410382962842e-22, 7.015008708550265e-25, 3.5202235846349e-22, 1.0518145402946823e-30]
```

```
In [27]: print("Enter year in +1900 i.e., if 2010, input 110")
         a=int(input("Enter a year to predict the time taken to complete women's 100m race:"))
         poly_reg=PolynomialFeatures(degree=3)
         lr3.predict(poly_reg.fit_transform([[a]]))

         Enter year in +1900 i.e., if 2010, input 110
         Enter a year to predict the time taken to complete women's 100m race:122

Out[27]: array([[10.16960149]])

In [ ]:

In [ ]:
```

Source Code:

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import sklearn
import numpy as np
from random import randrange

df=pd.read_csv('W100MTimes.txt',delimiter='\t',header=None, names=['Year','Seconds'])

no_of_rows=df.iat[0, 0]
no_of_cols=df.iat[0, 1]
df = df.iloc[1: , :]

x = df.iloc[:, 0].values
y = df.iloc[:, 1].values
```

```
def cv_split(dataset, folds):
        dataset_split = []
        df_copy = dataset
        fold_size = int(df_copy.shape[0] / folds)
        for i in range(folds):
            fold = []
            while len(fold) <fold_size+1
                r = randrange(df_copy.shape[0])
                index = df_copy.index[r]
                fold.append(df_copy.loc[index].values.tolist())
                df_copy = df_copy.drop(index)
                if df_copy.empty:
                    break
            dataset_split.append(np.asarray(fold))

        return dataset_split
folds = cross_validation_split(df,5)
folds=np.array(folds,dtype=object)
K=[]
K.append(pd.DataFrame(np.array(folds[0])))
K.append(pd.DataFrame(np.array(folds[1])))
K.append(pd.DataFrame(np.array(folds[2])))
K.append(pd.DataFrame(np.array(folds[3])))
K.append(pd.DataFrame(np.array(folds[4])))

from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score
errors=[]
quad_errors=[]
cubic_errors=[]
for i in range(5):
    train_set= pd.DataFrame()
    test_set=K[i]
    for j in range(5):
        if j!=i:
            train_set=train_set.append(K[i], ignore_index = True)

    #linear
    lr=LinearRegression()
    X_train=pd.DataFrame(train_set.iloc[:,0].values)
    Y_train=pd.DataFrame(train_set.iloc[:,1].values)
    X_test=pd.DataFrame(test_set.iloc[:,0].values)
    Y_test=pd.DataFrame(test_set.iloc[:,1].values)


    lr.fit(X_train,Y_train)
    Y_pred_linear=lr.predict(X_test)

    mse=mean_squared_error(Y_test,Y_pred_linear)
    errors.append(mse)



    #quadratic
    poly_reg=PolynomialFeatures(degree=2)
    X_train_poly=poly_reg.fit_transform(X_train)
    X_test_poly=poly_reg.fit_transform(X_test)
    lr2=LinearRegression()
    lr2.fit(X_train_poly,Y_train)
    Y_pred_quadratic=lr2.predict(X_test_poly)

    mse=mean_squared_error(Y_test,Y_pred_quadratic)
    quadratic_errors.append(mse)

    #cubic
    poly_reg=PolynomialFeatures(degree=3)
    X_train_poly=poly_reg.fit_transform(X_train)
    X_test_poly=poly_reg.fit_transform(X_test)
    lr3=LinearRegression()
    lr3.fit(X_train_poly,Y_train)
    Y_pred_cubic=lr3.predict(X_test_poly)

    mse=mean_squared_error(Y_test,Y_pred_cubic)
    cubic_errors.append(mse)

print('Linear errors')
print(errors)
print('Quadratic errors')
print(quadratic_errors)
print('Cubic errors')
print(cubic_errors)

print("Enter year in +1900 i.e., if 2010, input 110")
a=int(input("Enter a year to predict the time taken to complete women's 100m race:"))
poly_reg=PolynomialFeatures(degree=3)
lr3.predict(poly_reg.fit_transform([[a]]))
```

	Linear	Quadratic	Cubic
1234	0.058399148	0.027272229	1.57384E-21
5	0.015619847	1.11E-28	1.05E-30
1235	0.060610882	0.025654251	1.54694E-21
4	0.006772908	0.00647191	1.08E-22
1245	0.042768601	0.031152333	1.59883E-21
3	0.02461519	0.000973828	5.57E-23
1345	0.06005056	0.031391661	1.62176E-21
2	0.007333231	0.000734501	3.27E-23
2345	0.042626291	0.008180239	1.95981E-22
1	0.0247575	0.023945922	1.46E-21
Mean train	**0.230354448**	**0.117106521**	**6.38057E-21**
Mean test	**0.066602798**	**0.032126161**	**1.65451E-21**