27. Data Science – Machine Learning – Lasso & Ridge Regression

Contents

27. Data Science – Machine Learning – Lasso & Ridge Regression

1. Linear Regression

- ✓ Linear Regression is a standard algorithm for regression and it is used to explain the relationship between a two variables.
- ✓ Also called as it's a relationship in between dependent variable and one or more independent variables.

2. Lasso Regression

- ✓ In linear regression with a single input variable, this relationship is a line, and with higher dimensions, this relationship can be hyperplane that connects the input variables to the target variable.
- ✓ The coefficients of the model are found via an optimization process which minimize error.

3. Coefficients can be large

- ✓ A problem with linear regression, the estimated coefficients of the model can become large and may become model unstable
- ✓ One approach to address the stability of regression models is to change the loss function to include additional costs for a model that has large coefficients.

4. L1 penalty

- ✓ **Lasso Regression** is a popular type of regularized linear regression that includes an L1 penalty.
- ✓ It penalize a model based on the sum of the absolute coefficient values. This is called the L1 penalty.
- ✓ An L1 penalty minimizes the size of all coefficients and some coefficients to be minimized to the value zero, effectively removing input features from the model.

5. L1 Regularization

- ✓ A regression model that uses **L1** regularization technique is called **Lasso** Regression.
- ✓ Lasso full form is, **L**east **A**bsolute **S**hrinkage and **S**election **O**perator

- ✓ Minimization objective = LS Obj + α * (sum of the absolute value of coefficients)

6. How to avoid overfitting issue?

- ✓ Regularization is an important concept that is used to avoid overfitting of the data,
- ✓ We can address this issue by using L1 and L2 Regularization
- ✓ Regularization is implemented by adding a “penalty” term to the best fit derived from the trained data, to achieve a *lesser variance*

7. L1 Regularization and L2 Regularization

- ✓ When you have a large number of features in your dataset, some of the Regularization techniques used to address over-fitting.

Program Name

Importing required libraries
demo1.py

```
import numpy as np
import pandas as pd

import warnings
warnings.filterwarnings('ignore')

print("Importing required libraries")
```

Output

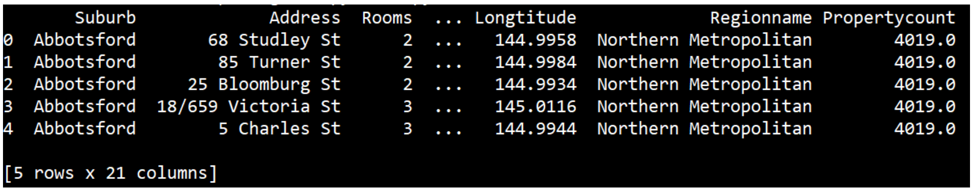
```
Importing required libraries
```

Program Name

Loading the dataset
demo2.py

```
import numpy as np
import pandas as pd

import warnings
warnings.filterwarnings('ignore')

dataset = pd.read_csv('Melbourne_housing_FULL.csv')
print(dataset.head())
```

Output



```
      Suburb           Address  Rooms  ... Longtitude             Regionname Propertycount
0  Abbotsford     68 Studley St      2  ...   144.9958  Northern Metropolitan        4019.0
1  Abbotsford      85 Turner St      2  ...   144.9984  Northern Metropolitan        4019.0
2  Abbotsford   25 Bloomburg St      2  ...   144.9934  Northern Metropolitan        4019.0
3  Abbotsford  18/659 Victoria St      3  ...   145.0116  Northern Metropolitan        4019.0
4  Abbotsford      5 Charles St      3  ...   144.9944  Northern Metropolitan        4019.0

[5 rows x 21 columns]
```

8. Dataset Details: Melbourne house sale price

- ✓ The dataset is about the housing market in Melbourne and contains information about the house sale price
- ✓ Notes on Specific Variables
    - Rooms: Number of rooms
    - Price: Price in dollars
    - Method: S - property sold; SP - property sold prior; PI - property passed in; PN - sold prior not disclosed; SN - sold not disclosed; NB - no bid; VB - vendor bid; W - withdrawn prior to auction; SA - sold after auction; SS - sold after auction price not disclosed. N/A - price or highest bid not available.
    - Type: br - bedroom(s); h - house, cottage, villa, semi, terrace; u - unit, duplex; t - townhouse; dev site - development site; o res - other residential.
    - SellerG: Real Estate Agent
    - Date: Date sold
    - Distance: Distance from CBD
    - Region name: General Region (West, North West, North, North east …etc)
    - Property count: Number of properties that exist in the suburb (an outlying district of a city, especially a residential one.).
    - Bedroom2 : Scraped # of Bedrooms (from different source)
    - Bathroom: Number of Bathrooms
    - Car: Number of cars pots
    - Landsize: Land Size
    - BuildingArea: Building Size
    - Council Area: Governing council for the area

Program Name

Rows and columns in DataFrame
demo3.py

```
import numpy as np
import pandas as pd

import warnings
warnings.filterwarnings('ignore')

dataset = pd.read_csv('Melbourne_housing_FULL.csv')
print("Rows and columns:", dataset.shape)
```

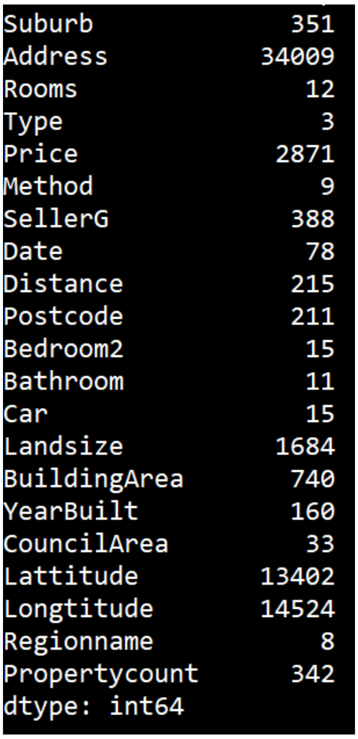
Output

```
Rows and columns: (34857, 21)
```

Program Name
Unique values
demo4.py

```
import numpy as np
import pandas as pd

import warnings
warnings.filterwarnings('ignore')

dataset = pd.read_csv('Melbourne_housing_FULL.csv')
print(dataset.nunique())
```

Output



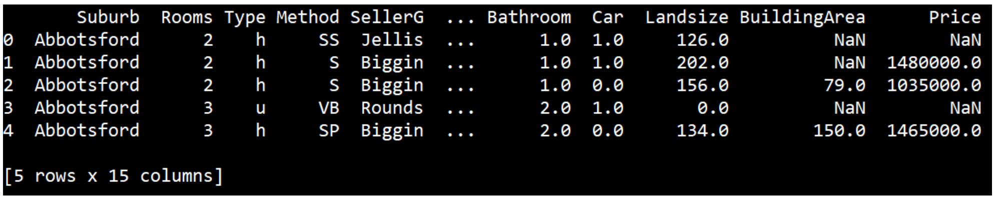
```
Suburb              351
Address           34009
Rooms                12
Type                  3
Price              2871
Method                9
SellerG             388
Date                 78
Distance            215
Postcode            211
Bedroom2             15
Bathroom             11
Car                  15
Landsize           1684
BuildingArea        740
YearBuilt           160
CouncilArea          33
Lattitude         13402
Longtitude        14524
Regionname            8
Propertycount       342
dtype: int64
```

Program Name

Get the required columns
demo5.py

```
import numpy as np
import pandas as pd

import warnings
warnings.filterwarnings('ignore')

dataset = pd.read_csv('Melbourne_housing_FULL.csv')
print(dataset.shape)

cols_to_use = ['Suburb', 'Rooms', 'Type', 'Method', 'SellerG',
'Regionname', 'Propertycount', 'Distance', 'CouncilArea',
'Bedroom2', 'Bathroom', 'Car', 'Landsize', 'BuildingArea', 'Price']

dataset = dataset[cols_to_use]

print(dataset.head())
```

Output



```
       Suburb  Rooms Type Method SellerG  ... Bathroom  Car  Landsize BuildingArea      Price
0  Abbotsford      2    h     SS  Jellis  ...      1.0  1.0     126.0          NaN        NaN
1  Abbotsford      2    h      S  Biggin  ...      1.0  1.0     202.0          NaN  1480000.0
2  Abbotsford      2    h      S  Biggin  ...      1.0  0.0     156.0         79.0  1035000.0
3  Abbotsford      3    u     VB  Rounds  ...      2.0  1.0       0.0          NaN        NaN
4  Abbotsford      3    h     SP  Biggin  ...      2.0  0.0     134.0        150.0  1465000.0

[5 rows x 15 columns]
```

Program Name

Rows and columns in DataFrame
demo6.py

```
import numpy as np
import pandas as pd

import warnings
warnings.filterwarnings('ignore')

dataset = pd.read_csv('Melbourne_housing_FULL.csv')

cols_to_use = ['Suburb', 'Rooms', 'Type', 'Method', 'SellerG',
'Regionname', 'Propertycount', 'Distance', 'CouncilArea',
'Bedroom2', 'Bathroom', 'Car', 'Landsize', 'BuildingArea', 'Price']

dataset = dataset[cols_to_use]
print("Rows and columns:", dataset.shape)
```

Output

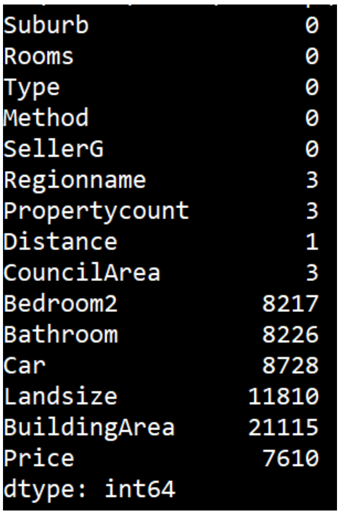
```
Rows and columns: (34857, 15)
```

Program Name

Checking NaN values
demo7.py

```
import numpy as np
import pandas as pd

import warnings
warnings.filterwarnings('ignore')

dataset = pd.read_csv('Melbourne_housing_FULL.csv')

cols_to_use = ['Suburb', 'Rooms', 'Type', 'Method', 'SellerG',
'Regionname', 'Propertycount', 'Distance', 'CouncilArea',
'Bedroom2', 'Bathroom', 'Car', 'Landsize', 'BuildingArea', 'Price']

dataset = dataset[cols_to_use]

print(dataset.isna().sum())
```

Output



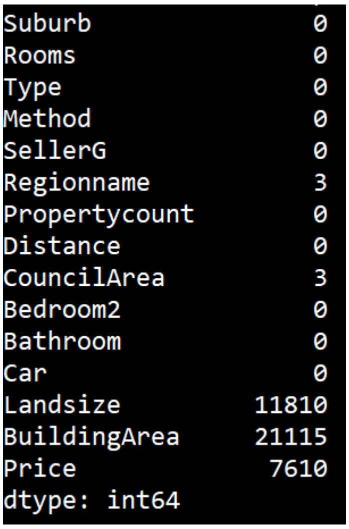
```
Suburb               0
Rooms                0
Type                 0
Method               0
SellerG              0
Regionname           3
Propertycount        3
Distance             1
CouncilArea          3
Bedroom2          8217
Bathroom          8226
Car               8728
Landsize         11810
BuildingArea     21115
Price             7610
dtype: int64
```

Program Name

Few of the columns filling with zero
demo8.py

```
import numpy as np
import pandas as pd

import warnings
warnings.filterwarnings('ignore')

dataset = pd.read_csv('Melbourne_housing_FULL.csv')

cols_to_use = ['Suburb', 'Rooms', 'Type', 'Method', 'SellerG',
'Regionname', 'Propertycount', 'Distance', 'CouncilArea',
'Bedroom2', 'Bathroom', 'Car', 'Landsize', 'BuildingArea', 'Price']

dataset = dataset[cols_to_use]

cols_to_fill_zero = ['Propertycount', 'Distance', 'Bedroom2',
'Bathroom', 'Car']
dataset[cols_to_fill_zero] = dataset[cols_to_fill_zero].fillna(0)

print(dataset.isna().sum())
```

Output

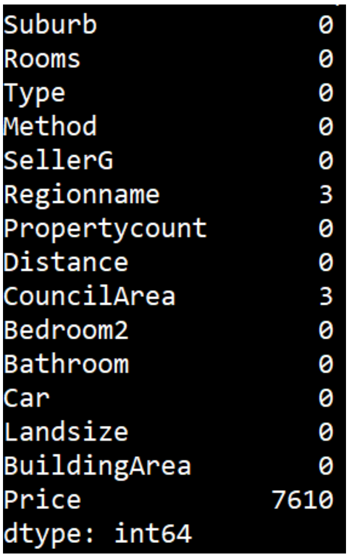


```
Suburb               0
Rooms                0
Type                 0
Method               0
SellerG              0
Regionname           3
Propertycount        0
Distance             0
CouncilArea          3
Bedroom2             0
Bathroom             0
Car                  0
Landsize         11810
BuildingArea     21115
Price             7610
dtype: int64
```

Program Name	Filling Landsize and BuildingArea columns with mean value demo9.py

```
import numpy as np
import pandas as pd

import warnings
warnings.filterwarnings('ignore')

dataset = pd.read_csv('Melbourne_housing_FULL.csv')

cols_to_use = ['Suburb', 'Rooms', 'Type', 'Method', 'SellerG',
'Regionname', 'Propertycount', 'Distance', 'CouncilArea',
'Bedroom2', 'Bathroom', 'Car', 'Landsize', 'BuildingArea', 'Price']

dataset = dataset[cols_to_use]

cols_to_fill_zero = ['Propertycount', 'Distance', 'Bedroom2',
'Bathroom', 'Car']
dataset[cols_to_fill_zero] = dataset[cols_to_fill_zero].fillna(0)

dataset['Landsize'] =
dataset['Landsize'].fillna(dataset.Landsize.mean())

dataset['BuildingArea'] =
dataset['BuildingArea'].fillna(dataset.BuildingArea.mean())

print(dataset.isna().sum())
```

Output

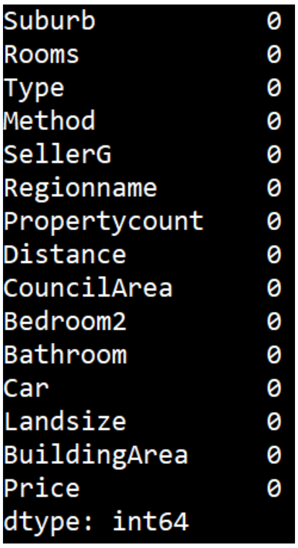


```
Suburb              0
Rooms               0
Type                0
Method              0
SellerG             0
Regionname          3
Propertycount       0
Distance            0
CouncilArea         3
Bedroom2            0
Bathroom            0
Car                 0
Landsize            0
BuildingArea        0
Price            7610
dtype: int64
```

Program Name	Dropping NaN values demo10.py

```
import numpy as np
import pandas as pd

import warnings
warnings.filterwarnings('ignore')

dataset = pd.read_csv('Melbourne_housing_FULL.csv')

cols_to_use = ['Suburb', 'Rooms', 'Type', 'Method', 'SellerG',
'Regionname', 'Propertycount', 'Distance', 'CouncilArea',
'Bedroom2', 'Bathroom', 'Car', 'Landsize', 'BuildingArea', 'Price']

dataset = dataset[cols_to_use]

cols_to_fill_zero = ['Propertycount', 'Distance', 'Bedroom2',
'Bathroom', 'Car']
dataset[cols_to_fill_zero] = dataset[cols_to_fill_zero].fillna(0)

dataset['Landsize'] =
dataset['Landsize'].fillna(dataset.Landsize.mean())

dataset['BuildingArea'] =
dataset['BuildingArea'].fillna(dataset.BuildingArea.mean())

dataset.dropna(inplace = True)

print(dataset.isna().sum())
```

Output



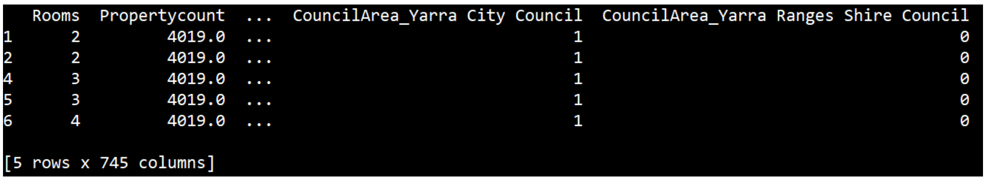
```
Suburb           0
Rooms            0
Type             0
Method           0
SellerG          0
Regionname       0
Propertycount    0
Distance         0
CouncilArea      0
Bedroom2         0
Bathroom         0
Car              0
Landsize         0
BuildingArea     0
Price            0
dtype: int64
```

Program Name

Creating dummy variables for characters data
demo11.py

```
import numpy as np
import pandas as pd

import warnings
warnings.filterwarnings('ignore')

dataset = pd.read_csv('Melbourne_housing_FULL.csv')

cols_to_use = ['Suburb', 'Rooms', 'Type', 'Method', 'SellerG',
'Regionname', 'Propertycount', 'Distance', 'CouncilArea',
'Bedroom2', 'Bathroom', 'Car', 'Landsize', 'BuildingArea', 'Price']

dataset = dataset[cols_to_use]

cols_to_fill_zero = ['Propertycount', 'Distance', 'Bedroom2',
'Bathroom', 'Car']
dataset[cols_to_fill_zero] = dataset[cols_to_fill_zero].fillna(0)

dataset['Landsize'] =
dataset['Landsize'].fillna(dataset.Landsize.mean())

dataset['BuildingArea'] =
dataset['BuildingArea'].fillna(dataset.BuildingArea.mean())

dataset.dropna(inplace = True)

dataset = pd.get_dummies(dataset, drop_first = True)

print(dataset.head())
```

Output



```
   Rooms  Propertycount  ...  CouncilArea_Yarra City Council  CouncilArea_Yarra Ranges Shire Council
1      2         4019.0  ...                               1                                       0
2      2         4019.0  ...                               1                                       0
4      3         4019.0  ...                               1                                       0
5      3         4019.0  ...                               1                                       0
6      4         4019.0  ...                               1                                       0

[5 rows x 745 columns]
```

Program Name

Creating features and labels
demo12.py

```
import numpy as np
import pandas as pd

import warnings
warnings.filterwarnings('ignore')

dataset = pd.read_csv('Melbourne_housing_FULL.csv')

cols_to_use = ['Suburb', 'Rooms', 'Type', 'Method', 'SellerG',
'Regionname', 'Propertycount', 'Distance', 'CouncilArea',
'Bedroom2', 'Bathroom', 'Car', 'Landsize', 'BuildingArea', 'Price']

dataset = dataset[cols_to_use]

cols_to_fill_zero = ['Propertycount', 'Distance', 'Bedroom2',
'Bathroom', 'Car']
dataset[cols_to_fill_zero] = dataset[cols_to_fill_zero].fillna(0)

dataset['Landsize'] =
dataset['Landsize'].fillna(dataset.Landsize.mean())

dataset['BuildingArea'] =
dataset['BuildingArea'].fillna(dataset.BuildingArea.mean())

dataset.dropna(inplace = True)

dataset = pd.get_dummies(dataset, drop_first = True)

X = dataset.drop('Price', axis = 1)
y = dataset['Price']

print("Created features and labels")
```

Output

```
Created features and labels
```

Program Name

Splitting training and testing datasets
demo13.py

```
from sklearn.model_selection import train_test_split
import numpy as np
import pandas as pd

import warnings
warnings.filterwarnings('ignore')

dataset = pd.read_csv('Melbourne_housing_FULL.csv')

cols_to_use = ['Suburb', 'Rooms', 'Type', 'Method', 'SellerG',
'Regionname', 'Propertycount', 'Distance', 'CouncilArea',
'Bedroom2', 'Bathroom', 'Car', 'Landsize', 'BuildingArea', 'Price']

dataset = dataset[cols_to_use]

cols_to_fill_zero = ['Propertycount', 'Distance', 'Bedroom2',
'Bathroom', 'Car']
dataset[cols_to_fill_zero] = dataset[cols_to_fill_zero].fillna(0)

dataset['Landsize'] =
dataset['Landsize'].fillna(dataset.Landsize.mean())

dataset['BuildingArea'] =
dataset['BuildingArea'].fillna(dataset.BuildingArea.mean())

dataset.dropna(inplace = True)

dataset = pd.get_dummies(dataset, drop_first = True)

X = dataset.drop('Price', axis = 1)
y = dataset['Price']
```

```
train_X, test_X, train_y, test_y = train_test_split(X, y, test_size =
0.3, random_state=2)

print("Splitting train and test datasets")
```

Output

```
Splitting train and test datasets
```

Program Name	Creating LinearRegression model and training demo14.py

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression

import numpy as np
import pandas as pd

import warnings
warnings.filterwarnings('ignore')

dataset = pd.read_csv('Melbourne_housing_FULL.csv')

cols_to_use = ['Suburb', 'Rooms', 'Type', 'Method', 'SellerG',
'Regionname', 'Propertycount', 'Distance', 'CouncilArea',
'Bedroom2', 'Bathroom', 'Car', 'Landsize', 'BuildingArea', 'Price']

dataset = dataset[cols_to_use]

cols_to_fill_zero = ['Propertycount', 'Distance', 'Bedroom2',
'Bathroom', 'Car']
dataset[cols_to_fill_zero] = dataset[cols_to_fill_zero].fillna(0)

dataset['Landsize'] =
dataset['Landsize'].fillna(dataset.Landsize.mean())

dataset['BuildingArea'] =
dataset['BuildingArea'].fillna(dataset.BuildingArea.mean())

dataset.dropna(inplace = True)

dataset = pd.get_dummies(dataset, drop_first = True)

X = dataset.drop('Price', axis = 1)
y = dataset['Price']

train_X, test_X, train_y, test_y = train_test_split(X, y, test_size =
0.3, random_state=2)

reg = LinearRegression()

print("Created LinearRegression model and training")

reg.fit(train_X, train_y)
```

Output

```
Created LinearRegression model
```

Program Name

Creating LinearRegression model
demo15.py

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression

import numpy as np
import pandas as pd

import warnings
warnings.filterwarnings('ignore')

dataset = pd.read_csv('Melbourne_housing_FULL.csv')

cols_to_use = ['Suburb', 'Rooms', 'Type', 'Method', 'SellerG',
'Regionname', 'Propertycount', 'Distance', 'CouncilArea',
'Bedroom2', 'Bathroom', 'Car', 'Landsize', 'BuildingArea', 'Price']

dataset = dataset[cols_to_use]

cols_to_fill_zero = ['Propertycount', 'Distance', 'Bedroom2',
'Bathroom', 'Car']
dataset[cols_to_fill_zero] = dataset[cols_to_fill_zero].fillna(0)

dataset['Landsize'] =
dataset['Landsize'].fillna(dataset.Landsize.mean())

dataset['BuildingArea'] =
dataset['BuildingArea'].fillna(dataset.BuildingArea.mean())

dataset.dropna(inplace = True)

dataset = pd.get_dummies(dataset, drop_first = True)

X = dataset.drop('Price', axis = 1)
y = dataset['Price']
```

```
train_X, test_X, train_y, test_y = train_test_split(X, y, test_size =
0.3, random_state=2)

reg = LinearRegression()

reg.fit(train_X, train_y)

print("Training LinearRegression model")
```

Output

```
Training LinearRegression model
```

Program Name

Linear Regression: Training, training dataset score
demo16.py

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression

import numpy as np
import pandas as pd

import warnings
warnings.filterwarnings('ignore')

dataset = pd.read_csv('Melbourne_housing_FULL.csv')

cols_to_use = ['Suburb', 'Rooms', 'Type', 'Method', 'SellerG',
'Regionname', 'Propertycount', 'Distance', 'CouncilArea',
'Bedroom2', 'Bathroom', 'Car', 'Landsize', 'BuildingArea', 'Price']

dataset = dataset[cols_to_use]

cols_to_fill_zero = ['Propertycount', 'Distance', 'Bedroom2',
'Bathroom', 'Car']
dataset[cols_to_fill_zero] = dataset[cols_to_fill_zero].fillna(0)

dataset['Landsize'] =
dataset['Landsize'].fillna(dataset.Landsize.mean())

dataset['BuildingArea'] =
dataset['BuildingArea'].fillna(dataset.BuildingArea.mean())

dataset.dropna(inplace = True)

dataset = pd.get_dummies(dataset, drop_first = True)

X = dataset.drop('Price', axis = 1)
y = dataset['Price']
```

```
train_X, test_X, train_y, test_y = train_test_split(X, y, test_size =
0.3, random_state=2)

reg = LinearRegression()
reg.fit(train_X, train_y)

print("Training dataset score is:")
print(reg.score(train_X, train_y))
```

Output

```
Training dataset score is:
0.6827792395792723
```

Program Name Linear Regression: Training, test dataset score
demo17.py

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression

import numpy as np
import pandas as pd

import warnings
warnings.filterwarnings('ignore')

dataset = pd.read_csv('Melbourne_housing_FULL.csv')

cols_to_use = ['Suburb', 'Rooms', 'Type', 'Method', 'SellerG',
'Regionname', 'Propertycount', 'Distance', 'CouncilArea',
'Bedroom2', 'Bathroom', 'Car', 'Landsize', 'BuildingArea', 'Price']

dataset = dataset[cols_to_use]

cols_to_fill_zero = ['Propertycount', 'Distance', 'Bedroom2',
'Bathroom', 'Car']
dataset[cols_to_fill_zero] = dataset[cols_to_fill_zero].fillna(0)

dataset['Landsize'] =
dataset['Landsize'].fillna(dataset.Landsize.mean())

dataset['BuildingArea'] =
dataset['BuildingArea'].fillna(dataset.BuildingArea.mean())

dataset.dropna(inplace = True)

dataset = pd.get_dummies(dataset, drop_first = True)

X = dataset.drop('Price', axis = 1)
y = dataset['Price']
```

```
train_X, test_X, train_y, test_y = train_test_split(X, y, test_size =
0.3, random_state=2)

reg = LinearRegression()
reg.fit(train_X, train_y)

print("Creating Linear Regression model and training with test
dataset:")
print(reg.score(test_X, test_y))
```

Output

```
Creating Linear Regression model and training with test dataset
0.1385368316157145
```

Note

- ✓ If training score is very good and test score is very low then it called as over fit

Program Name

Lasso Regression
demo18.py

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import Lasso

import numpy as np
import pandas as pd

import warnings
warnings.filterwarnings('ignore')

dataset = pd.read_csv('Melbourne_housing_FULL.csv')

cols_to_use = ['Suburb', 'Rooms', 'Type', 'Method', 'SellerG',
'Regionname', 'Propertycount', 'Distance', 'CouncilArea',
'Bedroom2', 'Bathroom', 'Car', 'Landsize', 'BuildingArea', 'Price']

dataset = dataset[cols_to_use]

cols_to_fill_zero = ['Propertycount', 'Distance', 'Bedroom2',
'Bathroom', 'Car']
dataset[cols_to_fill_zero] = dataset[cols_to_fill_zero].fillna(0)

dataset['Landsize'] =
dataset['Landsize'].fillna(dataset.Landsize.mean())

dataset['BuildingArea'] =
dataset['BuildingArea'].fillna(dataset.BuildingArea.mean())

dataset.dropna(inplace = True)

dataset = pd.get_dummies(dataset, drop_first = True)

X = dataset.drop('Price', axis = 1)
y = dataset['Price']
```

```
train_X, test_X, train_y, test_y = train_test_split(X, y, test_size =
0.3, random_state=2)

lasso_reg = Lasso(alpha = 50, max_iter = 100, tol = 0.1)
lasso_reg.fit(train_X, train_y)

print("Creating Lasso Regression model and training with train
dataset")
print(lasso_reg.score(train_X, train_y))
```

Output

```
Creating Lasso Regression model and training with train dataset
0.6766985624766824
```

Program Name	Lasso Regression: Training, testing dataset score demo19.py

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import Lasso

import numpy as np
import pandas as pd

import warnings
warnings.filterwarnings('ignore')

dataset = pd.read_csv('Melbourne_housing_FULL.csv')

cols_to_use = ['Suburb', 'Rooms', 'Type', 'Method', 'SellerG',
'Regionname', 'Propertycount', 'Distance', 'CouncilArea',
'Bedroom2', 'Bathroom', 'Car', 'Landsize', 'BuildingArea', 'Price']

dataset = dataset[cols_to_use]

cols_to_fill_zero = ['Propertycount', 'Distance', 'Bedroom2',
'Bathroom', 'Car']
dataset[cols_to_fill_zero] = dataset[cols_to_fill_zero].fillna(0)

dataset['Landsize'] =
dataset['Landsize'].fillna(dataset.Landsize.mean())

dataset['BuildingArea'] =
dataset['BuildingArea'].fillna(dataset.BuildingArea.mean())

dataset.dropna(inplace = True)

dataset = pd.get_dummies(dataset, drop_first = True)

X = dataset.drop('Price', axis = 1)
y = dataset['Price']
```

```
train_X, test_X, train_y, test_y = train_test_split(X, y, test_size =
0.3, random_state=2)

lasso_reg = Lasso(alpha = 50, max_iter = 100, tol = 0.1)
lasso_reg.fit(train_X, train_y)

print("Creating Lasso Regression model checking test dataset
score")
print(lasso_reg.score(test_X, test_y))
```

Output

```
Creating Lasso Regression model checking test dataset score
0.6636111369404489
```

9. Ridge Regression

- ✓ In linear regression with a single input variable, this relationship is a line, and with higher dimensions, this relationship can be hyperplane that connects the input variables to the target variable.
- ✓ The coefficients of the model are found via an optimization process which minimize error.

10. Coefficients can be large

- ✓ A problem with linear regression, the estimated coefficients of the model can become large and may become model unstable
- ✓ One approach to address the stability of regression models is to change the loss function to include additional costs for a model that has large coefficients.

11. L2 penalty

- ✓ **Ridge Regression** is a popular type of regularized linear regression that includes an **L2** penalty.
- ✓ It penalize a model based on sum of the squared coefficient value. This is called the L2 penalty.
- ✓ An L2 penalty minimizes the size of all coefficients and some coefficients to be minimized to the value zero, effectively removing input features from the model.

- ✓ Minimization objective = LS Obj + α * (sum of square of coefficients)

Program Name

Ridge Regression: Training, training dataset score
demo20.py

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import Ridge

import numpy as np
import pandas as pd

import warnings
warnings.filterwarnings('ignore')

dataset = pd.read_csv('Melbourne_housing_FULL.csv')

cols_to_use = ['Suburb', 'Rooms', 'Type', 'Method', 'SellerG',
'Regionname', 'Propertycount', 'Distance', 'CouncilArea',
'Bedroom2', 'Bathroom', 'Car', 'Landsize', 'BuildingArea', 'Price']

dataset = dataset[cols_to_use]

cols_to_fill_zero = ['Propertycount', 'Distance', 'Bedroom2',
'Bathroom', 'Car']
dataset[cols_to_fill_zero] = dataset[cols_to_fill_zero].fillna(0)

dataset['Landsize'] =
dataset['Landsize'].fillna(dataset.Landsize.mean())

dataset['BuildingArea'] =
dataset['BuildingArea'].fillna(dataset.BuildingArea.mean())

dataset.dropna(inplace = True)

dataset = pd.get_dummies(dataset, drop_first = True)

X = dataset.drop('Price', axis = 1)
y = dataset['Price']
```

```
train_X, test_X, train_y, test_y = train_test_split(X, y, test_size =
0.3, random_state=2)

ridge_reg = Ridge(alpha = 50, max_iter = 100, tol = 0.1)
ridge_reg.fit(train_X, train_y)

print("Ridge Regression model score with train dataset:")
print(ridge_reg.score(train_X, train_y))
```

Output

```
Ridge Regression model score with train dataset:
0.6670848945194958
```

Program Name

Ridge Regression: Training, testing dataset score
demo21.py

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import Ridge

import numpy as np
import pandas as pd

import warnings
warnings.filterwarnings('ignore')

dataset = pd.read_csv('Melbourne_housing_FULL.csv')

cols_to_use = ['Suburb', 'Rooms', 'Type', 'Method', 'SellerG',
'Regionname', 'Propertycount', 'Distance', 'CouncilArea',
'Bedroom2', 'Bathroom', 'Car', 'Landsize', 'BuildingArea', 'Price']

dataset = dataset[cols_to_use]

cols_to_fill_zero = ['Propertycount', 'Distance', 'Bedroom2',
'Bathroom', 'Car']
dataset[cols_to_fill_zero] = dataset[cols_to_fill_zero].fillna(0)

dataset['Landsize'] =
dataset['Landsize'].fillna(dataset.Landsize.mean())

dataset['BuildingArea'] =
dataset['BuildingArea'].fillna(dataset.BuildingArea.mean())

dataset.dropna(inplace = True)

dataset = pd.get_dummies(dataset, drop_first = True)

X = dataset.drop('Price', axis = 1)
y = dataset['Price']
```

```
train_X, test_X, train_y, test_y = train_test_split(X, y, test_size =
0.3, random_state=2)

ridge_reg = Ridge(alpha=50, max_iter=100, tol=0.1)
ridge_reg.fit(train_X, train_y)

print("Ridge Regression model score with test dataset:")
print(ridge_reg.score(test_X, test_y))
```

Output

```
Ridge Regression model score with test dataset:
0.6670848945194958
```