1.Cross Validation for Multiple Linear Regression

October 28, 2021

Cross Validation for Multiple Linear Regression

[44]:
```
# Import necessary package
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings # supress warnings
warnings.filterwarnings('ignore')
```

0.1 Building Multiple Linear Regression Model Without Cross-Validation

0.1.1 Step 1: Load the dataset

[45]:
```
df1 = pd.read_csv('E:\\MY LECTURES\\DATA SCIENCE\\3.Programs\\dataset\\Housing.
↪csv')
df1.head()
```

[45]:

	price	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	\
0	13300000	7420	4	2	3	yes	no	no	
1	12250000	8960	4	4	4	yes	no	no	
2	12250000	9960	3	2	2	yes	no	yes	
3	12215000	7500	4	2	2	yes	no	yes	
4	11410000	7420	4	1	2	yes	yes	yes	

	hotwaterheating	airconditioning	parking	prefarea	furnishingstatus
0	no	yes	2	yes	furnished
1	no	yes	3	no	furnished
2	no	no	2	yes	semi-furnished
3	no	yes	3	yes	furnished
4	no	yes	2	no	furnished

0.1.2 Step 2: Apply EDA

Univariate and bivariate analysis

0.1.3 Step 3. Pre-process and extract the features

[46]:
```
# data preparation - list of all the "yes-no" binary categorical variables
# we will map yes to 1 and no to 0
binary_vars_list =  ['mainroad', 'guestroom', 'basement', 'hotwaterheating',␣
 ↪'airconditioning', 'prefarea']

# defining the map function
def binary_map(x):
    return x.map({'yes': 1, "no": 0})

# applying the function to the housing variables list
df1[binary_vars_list] = df1[binary_vars_list].apply(binary_map)
df1.head()
```

[46]:

	price	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	hotwaterheating	airconditioning	parking	prefarea	furnishingstatus
0	13300000	7420	4	2	3	1	0	0	0	1	2	1	furnished
1	12250000	8960	4	4	4	1	0	0	0	1	3	0	furnished
2	12250000	9960	3	2	2	1	0	1	0	0	2	1	semi-furnished
3	12215000	7500	4	2	2	1	0	1	0	1	3	1	furnished
4	11410000	7420	4	1	2	1	1	1	0	1	2	0	furnished

[47]:
```
# 'dummy' variables
# get dummy variables for 'furnishingstatus'
# also, drop the first column of the resulting df (since n-1 dummy vars suffice)
status = pd.get_dummies(df1['furnishingstatus'], drop_first = True)
status.head()
```

[47]:

	semi-furnished	unfurnished
0	0	0
1	0	0
2	1	0
3	0	0

```
4               0            0
```

[48]:
```
# concat the dummy variable df with the main df
df1 = pd.concat([df1, status], axis = 1)
df1.head()
```

[48]:
```
      price  area  bedrooms  bathrooms  stories  mainroad  guestroom  \
0  13300000  7420         4          2        3         1          0
1  12250000  8960         4          4        4         1          0
2  12250000  9960         3          2        2         1          0
3  12215000  7500         4          2        2         1          0
4  11410000  7420         4          1        2         1          1

   basement  hotwaterheating  airconditioning  parking  prefarea  \
0         0                0                1        2         1
1         0                0                1        3         0
2         1                0                0        2         1
3         1                0                1        3         1
4         1                0                1        2         0

  furnishingstatus  semi-furnished  unfurnished
0        furnished               0            0
1        furnished               0            0
2   semi-furnished               1            0
3        furnished               0            0
4        furnished               0            0
```

[49]:
```
# remove 'furnishingstatus' since we alreday have the dummy vars
df1.drop(['furnishingstatus'], axis = 1, inplace = True)
df1.head()
```

[49]:
```
      price  area  bedrooms  bathrooms  stories  mainroad  guestroom  \
0  13300000  7420         4          2        3         1          0
1  12250000  8960         4          4        4         1          0
2  12250000  9960         3          2        2         1          0
3  12215000  7500         4          2        2         1          0
4  11410000  7420         4          1        2         1          1

   basement  hotwaterheating  airconditioning  parking  prefarea  \
0         0                0                1        2         1
1         0                0                1        3         0
2         1                0                0        2         1
3         1                0                1        3         1
4         1                0                1        2         0

   semi-furnished  unfurnished
0               0            0
```

```
1                0              0
2                1              0
3                0              0
4                0              0
```

[50]:
```
# extracting relevant features
numeric_vars = ['area', 'bedrooms', 'bathrooms', 'stories', 'parking','price']
temp = df1[numeric_vars]
temp.head()
```

[50]:

	area	bedrooms	bathrooms	stories	parking	price
0	7420	4	2	3	2	13300000
1	8960	4	4	4	3	12250000
2	9960	3	2	2	2	12250000
3	7500	4	2	2	3	12215000
4	7420	4	1	2	2	11410000

[51]:
```
# rescale the features between 0 to 1
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
temp1 = scaler.fit_transform(temp)
temp1 = pd.DataFrame(temp1,columns=['area', 'bedrooms', 'bathrooms', 'stories',␣
↪'parking','price'])
temp1.head()
```

[51]:

	area	bedrooms	bathrooms	stories	parking	price
0	0.396564	0.6	0.333333	0.666667	0.666667	1.000000
1	0.502405	0.6	1.000000	1.000000	1.000000	0.909091
2	0.571134	0.4	0.333333	0.333333	0.666667	0.909091
3	0.402062	0.6	0.333333	0.333333	1.000000	0.906061
4	0.396564	0.6	0.000000	0.333333	0.666667	0.836364

[52]:
```
df2 = df1[["mainroad","guestroom", "basement", "hotwaterheating",␣
↪"airconditioning", "prefarea", "semi-furnished"]]
df2.head()
```

[52]:

	mainroad	guestroom	basement	hotwaterheating	airconditioning	prefarea	\
0	1	0	0	0	1	1	
1	1	0	0	0	1	0	
2	1	0	1	0	0	1	
3	1	0	1	0	1	1	
4	1	1	1	0	1	0	

	semi-furnished
0	0
1	0
2	1

```
3                0
4                0
```

[53]:
```
pre_processed_data = pd.concat([df2,temp1],axis=1)
pre_processed_data.head()
```

[53]:

	mainroad	guestroom	basement	hotwaterheating	airconditioning	prefarea	\
0	1	0	0	0	1	1	
1	1	0	0	0	1	0	
2	1	0	1	0	0	1	
3	1	0	1	0	1	1	
4	1	1	1	0	1	0	

	semi-furnished	area	bedrooms	bathrooms	stories	parking	price
0	0	0.396564	0.6	0.333333	0.666667	0.666667	1.000000
1	0	0.502405	0.6	1.000000	1.000000	1.000000	0.909091
2	1	0.571134	0.4	0.333333	0.333333	0.666667	0.909091
3	0	0.402062	0.6	0.333333	0.333333	1.000000	0.906061
4	0	0.396564	0.6	0.000000	0.333333	0.666667	0.836364

[54]:
```
pre_processed_data.shape
```

[54]:
```
(545, 13)
```

0.1.4 Step 4. Split the data for training and testing

[55]:
```
from sklearn.model_selection import train_test_split
train, test = train_test_split(pre_processed_data, train_size = 0.8, test_size␣
 ↪= 0.2, random_state = 100)
```

[56]:
```
# divide into X_train, y_train, X_test, y_test
y_train = train.pop('price')
x_train = train

y_test = test.pop('price')
x_test = test
```

[57]:
```
# Fitting the model
from sklearn.linear_model import LinearRegression
lm = LinearRegression()
lm.fit(x_train, y_train)

# predict prices of X_test
y_pred = lm.predict(x_test)
```

0.1.5 R2 Score

[58]:
```
import sklearn.metrics
r2 = sklearn.metrics.r2_score(y_test, y_pred)
print(r2)
```

```
0.67767446576476
```

0.2 Cross-Validation for Multiple Linear Regression

K-Fold Cross-Validation

[59]:
```
# k-fold CV (using all the 13 variables)
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold
lm = LinearRegression()
scores = cross_val_score(lm, x_train, y_train, scoring='r2', cv=5)
scores
```

[59]:
```
array([0.64469087, 0.70115779, 0.61726174, 0.66130751, 0.59038153])
```

[60]:
```
# the other way of doing the same thing (more explicit)
# create a KFold object with 5 splits
folds = KFold(n_splits = 5, shuffle = True, random_state = 100)
scores = cross_val_score(lm, x_train, y_train, scoring='r2', cv=folds)
scores
```

[60]:
```
array([0.59246218, 0.69380963, 0.67252609, 0.62082293, 0.60796825])
```

[61]:
```
# can tune other metrics, such as MSE
scores = cross_val_score(lm, x_train, y_train,␣
 ↪scoring='neg_mean_squared_error', cv=5)
scores
```

[61]:
```
array([-0.01115667, -0.00765919, -0.00606751, -0.00748722, -0.01390484])
```