Least_Square_Method

August 19, 2022

[1]:
```
import pandas as pd
import numpy as np
from sklearn.datasets import load_boston
import statsmodels.api as sm
import statsmodels.formula.api as smf
```

[2]:
```
my_data = load_boston()
```

[3]:
```
print(my_data.DESCR)
```

```
.. _boston_dataset:

Boston house prices dataset
---------------------------

**Data Set Characteristics:**

    :Number of Instances: 506

    :Number of Attributes: 13 numeric/categorical predictive. Median Value
(attribute 14) is usually the target.

    :Attribute Information (in order):
        - CRIM     per capita crime rate by town
        - ZN       proportion of residential land zoned for lots over 25,000
sq.ft.
        - INDUS    proportion of non-retail business acres per town
        - CHAS     Charles River dummy variable (= 1 if tract bounds river; 0
otherwise)
        - NOX      nitric oxides concentration (parts per 10 million)
        - RM       average number of rooms per dwelling
        - AGE      proportion of owner-occupied units built prior to 1940
        - DIS      weighted distances to five Boston employment centres
        - RAD      index of accessibility to radial highways
        - TAX      full-value property-tax rate per $10,000
        - PTRATIO  pupil-teacher ratio by town
        - B        1000(Bk - 0.63)^2 where Bk is the proportion of black people
by town
```

```
        - LSTAT    % lower status of the population
        - MEDV     Median value of owner-occupied homes in $1000's

    :Missing Attribute Values: None

    :Creator: Harrison, D. and Rubinfeld, D.L.

This is a copy of UCI ML housing dataset.
https://archive.ics.uci.edu/ml/machine-learning-databases/housing/


This dataset was taken from the StatLib library which is maintained at Carnegie
Mellon University.

The Boston house-price data of Harrison, D. and Rubinfeld, D.L. 'Hedonic
prices and the demand for clean air', J. Environ. Economics & Management,
vol.5, 81-102, 1978.   Used in Belsley, Kuh & Welsch, 'Regression diagnostics
…', Wiley, 1980.   N.B. Various transformations are used in the table on
pages 244-261 of the latter.

The Boston house-price data has been used in many machine learning papers that
address regression
problems.

.. topic:: References

   - Belsley, Kuh & Welsch, 'Regression diagnostics: Identifying Influential
Data and Sources of Collinearity', Wiley, 1980. 244-261.
   - Quinlan,R. (1993). Combining Instance-Based and Model-Based Learning. In
Proceedings on the Tenth International Conference of Machine Learning, 236-243,
University of Massachusetts, Amherst. Morgan Kaufmann.
```

[4]:
```python
features = pd.DataFrame(my_data.data, columns=my_data.feature_names)
features.head()
```

[4]:
```
      CRIM    ZN  INDUS  CHAS    NOX     RM   AGE     DIS  RAD    TAX  \
0  0.00632  18.0   2.31   0.0  0.538  6.575  65.2  4.0900  1.0  296.0
1  0.02731   0.0   7.07   0.0  0.469  6.421  78.9  4.9671  2.0  242.0
2  0.02729   0.0   7.07   0.0  0.469  7.185  61.1  4.9671  2.0  242.0
3  0.03237   0.0   2.18   0.0  0.458  6.998  45.8  6.0622  3.0  222.0
4  0.06905   0.0   2.18   0.0  0.458  7.147  54.2  6.0622  3.0  222.0

   PTRATIO       B  LSTAT
0     15.3  396.90   4.98
1     17.8  396.90   9.14
2     17.8  392.83   4.03
```

```
3     18.7  394.63   2.94
4     18.7  396.90   5.33
```

[5]:
```
target = pd.DataFrame(my_data.target, columns=['Target'])
target.head()
```

[5]:
```
   Target
0    24.0
1    21.6
2    34.7
3    33.4
4    36.2
```

[6]:
```
#The baseline model using mean of the target
```

[7]:
```
mean_output = target['Target'].mean()
mean_output
```

[7]:
```
22.532806324110698
```

[8]:
```
#Sum squared Error
squared_error = pd.Series(mean_output-target['Target'])**2
print(squared_error)
```

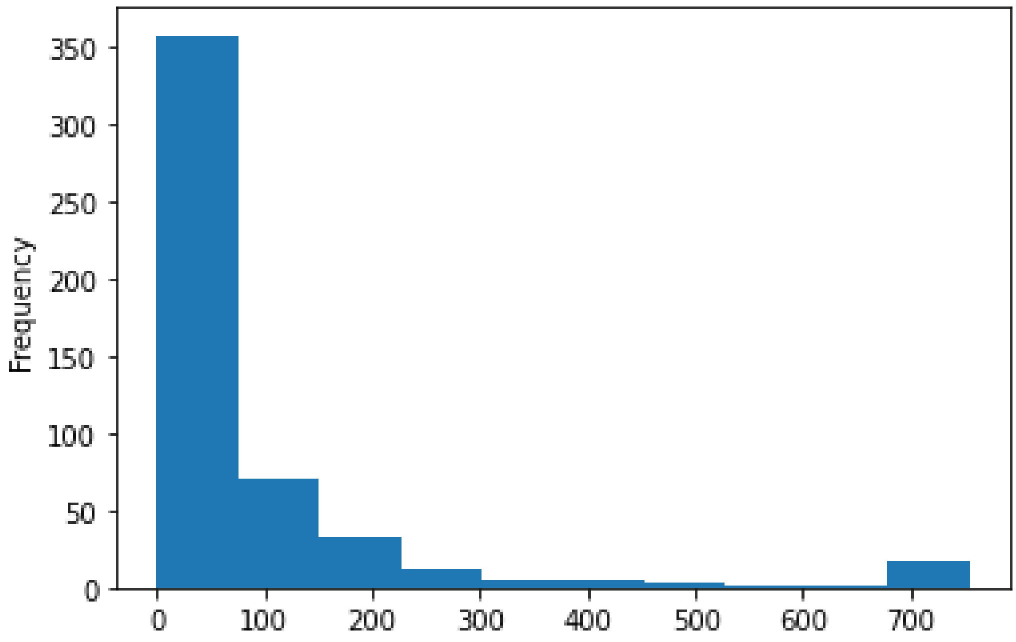
```
0        2.152657
1        0.870128
2      148.040602
3      118.095898
4      186.792183
          …
501      0.017638
502      3.735740
503      1.869219
504      0.283883
505    113.056570
Name: Target, Length: 506, dtype: float64
```

[9]:
```
sum_squared_error = np.sum(squared_error)
print(sum_squared_error)
```

```
42716.29541501976
```

[10]:
```
squared_error.plot(kind = 'hist')
```

[10]:
```
<AxesSubplot:ylabel='Frequency'>
```



[12]:
```python
ad_df=pd.read_csv('Advertising.csv')
ad_df
```

[12]:
```
     Unnamed: 0     TV  radio  newspaper  sales
0             1  230.1   37.8       69.2   22.1
1             2   44.5   39.3       45.1   10.4
2             3   17.2   45.9       69.3    9.3
3             4  151.5   41.3       58.5   18.5
4             5  180.8   10.8       58.4   12.9
..          …      …    …          …    …
195         196   38.2    3.7       13.8    7.6
196         197   94.2    4.9        8.1    9.7
197         198  177.0    9.3        6.4   12.8
198         199  283.6   42.0       66.2   25.5
199         200  232.1    8.6        8.7   13.4

[200 rows x 5 columns]
```

[11]:
```python
## Using the Least Square method
```

[13]:
```python
X = ad_df[['TV']]
y = ad_df[['sales']]
```

[14]:
```python
print(X.shape, y.shape)
```

```
(200, 1) (200, 1)
```

[15]:
```
X = sm.add_constant(X)
```

[16]:
```
X.head()
```

[16]:

	const	TV
0	1.0	230.1
1	1.0	44.5
2	1.0	17.2
3	1.0	151.5
4	1.0	180.8

[17]:
```
linear_regression = sm.OLS(y,X)
```

[18]:
```
#Estimation of coefficients
model = linear_regression.fit()
```

[20]:
```
#In case if you want to use smf
# linear_regression = smf.ols(formula='target ~ RM', data=data)
# model = linear_regression.fit()
```

[19]:
```
model.summary()
```

[19]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:                  sales   R-squared:                       0.612
Model:                            OLS   Adj. R-squared:                  0.610
Method:                 Least Squares   F-statistic:                     312.1
Date:                Fri, 01 Apr 2022   Prob (F-statistic):           1.47e-42
Time:                        03:06:59   Log-Likelihood:                -519.05
No. Observations:                 200   AIC:                             1042.
Df Residuals:                     198   BIC:                             1049.
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          7.0326      0.458     15.360      0.000       6.130       7.935
TV             0.0475      0.003     17.668      0.000       0.042       0.053
==============================================================================
Omnibus:                        0.531   Durbin-Watson:                   1.935
Prob(Omnibus):                  0.767   Jarque-Bera (JB):                0.669
Skew:                          -0.089   Prob(JB):                        0.716
Kurtosis:                       2.779   Cond. No.                         338.
==============================================================================
```

```
Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
"""
```

0.0.1 Let us compare the results

[22]:
```python
predicted_values = model.predict(X)
```

[23]:
```python
SSE_on_mean = np.sum((target['Target'] - target['Target'].mean())**2)
SSE_on_OLS = np.sum((target['Target'] - predicted_values)**2)
```

[24]:
```python
print('Error using Mean : ',SSE_on_mean)
print('Error using OLS : ', SSE_on_OLS)
```

```
Error using Mean :  42716.29541501976
Error using OLS :  22061.879196211798
```

[26]:
```python
# Guess this is what ???
(SSE_on_mean-SSE_on_OLS)/SSE_on_mean
```

[26]: 0.4835254559913341

[ ]:
```python

```