MLR_ASN_SVG

August 15, 2020

[1]:
```
# Importing required packages
import numpy as np
import pandas as pd
import matplotlib
import matplotlib.pyplot as plt
import statsmodels.api as sm
from sklearn.model_selection import train_test_split
from statsmodels.stats.outliers_influence import variance_inflation_factor
from sklearn import metrics
from statsmodels.graphics.regressionplots import influence_plot
```

[2]:
```
# Reading the CSV and creating a dataframe
ASN_df = pd.read_csv("C:/Venu/UCI DataSets/Airfoil Self Noise.csv")
ASN_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1503 entries, 0 to 1502
Data columns (total 6 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   Frequency        1503 non-null   int64  
 1   Angle of Attack  1503 non-null   float64
 2   Chord Length     1503 non-null   float64
 3   FS Velocity      1503 non-null   float64
 4   SSD Thickness    1503 non-null   float64
 5   SSP Level        1503 non-null   float64
dtypes: float64(5), int64(1)
memory usage: 70.6 KB
```

[3]:
```
#Displaying Summary Statistics
ASN_df.describe()
```

[3]:

	Frequency	Angle of Attack	Chord Length	FS Velocity	\
count	1503.000000	1503.000000	1503.000000	1503.000000	
mean	2886.380572	6.782302	0.136548	50.860745	
std	3152.573137	5.918128	0.093541	15.572784	
min	200.000000	0.000000	0.025400	31.700000	
25%	800.000000	2.000000	0.050800	39.600000	

```
50%     1600.000000          5.400000      0.101600    39.600000
75%     4000.000000          9.900000      0.228600    71.300000
max    20000.000000         22.200000      0.304800    71.300000

       SSD Thickness    SSP Level
count    1503.000000  1503.000000
mean        0.011140   124.835943
std         0.013150     6.898657
min         0.000401   103.380000
25%         0.002535   120.191000
50%         0.004957   125.721000
75%         0.015576   129.995500
max         0.058411   140.987000
```

[4]:
```
#Obtaining the correlation matrix
ASN_df.corr()
```

[4]:
```
                 Frequency  Angle of Attack  Chord Length  FS Velocity  \
Frequency         1.000000        -0.272765     -0.003661     0.133664
Angle of Attack  -0.272765         1.000000     -0.504868     0.058760
Chord Length     -0.003661        -0.504868      1.000000     0.003787
FS Velocity       0.133664         0.058760      0.003787     1.000000
SSD Thickness    -0.230107         0.753394     -0.220842    -0.003974
SSP Level        -0.390711        -0.156108     -0.236162     0.125103

                 SSD Thickness  SSP Level
Frequency            -0.230107  -0.390711
Angle of Attack       0.753394  -0.156108
Chord Length         -0.220842  -0.236162
FS Velocity          -0.003974   0.125103
SSD Thickness         1.000000  -0.312670
SSP Level            -0.312670   1.000000
```

[5]:
```
#Creating the feature dataset
X_features = list(ASN_df.columns)
X_features.remove("SSP Level")
X_features
```

[5]:
```
['Frequency',
 'Angle of Attack',
 'Chord Length',
 'FS Velocity',
 'SSD Thickness']
```

[8]:
```
#Dataframe of features
X_df = ASN_df[X_features]
X_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1503 entries, 0 to 1502
Data columns (total 5 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   Frequency        1503 non-null   int64  
 1   Angle of Attack  1503 non-null   float64
 2   Chord Length     1503 non-null   float64
 3   FS Velocity      1503 non-null   float64
 4   SSD Thickness    1503 non-null   float64
dtypes: float64(4), int64(1)
memory usage: 58.8 KB
```

[10]:
```
#Response Variable
y = ASN_df["SSP Level"]
y = y.to_frame()
```

[19]:
```
#adding constant
X_df = sm.add_constant(X_df)
X_df[0:5]
```

[19]:

	const	Frequency	Angle of Attack	Chord Length	FS Velocity	SSD Thickness
0	1.0	800	0.0	0.3048	71.3	0.002663
1	1.0	1000	0.0	0.3048	71.3	0.002663
2	1.0	1250	0.0	0.3048	71.3	0.002663
3	1.0	1600	0.0	0.3048	71.3	0.002663
4	1.0	2000	0.0	0.3048	71.3	0.002663

[20]:
```
#creating training and testing data set
train_X,test_X,train_y,test_y = train_test_split(X_df,y,train_size=0.
↪7,random_state=42)
```

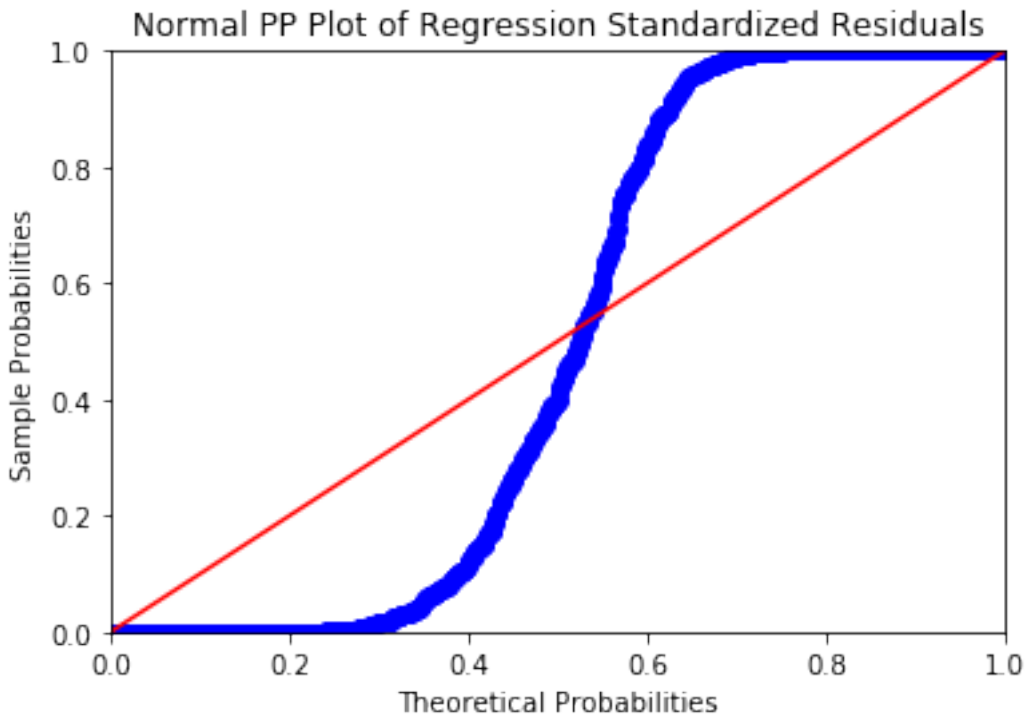
[21]:
```
#Regression using Statmodels API
ASN_reg1 = sm.OLS(train_y,train_X).fit()
ASN_reg1.summary2()
```

[21]:
```
<class 'statsmodels.iolib.summary2.Summary'>
"""
                    Results: Ordinary least squares
=================================================================
Model:                OLS              Adj. R-squared:     0.519
Dependent Variable:   SSP Level        AIC:                6288.0403
Date:                 2020-08-15 16:28 BIC:                6317.7910
No. Observations:     1052             Log-Likelihood:     -3138.0
Df Model:             5                F-statistic:        227.9
Df Residuals:         1046             Prob (F-statistic): 1.38e-164
R-squared:            0.521            Scale:              22.958
```

```
------------------------------------------------------------------------------
                 Coef.    Std.Err.      t      P>|t|     [0.025     0.975]
------------------------------------------------------------------------------
const            132.4778    0.6514 203.3671 0.0000   131.1996   133.7560
Frequency         -0.0013    0.0000 -26.4566 0.0000    -0.0014    -0.0012
Angle of Attack   -0.3653    0.0471  -7.7544 0.0000    -0.4577    -0.2728
Chord Length     -34.6170    1.9282 -17.9526 0.0000   -38.4006   -30.8333
FS Velocity        0.1000    0.0096  10.3652 0.0000     0.0811     0.1189
SSD Thickness   -157.2017   18.2741  -8.6024 0.0000  -193.0597  -121.3437
------------------------------------------------------------------------------
Omnibus:                9.824          Durbin-Watson:              2.122
Prob(Omnibus):          0.007          Jarque-Bera (JB):          14.313
Skew:                  -0.029          Prob(JB):                   0.001
Kurtosis:               3.569          Condition No.:             546543
==============================================================================
* The condition number is large (5e+05). This might indicate
strong multicollinearity or other numerical problems.
"""
```

[22]:
```
#Normality Assuption check using P-P plot
ASN_resid = ASN_reg1.resid
probplot = sm.ProbPlot(ASN_resid)
plt.figure(figsize=(8,6))
probplot.ppplot(line='45')
plt.title("Normal PP Plot of Regression Standardized Residuals")
plt.savefig("ASN_resid.png")
plt.show(block=True)
```

```
<Figure size 576x432 with 0 Axes>
```



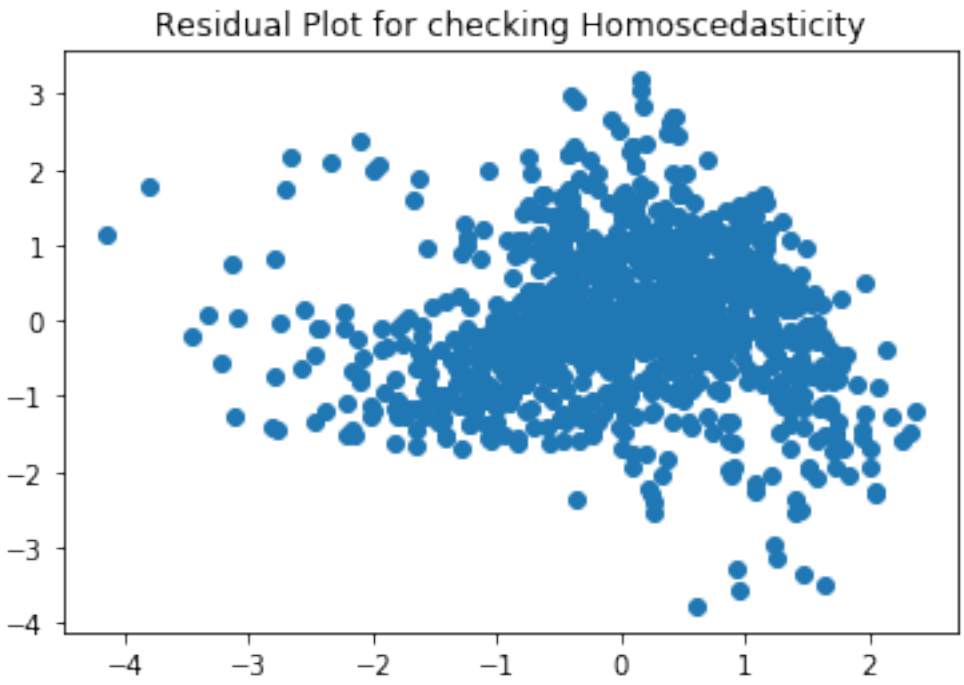
[25]:
```
# Calculating RMSE of the validation set
np.sqrt(metrics.mean_squared_error(ASN_reg1_pred_y,test_y))
```

[25]: 4.867017391586268

[27]:
```
# calculating R-squared value
np.round(metrics.r2_score(ASN_reg1_pred_y,test_y),3)
```

[27]: -0.092

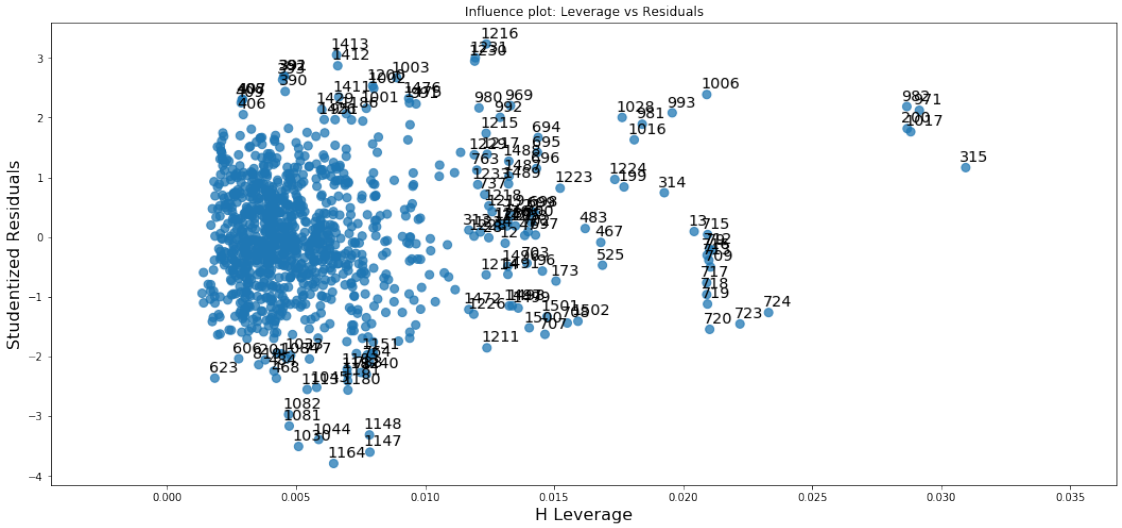
[24]:
```
#Test of Homoscedasticity
def get_standardized_values(vals):
    return(vals-vals.mean())/vals.std()
plt.scatter(get_standardized_values(ASN_reg1.
 ↪fittedvalues),get_standardized_values(ASN_reg1.resid))
plt.title("Residual Plot for checking Homoscedasticity")
plt.show()
```



[32]:
```
#Leverage points
fig,ax = plt.subplots(figsize=(18,8))
influence_plot(ASN_reg1,ax=ax,size = 8)
plt.title("Influence plot: Leverage vs Residuals")
plt.show()
k = train_X.shape[1]
n = train_X.shape[0]

print("var#\n",k)
print("n#\n",n)

leverage_cutoff = 3*((k+1)/n)
print("leverage Value: \n", leverage_cutoff)
```




```
var#
 6
n#
 1052
leverage Value:
 0.019961977186311784
```

[36]:
```
#Multicollinearity
X_df
ASN1_df = X_df.drop(["const"],axis =1)
ASN1_df
ASN1_df_cor= ASN1_df.corr()
ASN1_df_cor
```

[36]:

	Frequency	Angle of Attack	Chord Length	FS Velocity	\
Frequency	1.000000	-0.272765	-0.003661	0.133664	
Angle of Attack	-0.272765	1.000000	-0.504868	0.058760	
Chord Length	-0.003661	-0.504868	1.000000	0.003787	
FS Velocity	0.133664	0.058760	0.003787	1.000000	
SSD Thickness	-0.230107	0.753394	-0.220842	-0.003974	

	SSD Thickness
Frequency	-0.230107
Angle of Attack	0.753394
Chord Length	-0.220842
FS Velocity	-0.003974
SSD Thickness	1.000000

[41]:
```
vif_df=pd.DataFrame(np.linalg.inv(ASN1_df.corr().values), index = ASN1_df_cor.
 ↪index, columns=ASN1_df_cor.columns)
vif = vif_df.to_numpy()
VIF = vif.diagonal()
VIF
```

[41]:
```
array([1.14444379, 3.44165752, 1.5107543 , 1.04169841, 2.53212699])
```

[43]:
```
#Multicollinearity
VIF_df = pd.DataFrame([ASN1_df_cor.columns,VIF])
VIF_df
```

[43]:

	0	1	2	3	4
0	Frequency	Angle of Attack	Chord Length	FS Velocity	SSD Thickness
1	1.14444	3.44166	1.51075	1.0417	2.53213

[42]:
```
#Calculating Prediction intervals
from statsmodels.sandbox.regression.predstd import wls_prediction_std
_,ASN_reg1_pred_y_low,ASN_reg1_pred_y_high =␣
 ↪wls_prediction_std(ASN_reg1,test_X,alpha=0.1)
ASN_reg1_pred_y_df = pd.DataFrame({ 'Predicted SSP Level': ASN_reg1_pred_y,␣
 ↪"Low": ASN_reg1_pred_y_low,"High": ASN_reg1_pred_y_high})
ASN_reg1_pred_y_df[0:10]
```

[42]:

	Predicted SSP Level	Low	High
51	124.066490	116.154498	131.978482
1465	119.464139	111.539009	127.389269
184	120.332541	112.423649	128.241433
1000	129.608868	121.690241	137.527495
746	129.786125	121.875727	137.696522
1032	132.644234	124.736703	140.551765
925	122.434904	114.527394	130.342414
394	126.749830	118.844189	134.655471
597	130.010457	122.107896	137.913017
588	125.982737	118.082887	133.882587

[ ]:
```

```