assignment4

April 26, 2024

[ ]:
```
import pandas as pd
import matplotlib.pyplot as plt
```

```
/home/oneautumleaf/.local/lib/python3.10/site-
packages/matplotlib/projections/__init__.py:63: UserWarning: Unable to import
Axes3D. This may be due to multiple versions of Matplotlib being installed (e.g.
as a system package and as a pip package). As a result, the 3D projection is not
available.
  warnings.warn("Unable to import Axes3D. This may be due to multiple versions
of "
```

[ ]:
```
df = pd.read_excel('./datasets/Real estate valuation data set.xlsx',␣
↪index_col="No")
```

[ ]:
```
df.head()
```

[ ]:
```
    X1 transaction date  X2 house age  X3 distance to the nearest MRT station  \
No
1           2012.916667          32.0                                84.87882
2           2012.916667          19.5                               306.59470
3           2013.583333          13.3                               561.98450
4           2013.500000          13.3                               561.98450
5           2012.833333           5.0                               390.56840

    X4 number of convenience stores  X5 latitude  X6 longitude  \
No
1                                10     24.98298     121.54024
2                                 9     24.98034     121.53951
3                                 5     24.98746     121.54391
4                                 5     24.98746     121.54391
5                                 5     24.97937     121.54245

    Y house price of unit area
No
1                         37.9
2                         42.2
3                         47.3
4                         54.8
```

```
5                          43.1
```

[ ]:
```
df.columns
```

[ ]:
```
Index(['X1 transaction date', 'X2 house age',
       'X3 distance to the nearest MRT station',
       'X4 number of convenience stores', 'X5 latitude', 'X6 longitude',
       'Y house price of unit area'],
      dtype='object')
```

[ ]:
```
print(f"Number of null values: {df.isnull().sum()}")
```

```
Number of null values: X1 transaction date                       0
X2 house age                              0
X3 distance to the nearest MRT station    0
X4 number of convenience stores           0
X5 latitude                               0
X6 longitude                              0
Y house price of unit area                0
dtype: int64
```

[ ]:
```
df.dtypes
```

[ ]:
```
X1 transaction date                       float64
X2 house age                              float64
X3 distance to the nearest MRT station    float64
X4 number of convenience stores             int64
X5 latitude                               float64
X6 longitude                              float64
Y house price of unit area                float64
dtype: object
```

0.0.1 Standardization

[ ]:
```
from sklearn.preprocessing import StandardScaler
```

[ ]:
```
standard_scaler = StandardScaler()
```

[ ]:
```
df_preprocessed = pd.DataFrame(
    standard_scaler.fit_transform(df), columns=df.columns)
```

[ ]:
```
print(f"Min:\n{df_preprocessed.min()}")
print('-------------------')
print(f"Max:\n{df_preprocessed.max()}")
print('-------------------')
print(f"Mean:\n{df_preprocessed.mean()}")
print('-------------------')
```

```
print(f"Std:\n{df_preprocessed.std()}")
print('-------------------')
```

```
Min:
X1 transaction date                        -1.712334
X2 house age                               -1.556639
X3 distance to the nearest MRT station    -0.841279
X4 number of convenience stores           -1.391638
X5 latitude                               -2.981805
X6 longitude                              -3.903223
Y house price of unit area                -2.235474
dtype: float64
------------------
Max:
X1 transaction date                        1.542244
X2 house age                               2.292652
X3 distance to the nearest MRT station     4.287008
X4 number of convenience stores            2.007407
X5 latitude                                3.675611
X6 longitude                               2.146891
Y house price of unit area                 5.851328
dtype: float64
------------------
Mean:
X1 transaction date                       -5.030522e-13
X2 house age                              -9.225042e-17
X3 distance to the nearest MRT station   -1.265762e-16
X4 number of convenience stores          -7.508755e-18
X5 latitude                                1.343166e-13
X6 longitude                              -1.614537e-12
Y house price of unit area                 8.581434e-17
dtype: float64
------------------
Std:
X1 transaction date                        1.00121
X2 house age                               1.00121
X3 distance to the nearest MRT station     1.00121
X4 number of convenience stores            1.00121
X5 latitude                                1.00121
X6 longitude                               1.00121
Y house price of unit area                 1.00121
dtype: float64
------------------
```

[ ]:
```
target_col = df.columns[6]
print(f"Target col: {target_col}")
X = df_preprocessed.drop(target_col, axis=1)
```

```
y = df_preprocessed[target_col]
```

```
Target col: Y house price of unit area
```

0.0.2 Train test split

[ ]:
```
from sklearn.model_selection import train_test_split

# Split the data into train and test sets (80% train, 30% test)
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=42)
```

[ ]:
```
print(f"X:{X.shape}\ntrain:{X_train.shape}\ntest:{X_test.shape}")
print("-----------------")
print(f"y:{y.shape}\ntrain:{y_train.shape}\ntest:{y_test.shape}")
```

```
X:(414, 6)
train:(289, 6)
test:(125, 6)
-----------------
y:(414,)
train:(289,)
test:(125,)
```

0.0.3 Train a linear regression model

[ ]:
```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, accuracy_score
```

[ ]:
```
for col in X_train.columns:
    single_feature_model = LinearRegression()
    single_feature_model.fit(X_train[[col]], y_train)
    print(f"Single Feature Model for {col}: ")
    print(f"Coefficient: {single_feature_model.coef_}")
    print(f"Intercept: {single_feature_model.intercept_}")
    train_error = mean_squared_error(
        y_train, single_feature_model.predict(X_train[[col]]))
    test_error = mean_squared_error(
        y_test, single_feature_model.predict(X_test[[col]]))
    print(f"Training Error: {train_error}")
    print(f"Testing Error: {test_error}")
    print("-----------------")
```

```
Single Feature Model for X1 transaction date:
Coefficient: [0.08241592]
Intercept: 0.029547786728138084
Training Error: 1.0302492484417247
Testing Error: 0.9076674792392131
```

```
-----------------
Single Feature Model for X2 house age:
Coefficient: [-0.20866208]
Intercept: 0.02914724748335111
Training Error: 0.9935035509361515
Testing Error: 0.8709965095606861
-----------------
Single Feature Model for X3 distance to the nearest MRT station:
Coefficient: [-0.69854448]
Intercept: 0.024025853151548482
Training Error: 0.5601043569794457
Testing Error: 0.5181750722242255
-----------------
Single Feature Model for X4 number of convenience stores:
Coefficient: [0.57665432]
Intercept: 0.017846499394452476
Training Error: 0.6951622711093801
Testing Error: 0.6260790308861538
-----------------
Single Feature Model for X5 latitude:
Coefficient: [0.55512985]
Intercept: 0.009076046888492957
Training Error: 0.7235531795501517
Testing Error: 0.651206134273995
-----------------
Single Feature Model for X6 longitude:
Coefficient: [0.5445728]
Intercept: 0.02871072243613803
Training Error: 0.7500513982012522
Testing Error: 0.6751910739071075
-----------------
```

[ ]:
```
multiple_features_model = LinearRegression()
multiple_features_model.fit(X_train, y_train)
```

[ ]: LinearRegression()

[ ]:
```
train_error_multiple = mean_squared_error(
    y_train, multiple_features_model.predict(X_train))
test_error_multiple = mean_squared_error(
    y_test, multiple_features_model.predict(X_test))
print("Training Error (MSE):", train_error_multiple)
print("Test Error (MSE):", test_error_multiple)
```

```
Training Error (MSE): 0.42912127728573446
Test Error (MSE): 0.398320264438555
```

[ ]:
```
multiple_features_model.coef_
```

[ ]:
```
array([ 0.12124468, -0.20308428, -0.47665578,  0.23261738,  0.21807861,
       -0.05891929])
```

[ ]:
```
multiple_features_model.intercept_
```

[ ]:
```
0.00026615452713350113
```

a. **General Multiple Linear Regression Equation:** In multiple linear regression, the relationship between the dependent variable (Y) and multiple independent variables (X1, X2, …, Xp) is modeled as a linear combination of the independent variables, each weighted by a coefficient:

   The general multiple linear regression equation can be represented as:

   `Y = b0 + b1*X1 + b2*X2 + ... + bp*Xp + e`

   - Y is the dependent variable (also known as the response variable or target variable).
   - X1, X2, …, Xp are the independent variables (also known as predictor variables or features).
   - b0 is the intercept term (the value of Y when all independent variables are zero).
   - b1, b2, …, bp are the coefficients (also known as regression coefficients or parameters), representing the change in Y for a one-unit change in the corresponding independent variable, holding other variables constant.
   - e is the error term, representing the difference between the observed value of Y and the value predicted by the model. It captures the effects of all other factors not included in the model.

b. **Concept of Dummy Variable and Calculation:**
   - A dummy variable (also known as an indicator variable) is a binary variable used to represent categorical data in regression analysis.
   - It takes on the value of 0 or 1 to indicate the absence or presence of a particular category, respectively.
   - Dummy variables are necessary in linear regression because categorical variables cannot be directly included in the model equation. Linear regression assumes numerical inputs.
   - To convert a nominal variable with k categories into dummy variables, we create (k-1) dummy variables. Each dummy variable represents one category of the nominal variable.
   - For example, if we have a nominal variable "Color" with categories {Red, Green, Blue}, we create two dummy variables: "Green" and "Blue". If both of these dummy variables are 0, it implies that the color is "Red".

c. **Assumptions in Linear Regression:** Linear regression relies on several key assumptions for its validity:
   1. **Linearity**: The relationship between the dependent variable and independent variables is linear.
   2. **Independence**: The residuals (the differences between observed and predicted values) are independent of each other.
   3. **Homoscedasticity**: The variance of the residuals is constant across all levels of the independent variables.

4. **Normality**: The residuals are normally distributed.
5. **No multicollinearity**: There is no perfect multicollinearity between independent variables, meaning that one independent variable should not be perfectly predicted by a linear combination of other independent variables.
6. **No autocorrelation**: The residuals are not correlated with each other (applicable to time series data).

Violations of these assumptions can lead to biased estimates and incorrect inferences drawn from the model. It's important to assess these assumptions when interpreting the results of a linear regression analysis.