IE4213 – Regression Example

PROBLEM DESCRIPTION

Construct regression models for HDB resale prices using the data in *hdb1720.csv*. This data was extracted from: https://data.gov.sg/dataset/resale-flat-prices.

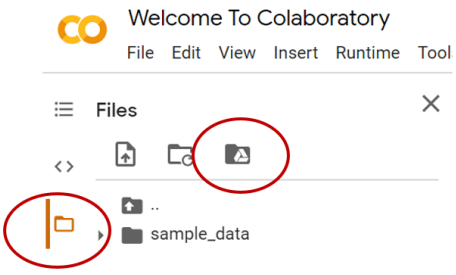
INITIAL SETUP

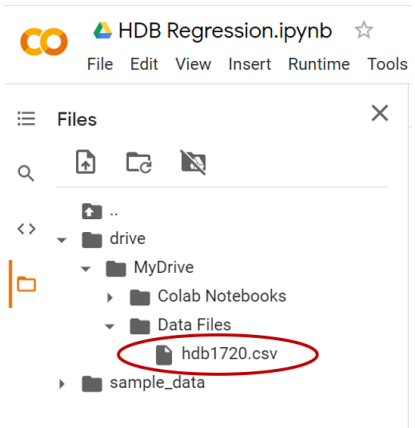
Upload data files on Google Drive:

1. Go to the Google Drive website.
2. Click "Go to Drive" on the top right.
3. Create a new folder "Data Files" by pressing Right-Click -> New Folder.
4. Open Data Files folder by double clicking on it.
5. Copy *hdb1720.csv* into this folder by dragging.

Mount Google Drive to allow access to Data Files.

1. Create a new notebook on Colab. Give your notebook a name.
2. Click on the Folder icon on the left to bring up Files menu. Next, click on Mount Drive. Grant access permission. See left image below.
3. You should be able to see your Data Files folder in Google Drive along with the files inside the folder. See right image below.





DATA PREPARATION

Read in data from csv file *hdb1720.csv* that is store on Google Drive under folder *Data Files*:

```
import pandas as pd
data = pd.read_csv("/content/drive/MyDrive/Data Files/hdb1720.csv")
```

View the number of rows and columns that this data set contains:

```
data.shape
```

1. This data set contains:

   - Number of transactions:
   - Number of data fields per transactions:

View the first 5 rows of the data set:

```
data.head(5)
```

For simplicity, let's exclude block and street name from our analysis. In addition, observe that lease commencement date and remaining lease are very similar. Hence, we exclude remaining lease from our analysis as well:

```
trim_data = data.drop(["block", "street_name", "remaining_lease"], axis=1)
trim_data.head(5)
```

2. Regression is performed on numerical data, but not all of our data are numerical. For these data, we will need to transform them into numerical value. First, we need to identify the data fields requiring this transformation. List down the non-numerical data in our data set.

   Data fields with non-numerical data:

Prepare our X and y data for regression:

```
from sklearn.preprocessing import OrdinalEncoder
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer

pipeline = ColumnTransformer([
("o", OrdinalEncoder(), ["month","flat_type","storey_range"]),
("n", OneHotEncoder(), ["town", "flat_model"]),
], remainder='passthrough')

X = pipeline.fit_transform(trim_data.drop(["resale_price"], axis=1))
y = trim_data["resale_price"]
```

3. The data under month were transformed using the ordinal encoder. For example, 2017-01 is converted into 0 and 2017-02 is converted into 1. The python code to view the values that were under the month data field and the sequence in which it is encoded is provided below.

View the values that were under the month data field:

```
encoder = OrdinalEncoder()
encoder.fit_transform(data[["month"]])
encoder.categories_
```

Determine how the values under storey_range were encoded:

Original Value	New Value
01 TO 03	
04 TO 06	
07 TO 09	
10 TO 12	
13 TO 15	
16 TO 18	
19 TO 21	
22 TO 24	
25 TO 27	
28 TO 30	
31 TO 33	
34 TO 36	
37 TO 39	
40 TO 42	
43 TO 45	
46 TO 48	
49 TO 51	

Determine how the values under flat_type were encoded:

Original Value	New Value
1 ROOM	
2 ROOM	
3 ROOM	
4 ROOM	
5 ROOM	
EXECUTIVE	
MULTI-GENERATION	

4. The data under town and flat_model were transformed using the one-hot encoder (i.e. binary type conversion). For these fields, an additional column is included for every unique data value. For example, since there are 26 different HDB towns, 26 columns were added. The python code to view the transformed data is provided below.

View the transformed data of the first transaction:

```
X.toarray()[0]
```

Each transaction (i.e., row) contains 51 data fields (i.e., columns). By examining the transformed data of the different transactions, list down what each of the columns corresponds to and the data type (numerical or binary):

Index	Data	Type
0	Month	Numerical
1		
2		
3		
4		
5		
6		
7		
8		
9		
10		
11		
12		
13		
14		
15		
16		
17		
18		
19		
20		
21		
22		
23		
24		
25		

Index	Data	Type
26		
27		
28		
29		
30		
31		
32		
33		
34		
35		
36		
37		
38		
39		
40		
41		
42		
43		
44		
45		
46		
47		
48		
49		
50		

5. Based on your solutions to the previous two questions, fill in the data fields for the second transaction (i.e., occurred on Jan 2017 for an Ang Mo Kio, 3-Room, Level 2, 67 sqm, New Generation flat whose lease started on 1978):

Index	0	1	2	3	4	5	6	7
Value								

Index	8	9	10	11	12	13	14	15
Value								

Index	16	17	18	19	20	21	22	23
Value								

Index	24	25	26	27	28	29	30	31
Value								

Index	32	33	34	35	36	37	38	39
Value								

Index	40	41	42	43	44	45	46	47
Value								

Index	48	49	50
Value			

The following code can be used to verify your solutions above:

```
X.toarray()[1]
```

REGRESSION

6. Having prepared our data, we are now ready to perform regression on it. Use the Python code below to perform linear regression and report the model R^2:

Perform linear regression:

```
from sklearn.linear_model import LinearRegression
lr = LinearRegression()
lr.fit(X, y)
print("Intercept: ", lr.intercept_)
print("Coefficient: ", lr.coef_)
print("R-sq of model: ", lr.score(X, y))
```

- R^2 of linear regression model:

Perform support vector regression:

```
from sklearn.svm import LinearSVR
svmr = LinearSVR(epsilon=0, C=100, max_iter=1000000, random_state=0).fit(X, y)
print("Intercept: ", svmr.intercept_)
print("Coefficient: ", svmr.coef_)
print("R-sq of model: ", svmr.score(X, y))
```

- R^2 of support vector regression model:

Perform decision tree regression:

```
from sklearn.tree import DecisionTreeRegressor
dtr = DecisionTreeRegressor(criterion='mse', max_depth=12, min_samples_leaf=1, random_state=0).fit(X, y)
print("R-sq of model: ", dtr.score(X, y))
```

- R^2 of decision tree regression model:

7. Use the Python code below to estimate the resale price of a Punggol, 5-Room, Level 25, 80 sqm, Standard flat whose lease started on 2001 selling on March 2017:

Estimate resale price:

```
import numpy as np
input = np.zeros(51)
input[0] = 2      #set month to Mar 2017
input[1] = 4      #set flat type to 5-Room
input[2] = 8      #set storey range to 25
input[20] = 1     #set town to Punggol
input[45] = 1     #set flat model to Standard
input[49] = 80    #set floor area to 80 sqm
input[50] = 2001 #set lease commencement to 2001
print(input)
print("Estimated resale price (linear regression)=> ", lr.predict([input]))
print("Estimated resale price (SVM regression)=> ", svmr.predict([input]))
print("Estimated resale price (DT regression)=> ", dtr.predict([input]))
```

Estimate resale price

- Linear Regression:
- Support Vector Regression:
- Decision Tree Regression:

MODEL VALIDATION

Previously, we have discussed how having a higher R^2 does not guarantee better predictions. We have also demonstrated how cross-validation can be used to evaluate the prediction accuracy of our models.

However, the HDB resale data is slightly different in that there is also the time component to consider. For example, it is illogical to predict 2019 prices using a model fitted with 2020 data. Next, we illustrate how the previous twelve months data can be used to fit a model for prediction and then use the subsequent month's data to evaluate the model's prediction accuracy.

Convert X and y into a Pandas DataFrame:

```
col_name = ["month","type","storey","AMK","BED","BIS","BBT","BMH","BPJ",
            "BTM","CEN","CCK","CLE","GEY","HOU","JRE","JRW","KAL","MAR",
            "PAS","PUN","QUE","SEM","SKG","SER","TAM","TOA","WOO",
            "YIS","2-room","Adjoined","Apartment","DBSS","Improved",
            "Improved-M","Maisonette","Model A","Model A-M","Model A2",
            "Multi Gen","New Gen","Premium Apt","Premium Apt Loft",
            "Premium M","Simplified","Standard","Terrace","Type S1",
            "Type S2","Area","Lease"]

df_X = pd.DataFrame(X.toarray(),columns=col_name)
df_Xy = df_X.assign(resale_price = y)
print(df_Xy)
```

Extract train and test data:

```
df_test = df_Xy[df_Xy.month == 12]
X_test = df_test.drop(["resale_price"], axis=1)
y_test = df_test.resale_price

df_train = df_Xy[(df_Xy.month >= 0) & (df_Xy.month <= 11)]
X_train = df_train.drop(["resale_price"], axis=1)
y_train = df_train.resale_price
```

Perform linear regression using train data and evaluate using test data:

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score

lr = LinearRegression()
lr.fit(X_train, y_train)
print("R-sq of model: ", lr.score(X_train, y_train))
pred = lr.predict(X_test)
print("RMSE: %.2f" % mean_squared_error(y_test, pred, squared=False))
print("R-sq of predictions: ", r2_score(y_test, pred))
```

8. Report the fit of model (based on train data) and its predictive accuracy (based on test data):

- R^2 of linear regression model:
- Root mean squared error of predictions:
- R^2 of predictions:

Perform support vector regression using train data and evaluate using test data:

```
from sklearn.svm import LinearSVR
from sklearn.metrics import mean_squared_error, r2_score

svmr = LinearSVR(epsilon=0, C=100, max_iter=1000000, random_state=0).fit(X_train, y_train)
print("R-sq of model: ", svmr.score(X_train, y_train))
pred = svmr.predict(X_test)
print("RMSE: %.2f" % mean_squared_error(y_test, pred, squared=False))
print("R-sq of predictions: ", r2_score(y_test, pred))
```

9. Report the fit of model (based on train data) and its predictive accuracy (based on test data):

- R^2 of linear regression model:
- Root mean squared error of predictions:
- R^2 of predictions:

Perform decision tree regression using train data and evaluate using test data:

```
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import mean_squared_error, r2_score

dtr = DecisionTreeRegressor(criterion='mse', max_depth=12, min_samples_leaf=1, random_state=0).fit(X_train, y_train)
print("R-sq of model: ", dtr.score(X_train, y_train))
pred = dtr.predict(X_test)
print("RMSE: %.2f" % mean_squared_error(y_test, pred, squared=False))
print("R-sq of predictions: ", r2_score(y_test, pred))
```

10. Report the fit of model (based on train data) and its predictive accuracy (based on test data):

- R^2 of decision tree regression model:
- Root mean squared error of predictions:
- R^2 of predictions: