Exercise 2 - Part B

November 19, 2021

Machine Learning Lab

Lab 02

Exercise 2 - Part B

Importing Packages

[1]:
```
import pandas as pd       #Importing Pandas
import numpy as np        #Importing Numpy
import math               #Importing Math
```

Reading Training CSV data into the Dataframe

[2]:
```
train_df = pd.read_csv('train.csv',low_memory=False)
#Setting Store column as my Index in Dataframe
train_df.set_index('Store',inplace=True)
train_df.head()
```

[2]:
```
       DayOfWeek        Date  Sales  Customers  Open  Promo StateHoliday  \
Store
1              5  2015-07-31   5263        555     1      1            0
2              5  2015-07-31   6064        625     1      1            0
3              5  2015-07-31   8314        821     1      1            0
4              5  2015-07-31  13995       1498     1      1            0
5              5  2015-07-31   4822        559     1      1            0

       SchoolHoliday
Store
1                  1
2                  1
3                  1
4                  1
5                  1
```

Reading Store CSV data into the Dataframe

[3]:
```
store_df = pd.read_csv('store.csv')
#Setting Store column as my Index in Dataframe
store_df.set_index('Store',inplace=True)
store_df.head()
```

[3]:
```
      StoreType Assortment  CompetitionDistance  CompetitionOpenSinceMonth  \
Store
1             c          a               1270.0                        9.0
2             a          a                570.0                       11.0
3             a          a              14130.0                       12.0
4             c          c                620.0                        9.0
5             a          a              29910.0                        4.0

       CompetitionOpenSinceYear  Promo2  Promo2SinceWeek  Promo2SinceYear  \
Store
1                        2008.0       0              NaN              NaN
2                        2007.0       1             13.0           2010.0
3                        2006.0       1             14.0           2011.0
4                        2009.0       0              NaN              NaN
5                        2015.0       0              NaN              NaN

         PromoInterval
Store
1                  NaN
2      Jan,Apr,Jul,Oct
3      Jan,Apr,Jul,Oct
4                  NaN
5                  NaN
```

Merging Training and Store Dataframes into a single Dataframe

[4]:
```
#Merging train and store dataframes on Store column
merged_df = pd.merge(train_df,store_df,how='inner',on='Store')
merged_df
```

[4]:
```
       DayOfWeek        Date  Sales  Customers  Open  Promo StateHoliday  \
Store
1              5  2015-07-31   5263        555     1      1            0
1              4  2015-07-30   5020        546     1      1            0
1              3  2015-07-29   4782        523     1      1            0
1              2  2015-07-28   5011        560     1      1            0
1              1  2015-07-27   6102        612     1      1            0
…          …          …   …      …   …    …          …
1115           6  2013-01-05   4771        339     1      0            0
1115           5  2013-01-04   4540        326     1      0            0
1115           4  2013-01-03   4297        300     1      0            0
1115           3  2013-01-02   3697        305     1      0            0
1115           2  2013-01-01      0          0     0      0            a

       SchoolHoliday StoreType Assortment  CompetitionDistance  \
Store
1                  1         c          a               1270.0
1                  1         c          a               1270.0
```

```
1                   1          c           a                1270.0
1                   1          c           a                1270.0
1                   1          c           a                1270.0
…                …        …         …                   …
1115                1          d           c                5350.0
1115                1          d           c                5350.0
1115                1          d           c                5350.0
1115                1          d           c                5350.0
1115                1          d           c                5350.0

       CompetitionOpenSinceMonth  CompetitionOpenSinceYear  Promo2  \
Store
1                            9.0                    2008.0       0
1                            9.0                    2008.0       0
1                            9.0                    2008.0       0
1                            9.0                    2008.0       0
1                            9.0                    2008.0       0
…                           …                        …     …
1115                         NaN                       NaN       1
1115                         NaN                       NaN       1
1115                         NaN                       NaN       1
1115                         NaN                       NaN       1
1115                         NaN                       NaN       1

       Promo2SinceWeek  Promo2SinceYear     PromoInterval
Store
1                  NaN              NaN               NaN
1                  NaN              NaN               NaN
1                  NaN              NaN               NaN
1                  NaN              NaN               NaN
1                  NaN              NaN               NaN
…                 …               …               …
1115              22.0           2012.0  Mar,Jun,Sept,Dec
1115              22.0           2012.0  Mar,Jun,Sept,Dec
1115              22.0           2012.0  Mar,Jun,Sept,Dec
1115              22.0           2012.0  Mar,Jun,Sept,Dec
1115              22.0           2012.0  Mar,Jun,Sept,Dec

[1017209 rows x 17 columns]
```

Cleaning and Preparing Dataframe for Analysis

[5]:
```
#For NA values in column Competition Distance, we are filling it with the␣
↪median value of the whole column
merged_df.CompetitionDistance.fillna(merged_df.CompetitionDistance.median(),␣
↪inplace = True)
#For NA in any other column, replacing NA with 0
merged_df.fillna(0, inplace = True)
```

Find all the stores that have sales recorded for 942 days

[6]:
```
merged_df = merged_df[merged_df.groupby(by='Store').count() >= 942]
#Taking only Store and Sales Columns
merged_df = merged_df[['Sales']]
merged_df
```

[6]:
```
        Sales
Store
1      5263.0
1      5020.0
1      4782.0
1      5011.0
1      6102.0
…        …
1115   4771.0
1115   4540.0
1115   4297.0
1115   3697.0
1115      0.0

[1017209 rows x 1 columns]
```

Create a data matrix of the shape (#_of_stores, 942) for the daily sales record of these stores.

[7]:
```
#First Grouping the dataframe by Store so to get (# of stores) rows
store_sales = merged_df.groupby(['Store'])

#For each store, converting its sales column to a list and storing it into a␣
↪single cell
store_sales = store_sales['Sales'].agg(lambda x: list(x))

#Converting the list each cell into 942 different columns and each row now␣
↪represent each store
store_sales = pd.DataFrame(store_sales.values.tolist(),columns=[x+1 for x in␣
↪range(942)])
store_sales
```

[7]:
```
            1        2        3        4        5    6       7       8   \
0      5263.0   5020.0   4782.0   5011.0   6102.0  0.0  4364.0  3706.0
1      6064.0   5567.0   6402.0   5671.0   6627.0  0.0  2512.0  3854.0
2      8314.0   8977.0   7610.0   8864.0   8107.0  0.0  3878.0  5080.0
3     13995.0  10387.0  10514.0  10275.0  11812.0  0.0  9322.0  8322.0
4      4822.0   4943.0   5899.0   6083.0   7059.0  0.0  2030.0  3815.0
…        …       …        …        …        …   …      …       …
1110   5723.0   5263.0   4907.0   6793.0   7742.0  0.0  2177.0  3918.0
1111   9626.0   9652.0   9179.0   9583.0  14383.0  0.0  6216.0  6220.0
1112   7289.0   7491.0   6640.0   6468.0   7582.0  0.0  4784.0  6399.0
```

```
1113  27508.0  24395.0  25840.0  25518.0  26720.0  0.0  21312.0  19627.0
1114   8680.0   8405.0   7661.0   8093.0  10712.0  0.0   6897.0   5816.0

            9       10   …      933      934      935      936  937      938  \
0      3769.0   3464.0   …   4892.0   5471.0   5580.0   7176.0  0.0   4997.0
1      4108.0   5093.0   …   5618.0   6763.0   6318.0   6775.0  0.0   2342.0
2      5702.0   5414.0   …   7772.0   8001.0   9800.0  12247.0  0.0   4523.0
3      7286.0   8503.0   …   9472.0   8857.0  10031.0  12112.0  0.0  10338.0
4      3713.0   3595.0   …   4999.0   5974.0   5718.0   6978.0  0.0   1590.0
…         …        …   …     …        …        …        …   …        …
1110   3587.0   4021.0   …   5887.0   5307.0   6472.0   9444.0  0.0   3325.0
1111   6730.0   6029.0   …  14366.0  14724.0  17058.0  25165.0  0.0   9513.0
1112   6410.0   4565.0   …   7508.0   6115.0   6866.0   8984.0  0.0   5194.0
1113  20564.0  20424.0   …  18075.0  17073.0  18816.0  21237.0  0.0  18856.0
1114   6150.0   5342.0   …   5007.0   4649.0   5243.0   6905.0  0.0   4771.0

          939      940      941  942
0      4486.0   4327.0   5530.0  0.0
1      4484.0   4159.0   4422.0  0.0
2      6069.0   5902.0   6823.0  0.0
3      8290.0   8247.0   9941.0  0.0
4      4456.0   3465.0   4253.0  0.0
…         …        …        …    …
1110   4640.0   4579.0   5097.0  0.0
1111   9788.0   8716.0  10797.0  0.0
1112   5524.0   5563.0   6218.0  0.0
1113  18371.0  18463.0  20642.0  0.0
1114   4540.0   4297.0   3697.0  0.0

[1115 rows x 942 columns]
```

Use the first 800 stores in this data matrix for training and the rest for testing. Also split the sales data into 2 parts, the 1st part contains the information about the first 900 days of sales (these would be the features) and the 2nd contains the information about the last 42 days of sales (these would be the targets).

[8]:
```
#Using iloc to split dataset and then converting this to numpy array for␣
 ↪further processing
X_train = store_sales.iloc[:800, :900].to_numpy()
X_test = store_sales.iloc[800:,:900].to_numpy()
Y_train = store_sales.iloc[:800,900:].to_numpy()
Y_test = store_sales.iloc[800:,900:].to_numpy()
```

[9]:
```
#Replacing every NaN value with 0 in training and test Dataset
X_train[np.isnan(X_train)] = 0
Y_train[np.isnan(Y_train)] = 0
X_test[np.isnan(X_test)] = 0
Y_test[np.isnan(Y_test)] = 0
```

Iteratively build multiple linear regression models for column vectors of . You are allowed to use the numpy routines for calculating inverses, transposing of matrices and matrix multiplication. You would need to create 42 models in this case (1 model for each day in the target sales matrix)

[10]:
```
#Calculating Matrix A which equals X^t.X
A = X_train.T @ X_train

#Creating a matrix which stores beta vector for next 42 days
all_beta = np.zeros(shape=(900,42))

#Iterating through the columns for Y train
for i in range(Y_train.shape[1]):
    #Initialize empty beta array
    beta = np.zeros(shape=(900,1))
    #Calculating beta vector using equation: (X^t.X)^-1.(X^t.Y)
    beta = np.array(np.dot(np.linalg.inv(A),np.dot(X_train.T,Y_train[:,i])))
    #Reshaping beta array to column vector
    beta = beta.reshape(-1,1)
    #Copying new beta values to beta matrix
    all_beta[:,i] = beta[:,0]
```

Verify that you have learned 0:900 for each of the 42 models and use these learned parameters to make predictions for each day ahead. In total 42 days.

[11]:
```
#Calculating predicted y y hat using the X test and beta values
y_hat = X_train @ all_beta
```

Calculate and print the daily RMSE and MAE for all 42 sales values using test split (as input). Also calculate and print overall average RMSE and MAE. (i.e. just the mean RMSE of all 42 models).

Function to calculate RMSE between Actual Y and Predicted Y

[12]:
```
def calculate_rmse(actual,predicted):
    N = len(actual)
    summation = 0
    for i in range(len(actual)):
        summation += ((actual[i] - predicted[i]) ** 2)
    return math.sqrt(summation/N)
```

Function to calculate MAE between Actual Y and Predicted Y

[13]:
```
def calculate_mae(actual,predicted):
    N = len(actual)
    summation = 0
    for i in range(len(actual)):
        summation += abs(actual[i] - predicted[i])
    return summation/N
```

[14]:

```python
#Initializing two empty array for storing RMSE and MAE
rmse = np.zeros(shape=(Y_train.shape[1],))
mae = np.zeros(shape=(Y_train.shape[1],))

#Iterating through Y values
for i in range(Y_train.shape[1]):
    rmse[i] = calculate_rmse(Y_train[:,i],y_hat[:,i])
    mae[i] = calculate_mae(Y_train[:,i],y_hat[:,i])

#Converting RMSE and MAE list to Dataframe
error_df = pd.DataFrame(np.hstack((rmse.reshape(-1,1),mae.
 ↪reshape(-1,1))),columns=['RMSE','MAE'])
error_df
```

[14]:

	RMSE	MAE
0	4.731422e+05	305195.909448
1	1.728305e+05	106405.221263
2	5.946421e+05	360909.887114
3	7.034829e+05	437619.131678
4	7.345277e+05	435543.805615
5	6.432622e+05	423011.717900
6	7.981136e+05	478594.333921
7	9.807624e+05	587478.966126
8	1.806327e+05	108853.805682
9	6.210497e+05	374447.000432
10	7.538311e+05	448565.176230
11	6.681774e+05	400354.796431
12	5.720995e+05	370470.913420
13	4.916015e+05	306657.210576
14	5.553930e+05	337416.307653
15	1.438492e+05	90126.687726
16	5.382004e+05	326645.534836
17	6.743641e+05	449194.804973
18	6.755049e+05	406534.428682
19	6.922066e+05	425230.854253
20	7.876939e+05	475979.539648
21	7.310623e+05	479163.659818
22	1.671382e+05	105600.794973
23	5.580886e+05	341107.885112
24	5.901033e+05	341277.665779
25	4.808007e+05	308412.184117
26	4.986938e+05	316512.642878
27	5.233060e+05	318318.903058
28	5.327361e+05	331078.892981
29	1.502887e+05	92746.161345
30	4.857587e+05	319835.723291
31	6.749813e+05	443260.238391

```
32  7.028667e+05  438025.457057
33  7.412883e+05  425917.227332
34  8.806699e+05  500656.773381
35  1.007064e+06  602554.784827
36  1.469232e+05   91284.073994
37  5.342813e+05  338508.310549
38  6.215749e+05  375212.543530
39  6.546487e+05  384561.950083
40  6.343508e+05  391866.106794
41  1.054245e+05   65956.638679
```

Reason why or why not Linear Regression is a good choice for this task. I think linear regression is not a good choice for this task, instead we can use time series forcast method Autoregressive model (AR) because it predict future behaviour based on past behaviour.The linear regression model uses the linear combination of predictors for predicting future values whereas AR model uses all the past values in time and then predict the future values.