Linear Regression End to End Script

September 28, 2020

[1]:
```
import pandas as pd
```

[2]:
```
cd
```

```
/home/labsuser
```

[3]:
```
cd Downloads
```

```
/home/labsuser/Downloads
```

[4]:
```
cement = pd.read_csv('concrete.csv')
```

[5]:
```
cement.head()
```

[5]:

	cement	slag	ash	water	superplastic	coarseagg	fineagg	age	strength
0	141.3	212.0	0.0	203.5	0.0	971.8	748.5	28	29.89
1	168.9	42.2	124.3	158.3	10.8	1080.8	796.2	14	23.51
2	250.0	0.0	95.7	187.4	5.5	956.9	861.2	28	29.22
3	266.0	114.0	0.0	228.0	0.0	932.0	670.0	28	45.85
4	154.8	183.4	0.0	193.3	9.1	1047.4	696.7	28	18.29

[109]:
```
X = cement_n.iloc[:,:4]
```

[86]:
```
y = cement_n.iloc[:,8:]
```

[51]:
```
X.head()
```

[51]:

	0	1	2	3	4	5	6
0	0.111269	0.166942	0.000000	0.160249	0.000000	0.765257	0.589416
1	0.123388	0.030829	0.090806	0.115645	0.007890	0.789569	0.581657
2	0.188137	0.000000	0.072019	0.141027	0.004139	0.720111	0.648093

```
3  0.220434  0.094472  0.000000  0.188944  0.000000  0.772349  0.555229
4  0.119480  0.141555  0.000000  0.149196  0.007024  0.808423  0.537739

          7
0  0.022049
1  0.010228
2  0.021071
3  0.023204
4  0.021611
```

[52]:
```
y.head()
```

[52]:
```
          8
0  0.023537
1  0.017175
2  0.021989
3  0.037996
4  0.014117
```

[110]:
```
from sklearn.model_selection import train_test_split

X_train,X_test,y_train,y_test = train_test_split(X,y, test_size = 0.2)
```

[95]:
```
from sklearn import linear_model

regr = linear_model.LinearRegression()
```

[111]:
```
#Training
regr.fit(X_train, y_train)
```

[111]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

[112]:
```
# The coefficients
print('Coefficients', regr.coef_)
print('Intercept',regr.intercept_)
```

```
Coefficients [[ 0.11936135  0.09484179  0.08090056 -0.14642297]]
Intercept [0.01322175]
```

[113]:
```
y_pred = regr.predict(X_test)
```

[83]:
```
y_pred
```

[83]:
```
array([[0.02753935],
       [0.01839669],
       [0.02477642],
       [0.04175916],
```

```
[0.02261538],
[0.04453115],
[0.02318381],
[0.03835459],
[0.0291453 ],
[0.02183154],
[0.04422213],
[0.03253533],
[0.01732013],
[0.02356899],
[0.02962872],
[0.04130679],
[0.02914709],
[0.02633743],
[0.021495  ],
[0.03414829],
[0.02347208],
[0.01757589],
[0.02625112],
[0.04154245],
[0.02053684],
[0.01853658],
[0.02263757],
[0.02730573],
[0.02361925],
[0.01652775],
[0.03068795],
[0.03217146],
[0.01784505],
[0.03578143],
[0.03017195],
[0.00738421],
[0.02033366],
[0.01589186],
[0.01375828],
[0.03081666],
[0.02243427],
[0.02695569],
[0.03127836],
[0.01451407],
[0.01566377],
[0.02161391],
[0.02723661],
[0.03670276],
[0.01963822],
[0.01468843],
[0.03149   ],
```

```
[0.01566257],
[0.02776941],
[0.03398618],
[0.04044695],
[0.03679075],
[0.05405293],
[0.03446678],
[0.02240015],
[0.02435314],
[0.03248075],
[0.01890045],
[0.02196203],
[0.04465708],
[0.05094896],
[0.01889282],
[0.04828824],
[0.01278381],
[0.03335898],
[0.02468988],
[0.0341344 ],
[0.01861665],
[0.02051053],
[0.01982148],
[0.01823564],
[0.02971793],
[0.02049265],
[0.02971165],
[0.03152152],
[0.03628562],
[0.02809098],
[0.04853507],
[0.02247581],
[0.01947087],
[0.02245706],
[0.03296799],
[0.0194362 ],
[0.02887897],
[0.03017289],
[0.02566747],
[0.02363377],
[0.03062014],
[0.01727048],
[0.03826103],
[0.02750522],
[0.03678579],
[0.03391597],
[0.02302863],
```

```
[0.02597563],
[0.02285109],
[0.02825553],
[0.02867894],
[0.01155664],
[0.04359901],
[0.03420264],
[0.01758178],
[0.02169238],
[0.04259999],
[0.04372918],
[0.02644906],
[0.03480479],
[0.02756074],
[0.01527543],
[0.02098225],
[0.04386329],
[0.03179675],
[0.03726591],
[0.03725727],
[0.02962043],
[0.02519256],
[0.01746061],
[0.02639518],
[0.02730573],
[0.01947727],
[0.0140498 ],
[0.022383  ],
[0.03554897],
[0.04899068],
[0.0288697 ],
[0.04205344],
[0.03311058],
[0.02948007],
[0.02369116],
[0.01688421],
[0.04161757],
[0.02802926],
[0.02439535],
[0.02813527],
[0.0401629 ],
[0.01749928],
[0.04274389],
[0.0191102 ],
[0.0145004 ],
[0.05177618],
[0.03101332],
```

```
[0.01292252],
[0.03827074],
[0.03964922],
[0.02142795],
[0.01954502],
[0.01958879],
[0.03892701],
[0.01751103],
[0.02788933],
[0.00552655],
[0.01492409],
[0.02710529],
[0.02517559],
[0.03711575],
[0.04021217],
[0.02061668],
[0.02341012],
[0.02554105],
[0.0343345 ],
[0.01945414],
[0.03302424],
[0.04853507],
[0.02093871],
[0.02307454],
[0.01590065],
[0.02525016],
[0.01533456],
[0.04274389],
[0.02578389],
[0.03839317],
[0.01500859],
[0.02462993],
[0.01791819],
[0.04128897],
[0.02332139],
[0.02308052],
[0.01426544],
[0.04955058],
[0.03310636],
[0.04052587],
[0.02753436],
[0.02730857],
[0.03029008],
[0.02761281],
[0.02352191],
[0.01985804],
[0.02321656],
```

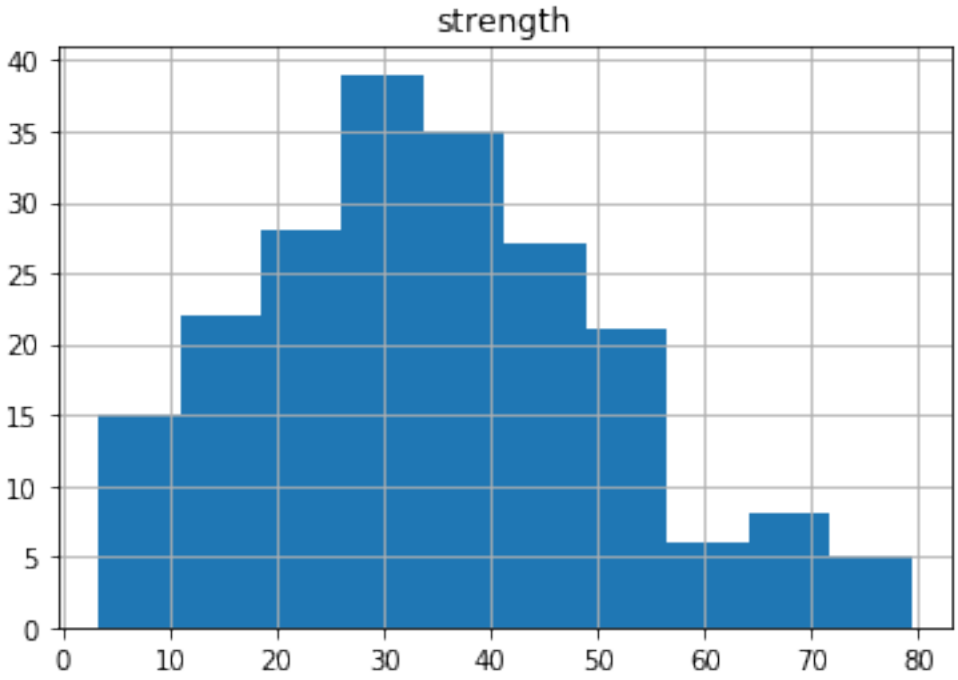
```
       [0.01552952],
       [0.02647785],
       [0.02556681],
       [0.02521882],
       [0.02031351],
       [0.02696305],
       [0.02129574],
       [0.03103917],
       [0.03398316],
       [0.04840944],
       [0.02390218],
       [0.0222183 ],
       [0.0438202 ],
       [0.01876431]])
```

[114]:
```
from sklearn.metrics import mean_squared_error, r2_score
print(r2_score(y_test, y_pred)) #Coefficient of Determination
print(mean_squared_error(y_test, y_pred)) #MSE
```

```
0.31408365194888666
0.00010533002545944778
```

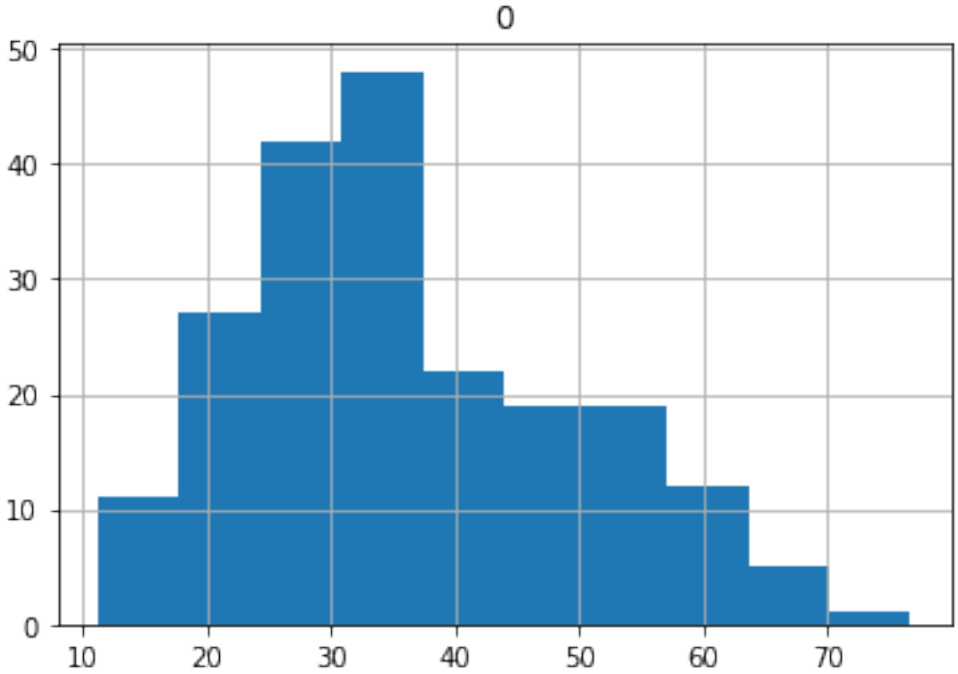
[28]:
```
y_test.hist()
```

[28]:
```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x1a19a0f208>]],
      dtype=object)
```



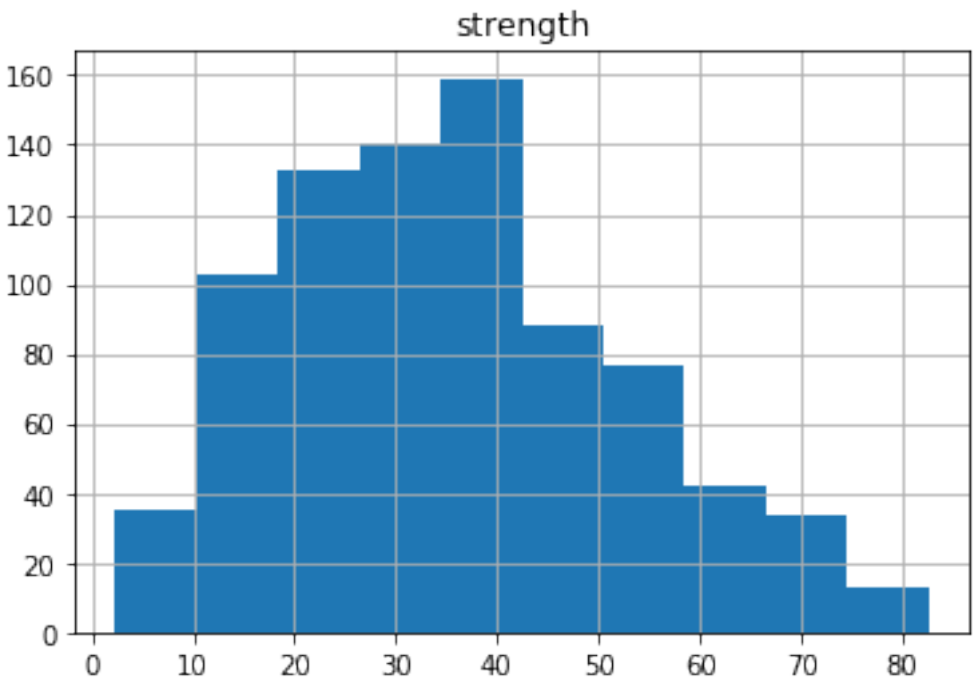
[29]:
```
pd.DataFrame(y_pred).hist()
```

[29]:
```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x1a19b2f898>]],
      dtype=object)
```



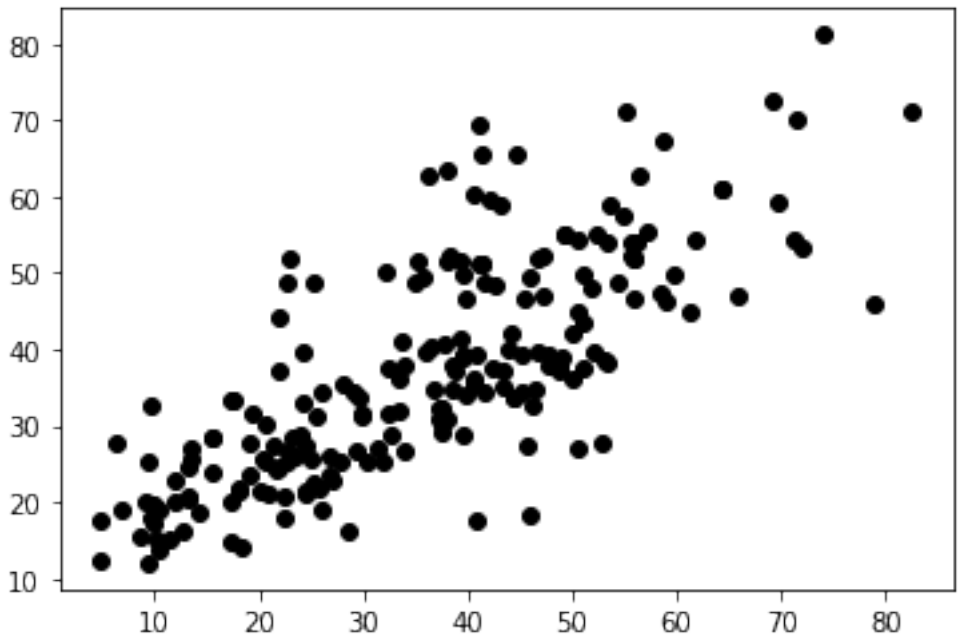
[20]:
```
y_train.hist()
```

[20]:
```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x1a19a0f278>]],
      dtype=object)
```



[45]:
```
import matplotlib.pyplot as plt
# Plot outputs
plt.scatter(y_test, y_pred,  color='black')
```

[45]: <matplotlib.collections.PathCollection at 0x1a19f52e80>



[48]:
```
from sklearn.preprocessing import Normalizer

cement_n = Normalizer().fit_transform(cement)
cement_n = pd.DataFrame(cement_n)
```

[63]:
```
#Multi-Collinearity
import numpy as np
X = np.array(X)
from statsmodels.stats.outliers_influence import variance_inflation_factor

vif = pd.DataFrame()
vif = [variance_inflation_factor(X, i) for i in range(X.shape[1])]
print(vif)
X = pd.DataFrame(X)
```

```
[15.229736835039063, 3.442254211877823, 4.155409480198613, 79.71718322439241,
5.426055391925735, 84.67589012649955, 72.23838841153335, 1.718995020540424]
```

[65]:
```
X = X.iloc[:, [1,2,4,7]]
```

[66]:
```
X.head()
```

[66]:

	1	2	4	7
0	0.166942	0.000000	0.000000	0.022049
1	0.030829	0.090806	0.007890	0.010228
2	0.000000	0.072019	0.004139	0.021071
3	0.094472	0.000000	0.000000	0.023204
4	0.141555	0.000000	0.007024	0.021611

[76]:
```
#Feature Selection
from sklearn.feature_selection import RFE

logreg = linear_model.LinearRegression()
rfe = RFE(logreg, 6)
rfe = rfe.fit(X_train, y_train)

print(rfe.support_)
print(rfe.ranking_)
```

```
[ True  True False False  True  True  True  True]
[1 1 3 2 1 1 1 1]
```

```
/Users/kanth/anaconda3/lib/python3.7/site-
packages/sklearn/utils/validation.py:724: DataConversionWarning: A column-vector
y was passed when a 1d array was expected. Please change the shape of y to
```

```
(n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

[77]:
```
d = rfe.support_

g = X.columns

a = g[d]

X_f = X[a]
```

[78]:
```
X_f.head()
```

[78]:

	0	1	4	5	6	7
0	0.111269	0.166942	0.000000	0.765257	0.589416	0.022049
1	0.123388	0.030829	0.007890	0.789569	0.581657	0.010228
2	0.188137	0.000000	0.004139	0.720111	0.648093	0.021071
3	0.220434	0.094472	0.000000	0.772349	0.555229	0.023204
4	0.119480	0.141555	0.007024	0.808423	0.537739	0.021611

[ ]: