complete-analysis

July 19, 2024

[ ]:
```
# Prepared By:
# Muhammad Yasir Rafiq
# Ph.D (Management Sceiences)
# Liaoning Technical University, China
```

[1]:
```
# Import a data file
import pandas as pd
data=pd.read_excel('mtcars.xlsx')
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32 entries, 0 to 31
Data columns (total 12 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Model   32 non-null     object
 1   mpg     32 non-null     float64
 2   cyl     32 non-null     int64
 3   disp    32 non-null     float64
 4   hp      32 non-null     int64
 5   drat    32 non-null     float64
 6   wt      32 non-null     float64
 7   qsec    32 non-null     float64
 8   vs      32 non-null     int64
 9   am      32 non-null     int64
 10  gear    32 non-null     int64
 11  carb    32 non-null     int64
dtypes: float64(5), int64(6), object(1)
memory usage: 3.1+ KB
```

[2]:
```
# Show top few lines
data.head()
```

[2]:
```
               Model   mpg  cyl   disp   hp  drat     wt   qsec  vs  am  gear  \
0          Mazda RX4  21.0    6  160.0  110  3.90  2.620  16.46   0   1     4
1      Mazda RX4 Wag  21.0    6  160.0  110  3.90  2.875  17.02   0   1     4
2         Datsun 710  22.8    4  108.0   93  3.85  2.320  18.61   1   1     4
3     Hornet 4 Drive  21.4    6  258.0  110  3.08  3.215  19.44   1   0     3
```

```
4  Hornet Sportabout  18.7    8  360.0  175  3.15  3.440  17.02   0   0    3

   carb
0     4
1     4
2     1
3     1
4     2
```

[4]:
```
# # Show last few lines
data.tail(2)
```

[4]:
```
            Model   mpg  cyl   disp   hp  drat    wt  qsec  vs  am  gear  carb
30  Maserati Bora  15.0    8  301.0  335  3.54  3.57  14.6   0   1     5     8
31     Volvo 142E  21.4    4  121.0  109  4.11  2.78  18.6   1   1     4     2
```

[5]:
```
# # Show the Column headings
data.columns
```

[5]:
```
Index(['Model', 'mpg', 'cyl', 'disp', 'hp', 'drat', 'wt', 'qsec', 'vs', 'am',
       'gear', 'carb'],
      dtype='object')
```

[6]:
```
# # Show missing
data.isna().sum()
```

[6]:
```
Model    0
mpg      0
cyl      0
disp     0
hp       0
drat     0
wt       0
qsec     0
vs       0
am       0
gear     0
carb     0
dtype: int64
```

[7]:
```
# Selecting particular rows and columns
# Method-1 loc - col names ()
# Method-2 iloc - positions []
```

[8]:
```
data.loc[:, ('mpg','wt','disp')]
```

[8]:

	mpg	wt	disp
0	21.0	2.620	160.0
1	21.0	2.875	160.0
2	22.8	2.320	108.0
3	21.4	3.215	258.0
4	18.7	3.440	360.0
5	18.1	3.460	225.0
6	14.3	3.570	360.0
7	24.4	3.190	146.7
8	22.8	3.150	140.8
9	19.2	3.440	167.6
10	17.8	3.440	167.6
11	16.4	4.070	275.8
12	17.3	3.730	275.8
13	15.2	3.780	275.8
14	10.4	5.250	472.0
15	10.4	5.424	460.0
16	14.7	5.345	440.0
17	32.4	2.200	78.7
18	30.4	1.615	75.7
19	33.9	1.835	71.1
20	21.5	2.465	120.1
21	15.5	3.520	318.0
22	15.2	3.435	304.0
23	13.3	3.840	350.0
24	19.2	3.845	400.0
25	27.3	1.935	79.0
26	26.0	2.140	120.3
27	30.4	1.513	95.1
28	15.8	3.170	351.0
29	19.7	2.770	145.0
30	15.0	3.570	301.0
31	21.4	2.780	121.0

[9]:

```
data.iloc[:,[1,6,3]]
```

[9]:

	mpg	wt	disp
0	21.0	2.620	160.0
1	21.0	2.875	160.0
2	22.8	2.320	108.0
3	21.4	3.215	258.0
4	18.7	3.440	360.0
5	18.1	3.460	225.0
6	14.3	3.570	360.0
7	24.4	3.190	146.7
8	22.8	3.150	140.8
9	19.2	3.440	167.6

10	17.8	3.440	167.6
11	16.4	4.070	275.8
12	17.3	3.730	275.8
13	15.2	3.780	275.8
14	10.4	5.250	472.0
15	10.4	5.424	460.0
16	14.7	5.345	440.0
17	32.4	2.200	78.7
18	30.4	1.615	75.7
19	33.9	1.835	71.1
20	21.5	2.465	120.1
21	15.5	3.520	318.0
22	15.2	3.435	304.0
23	13.3	3.840	350.0
24	19.2	3.845	400.0
25	27.3	1.935	79.0
26	26.0	2.140	120.3
27	30.4	1.513	95.1
28	15.8	3.170	351.0
29	19.7	2.770	145.0
30	15.0	3.570	301.0
31	21.4	2.780	121.0

[10]:
```
data.loc[(data.cyl==6) & (data.mpg<20),('mpg','wt')]
```

[10]:

	mpg	wt
5	18.1	3.46
9	19.2	3.44
10	17.8	3.44
29	19.7	2.77

[11]:
```
# # Univariate Analysis
data.mpg.mean().round(2)
```

[11]: 20.09

[12]:
```
print("Mean is", data.mpg.mean().round(2))
```

```
Mean is 20.09
```

[13]:
```
print("Std dev is",data.mpg.std().round(2))
```

```
Std dev is 6.03
```

[14]:
```
print("Var is",data.mpg.var().round(2))
```

```
Var is 36.32
```

[15]:
```
print("Min is",data.mpg.min().round(2))
```

```
Min is 10.4
```

[16]:
```
print("Max is",data.mpg.max().round(2))
```

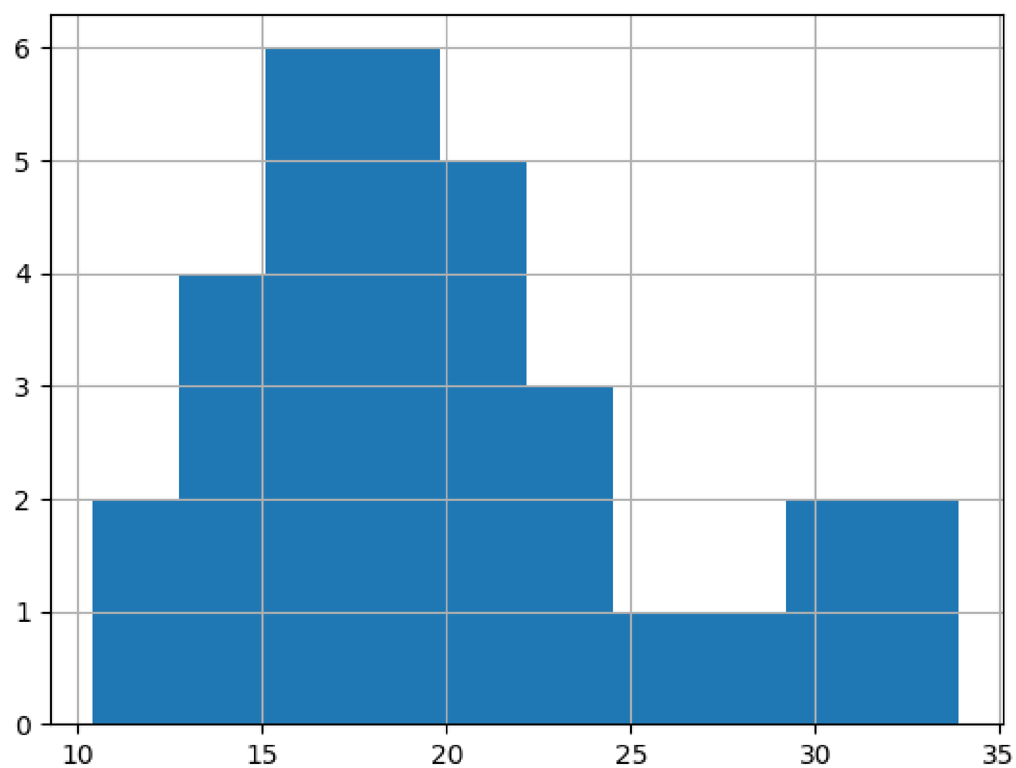
```
Max is 33.9
```

[17]:
```
print("Median is",data.mpg.median().round(2))
```

```
Median is 19.2
```

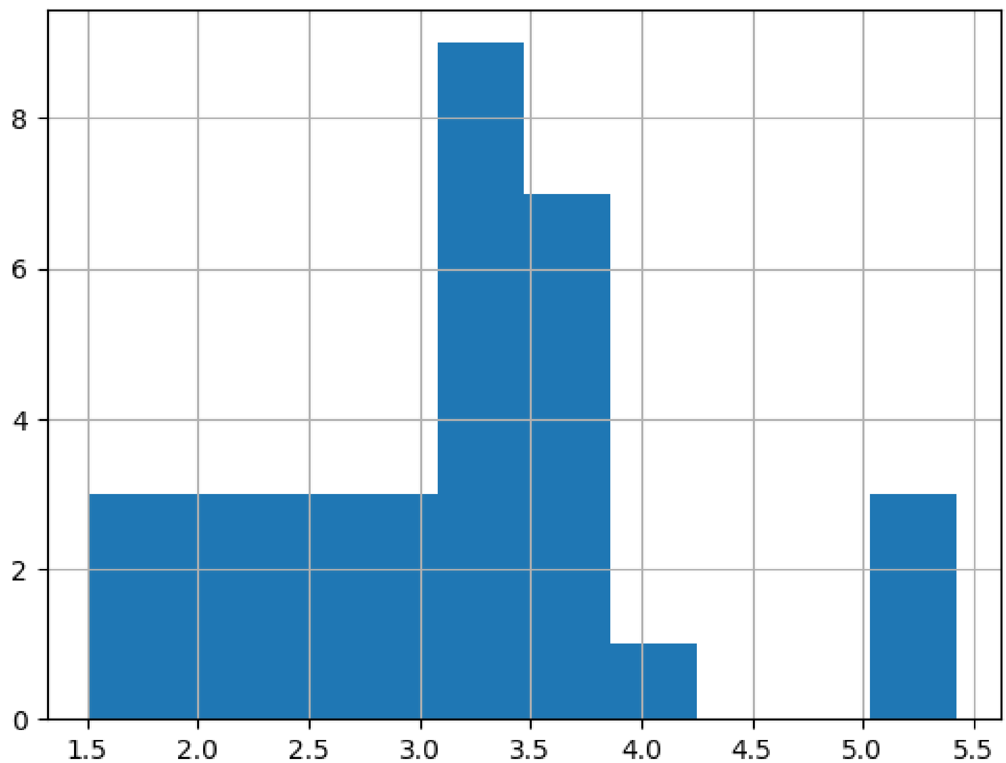
[20]:
```
# # Normality checking
data.mpg.hist()
```

[20]: <Axes: >



[21]:
```
data.wt.hist()
```

[21]: <Axes: >



[23]:

```
# # Statistical test
# Shapiro Wilk Test
# # Statistical test
# Shapiro Wilk Test
from scipy.stats import shapiro # Import the shapiro function
statistic, pvalue = shapiro(data.mpg)
print("Shapiro statistic is", round(statistic,2))
print("Shapiro pvalue is", round(pvalue,2))
```

```
Shapiro statistic is 0.95
Shapiro pvalue is 0.12
```

[24]:

```
# # Skewness and Kurtosis
print("Skewness is", data.mpg.skew().round(2))
print("Kurtosis is", data.mpg.kurt().round(2))
```

```
Skewness is 0.67
Kurtosis is -0.02
```

[25]:

```
## Bivariate Analysis
z=data.loc[:,('mpg','wt',"hp",'disp','drat')]
z.corr().round(2)
```

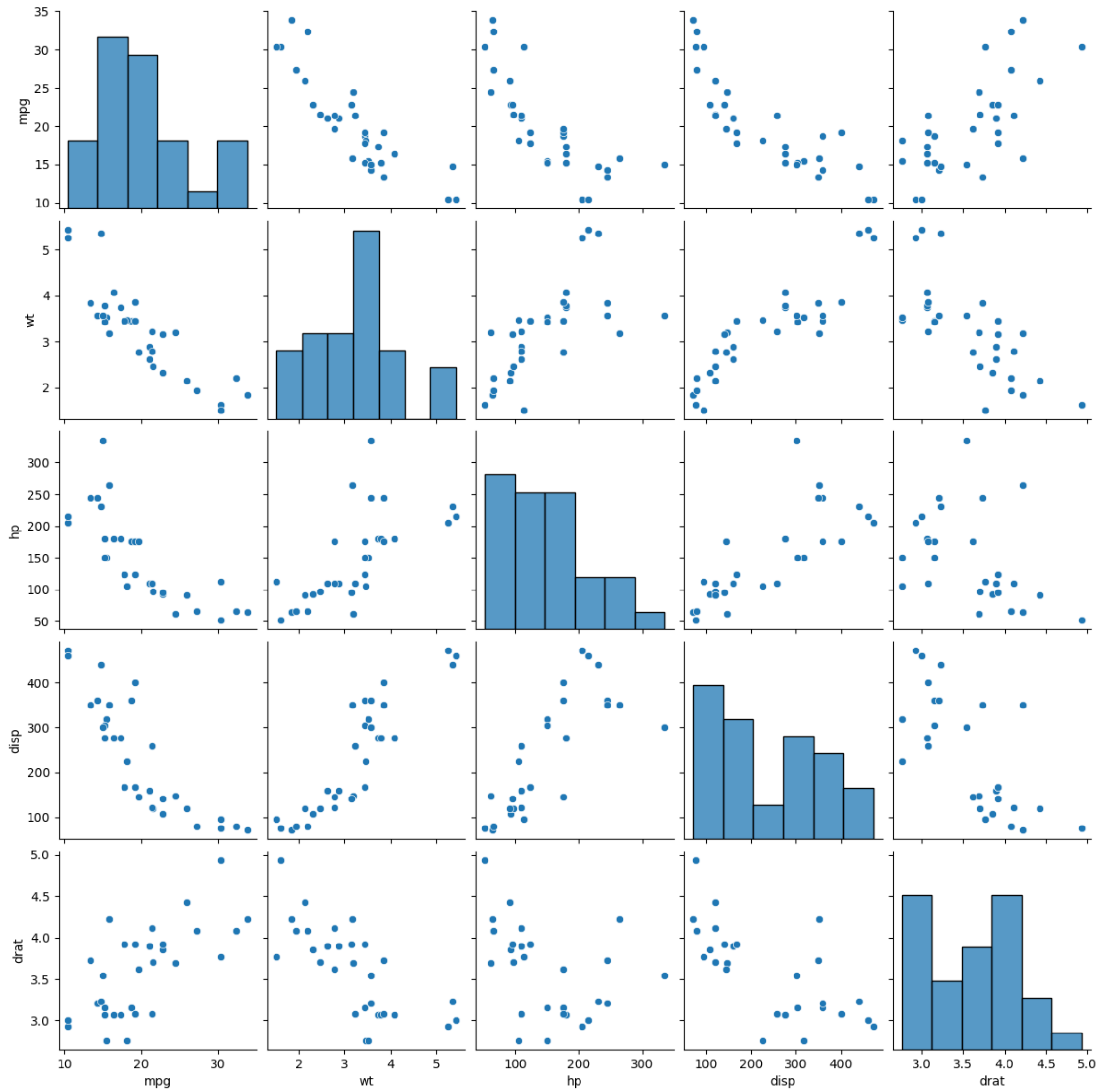
[25]:

```
      mpg    wt    hp  disp  drat
mpg   1.00 -0.87 -0.78 -0.85  0.68
wt   -0.87  1.00  0.66  0.89 -0.71
hp   -0.78  0.66  1.00  0.79 -0.45
disp -0.85  0.89  0.79  1.00 -0.71
drat  0.68 -0.71 -0.45 -0.71  1.00
```

[26]:

```python
## To draw correaltion using Pairplot
import seaborn as sns
sns.pairplot(z)
```

[26]: <seaborn.axisgrid.PairGrid at 0x7c6007e2ad70>



[28]:
```
## OTHER FORMULAS / FORMS OF CORRELATION
# PEARSON PRODUCT MOMENT CORREALTION COEFFICIENT
from scipy import stats
statistic, pvalue= stats.pearsonr(data.mpg,data.wt)
print("Pearson statistic is",statistic.round(2))
print("Pearson p-value is",pvalue.round(2))
print()
```

```
Pearson statistic is -0.87
Pearson p-value is 0.0
```

[29]:
```
# SPEARMAN RANK CORRELATION COEFFICIENT
from scipy import stats
statistic, pvalue= stats.spearmanr(data.mpg,data.wt)
print("Spearman statistic is",statistic.round(2))
print("Spearman p-value is",pvalue.round(2))
```

```
Spearman statistic is -0.89
Spearman p-value is 0.0
```

[31]:
```
## CORRELATION IN UPPER TRIANGULAR FORM
import numpy as np
print(np.triu(z.corr().round(2)))
```

```
[[ 1.   -0.87 -0.78 -0.85  0.68]
 [ 0.    1.    0.66  0.89 -0.71]
 [ 0.    0.    1.    0.79 -0.45]
 [ 0.    0.    0.    1.   -0.71]
 [ 0.    0.    0.    0.    1.  ]]
```

[34]:
```
# REGRESSION ANALYSI
import statsmodels.api as sm
y=data.loc[:,'mpg']
x=data.loc[:,('wt',"disp",'drat','hp')]
x=sm.add_constant(x)
model=sm.OLS(y,x).fit()
print(model.summary())
```

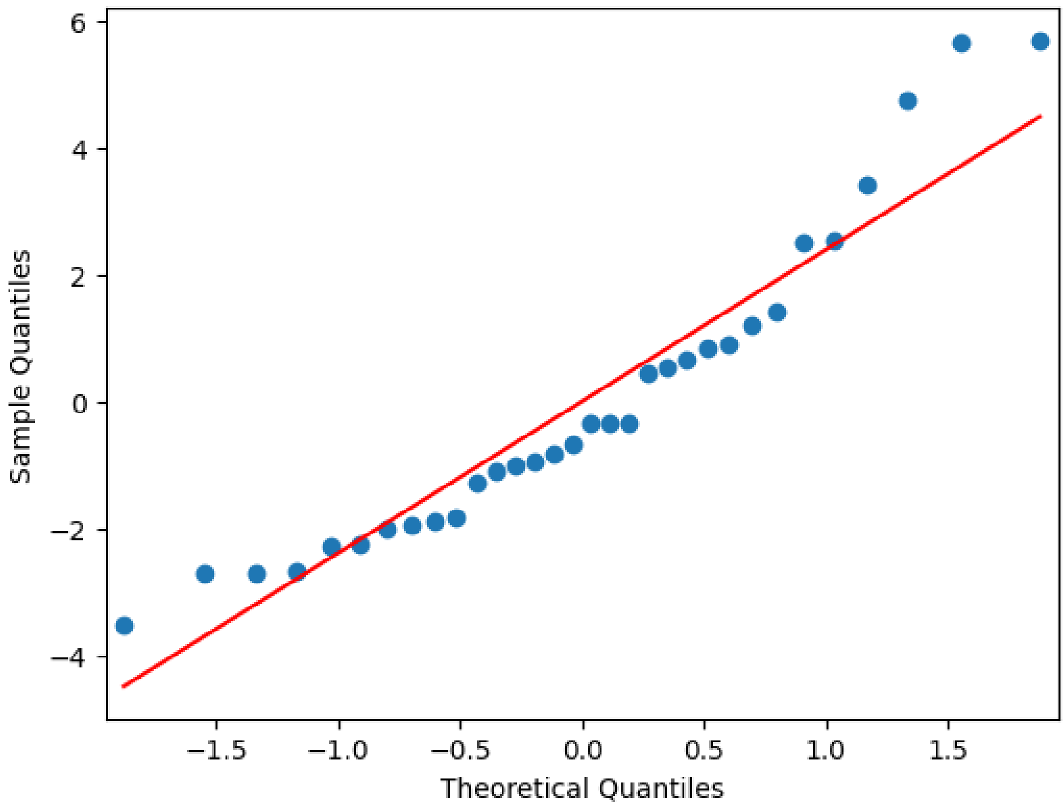
```
                            OLS Regression Results
==============================================================================
Dep. Variable:                    mpg   R-squared:                       0.838
Model:                            OLS   Adj. R-squared:                  0.814
Method:                 Least Squares   F-statistic:                     34.82
Date:                Fri, 19 Jul 2024   Prob (F-statistic):           2.70e-10
Time:                        12:16:10   Log-Likelihood:                -73.292
No. Observations:                  32   AIC:                             156.6
Df Residuals:                      27   BIC:                             163.9
```

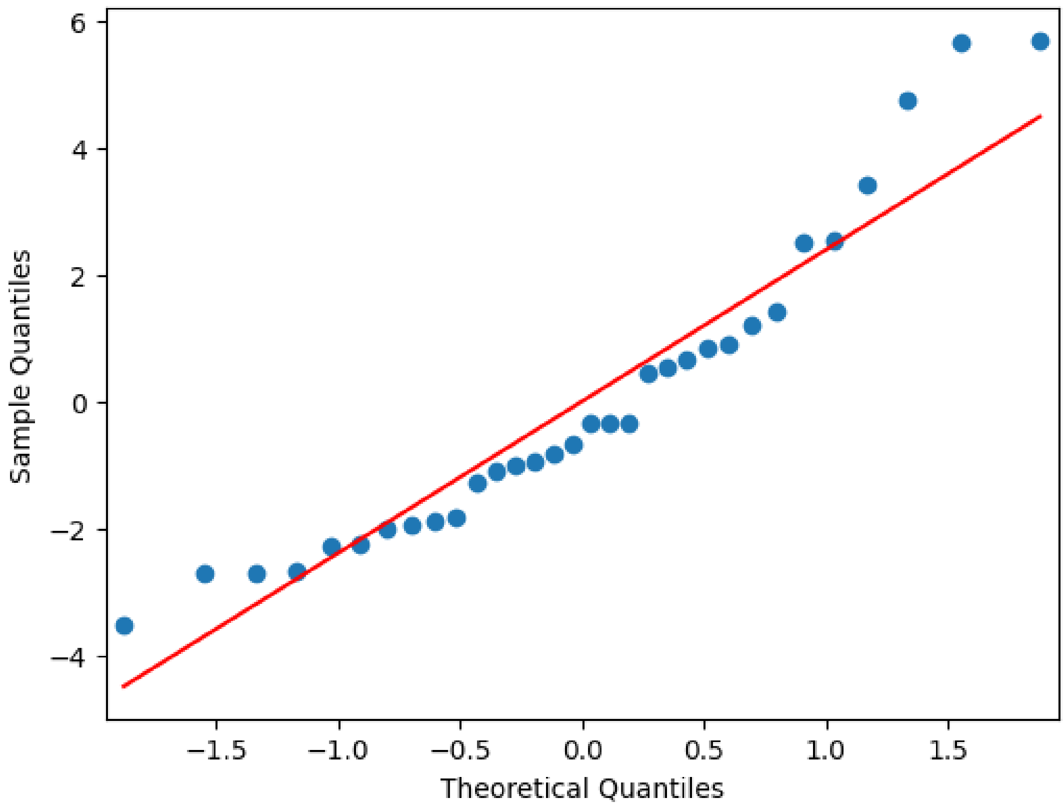
```
Df Model:                           4
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         29.1487      6.294      4.631      0.000      16.235      42.062
wt            -3.4797      1.078     -3.227      0.003      -5.692      -1.267
disp           0.0038      0.011      0.353      0.727      -0.018       0.026
drat           1.7680      1.320      1.340      0.192      -0.940       4.476
hp            -0.0348      0.012     -2.999      0.006      -0.059      -0.011
==============================================================================
Omnibus:                        5.267   Durbin-Watson:                   1.736
Prob(Omnibus):                  0.072   Jarque-Bera (JB):                4.327
Skew:                           0.899   Prob(JB):                        0.115
Kurtosis:                       3.102   Cond. No.                     4.26e+03
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 4.26e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
```

[36]:

```python
# REGRESSION ANALYSIS - QQ Plot
import statsmodels.api as sm
sm.qqplot(model.resid, line='s')
```

[36]:


6
4
2
0
−2
−4
Sample Quantiles
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
Theoretical Quantiles



[37]:
```
## HYPOTHESIS TESTING
# ONE SAMPLE T-TEST
# CHECK WHETHER THE MEAN SOCER OF mpg is 30?

from scipy import stats
stats.ttest_1samp(data.mpg,30)
```

[37]:
```
TtestResult(statistic=-9.300874936052095, pvalue=1.7572027848912617e-10, df=31)
```

[38]:
```
print("Mean of mpg is:", data.mpg.mean().round(2))
print("T-statistic value is:",round(statistic,2))
print("p value is:",round(pvalue,3))
```

```
Mean of mpg is: 20.09
T-statistic value is: -0.89
p value is: 0.0
```

[39]:
```
## TWO INDEPENDENT SAMPLE T-TEST
mpg0=data.loc[data.vs==0,"mpg"]
mpg1=data.loc[data.vs==1,"mpg"]
```

```
statistic, pvalue = stats.ttest_ind(mpg0,mpg1)

print("Mean of mpg0 is:", mpg0.mean().round(2))
print("Mean of mpg1 is:", mpg1.mean().round(2))

print("T-statistic value is:",round(statistic,2))
print("p value is:",round(pvalue,3))
```

```
Mean of mpg0 is: 16.62
Mean of mpg1 is: 24.56
T-statistic value is: -4.86
p value is: 0.0
```

[ ]:
```
## PAIRED T-TEST
```

[40]:
```
statistic, pvalue =stats.ttest_rel(data.wt,data.drat)

print("Mean of wt is:", data.wt.mean().round(2))
print("Mean of drat is:", data.drat.mean().round(2))

print("T-statistic value is:",round(statistic,2))
print("p value is:",round(pvalue,3))
```

```
Mean of wt is: 3.22
Mean of drat is: 3.6
T-statistic value is: -1.52
p value is: 0.138
```

[42]:
```
## ONE WAY ANOVA
from scipy.stats import f_oneway
mpg4=data.loc[data.cyl==4,"mpg"]
mpg6=data.loc[data.cyl==6,"mpg"]
mpg8=data.loc[data.cyl==8,"mpg"]

statistic, pvalue = stats.f_oneway(mpg4,mpg6, mpg8)

print("Mean of mpg4 is:", mpg4.mean().round(2))
print("Mean of mpg6 is:", mpg6.mean().round(2))
print("Mean of mpg8 is:", mpg8.mean().round(2))
print()
```

```
Mean of mpg4 is: 26.66
Mean of mpg6 is: 19.74
Mean of mpg8 is: 15.1
```

[43]:
```
print("F-statistic value is:",round(statistic,2))
print("p value is:",round(pvalue,3))
print()
```

```
F-statistic value is: 39.7
p value is: 0.0
```

[48]:
```
#Post hoc Comparison
from statsmodels.stats.multicomp import pairwise_tukeyhsd # Import the function

tukey=pairwise_tukeyhsd(data.mpg, data.cyl)
print(tukey)
```

```
 Multiple Comparison of Means - Tukey HSD, FWER=0.05
=====================================================
group1 group2 meandiff p-adj   lower    upper  reject
-----------------------------------------------------
     4      6  -6.9208 0.0003 -10.7693 -3.0722   True
     4      8 -11.5636    0.0 -14.7708 -8.3565   True
     6      8  -4.6429 0.0112  -8.3276 -0.9581   True
-----------------------------------------------------
```

[ ]:
```
# PERFORM CHI-SQUARE ANALYSIS BY WRITING A PROMPT
```

[49]:
```
# prompt: provide labels to vs as v-shape engine to 0 and straight engine to 1.␣
 ↪also provide label to am as automatic to 0 and manual to 1, then run the␣
 ↪crosstab. also run the chi-square test.

data['vs_labels'] = data['vs'].replace({0: 'v-shape', 1: 'straight'})
data['am_labels'] = data['am'].replace({0: 'automatic', 1: 'manual'})

# Crosstab
crosstab = pd.crosstab(data['vs_labels'], data['am_labels'])
print(crosstab)
# Chi-square test
from scipy.stats import chi2_contingency
chi2, p, dof, expected = chi2_contingency(crosstab)
print("Chi-square statistic:", chi2)
print("p-value:", p)
```

```
am_labels  automatic  manual
vs_labels
straight           7       7
v-shape           12       6
Chi-square statistic: 0.34753550543024225
p-value: 0.5555115470131495
```

[50]:
```python
# prompt: prompt: run the logistics regression to check the effect of mpg and␣
↪wt  on the outcome variable vs as defined above. also show the␣
↪classification table and confusion table

import statsmodels.api as sm

# Define the dependent and independent variables
y = data['vs']
x = data[['mpg', 'wt']]
x = sm.add_constant(x)

# Fit the logistic regression model
model = sm.Logit(y, x).fit()

# Print the model summary
print(model.summary())

# Predict the probabilities
y_pred_prob = model.predict(x)

# Classify the predictions based on a threshold (e.g., 0.5)
y_pred = (y_pred_prob >= 0.5).astype(int)

# Create a classification table
classification_table = pd.DataFrame({'Actual': y, 'Predicted': y_pred})
print("\nClassification Table:")
print(classification_table)

# Create a confusion matrix
from sklearn.metrics import confusion_matrix
confusion_matrix = confusion_matrix(y, y_pred)
print("\nConfusion Matrix:")
print(confusion_matrix)
```

```
Optimization terminated successfully.
         Current function value: 0.395279
         Iterations 7
                           Logit Regression Results
==============================================================================
Dep. Variable:                     vs   No. Observations:                   32
Model:                          Logit   Df Residuals:                       29
Method:                           MLE   Df Model:                            2
Date:                Fri, 19 Jul 2024   Pseudo R-squ.:                  0.4232
Time:                        13:40:13   Log-Likelihood:                -12.649
converged:                       True   LL-Null:                       -21.930
Covariance Type:            nonrobust   LLR p-value:                 9.317e-05
==============================================================================
```

```
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const        -12.5412      8.466     -1.481      0.139     -29.134       4.052
mpg            0.5241      0.260      2.012      0.044       0.014       1.034
wt             0.5829      1.184      0.492      0.623      -1.739       2.904
==============================================================================

Classification Table:
    Actual  Predicted
0        0          0
1        0          1
2        1          1
3        1          1
4        0          0
5        1          0
6        0          0
7        1          1
8        1          1
9        1          0
10       1          0
11       0          0
12       0          0
13       0          0
14       0          0
15       0          0
16       0          0
17       1          1
18       1          1
19       1          1
20       1          1
21       0          0
22       0          0
23       0          0
24       0          0
25       1          1
26       0          1
27       1          1
28       0          0
29       0          0
30       0          0
31       1          1

Confusion Matrix:
[[16  2]
 [ 3 11]]
```

[ ]:
```
## TEXT ANALYSIS USNG PROMPT
```

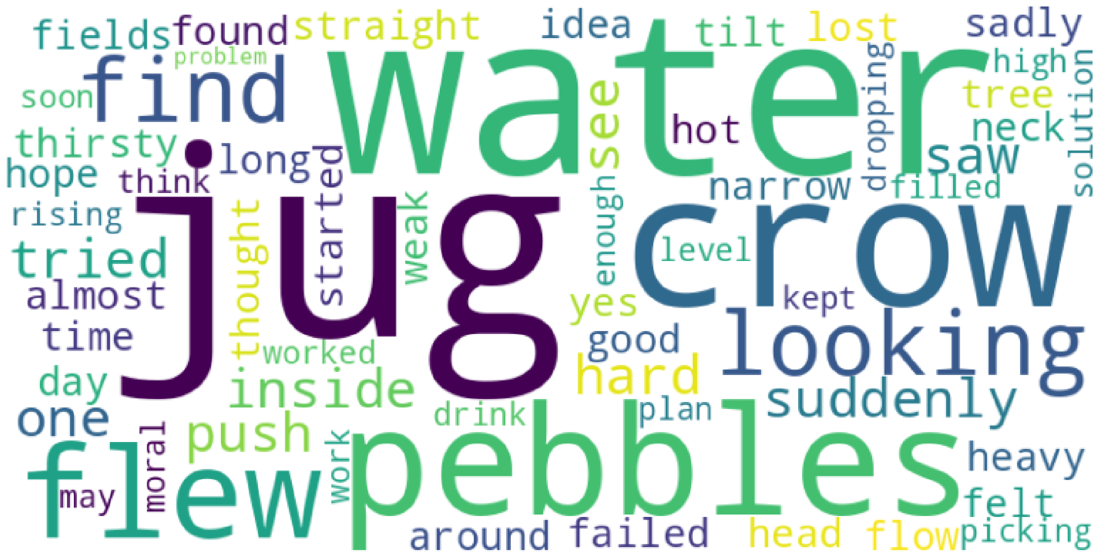
[51]:
```python
# prompt: prompt: import the file crow.txt, perform the text cleaning, remove
↪stopwords and then provide the most frequent words in a table, also make a
↪wordcloud, do attach all necessary packages

!pip install wordcloud
import nltk
nltk.download('punkt')
nltk.download('stopwords')
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from wordcloud import WordCloud
import matplotlib.pyplot as plt
from collections import Counter

# Import the text file
with open('crow.txt', 'r') as file:
  text = file.read()

# Tokenize the text
tokens = word_tokenize(text.lower())

# Remove stop words
stop_words = set(stopwords.words('english'))
filtered_tokens = [word for word in tokens if word.isalnum() and word not in
↪stop_words]

# Count word frequencies
word_counts = Counter(filtered_tokens)

# Create a table of most frequent words
top_words = word_counts.most_common(10)
print("Most Frequent Words:")
for word, count in top_words:
  print(f"{word}: {count}")

# Create a word cloud
wordcloud = WordCloud(width=800, height=400, background_color='white').
↪generate(' '.join(filtered_tokens))

# Display the word cloud
plt.figure(figsize=(10, 5))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.show()
```

```
Requirement already satisfied: wordcloud in /usr/local/lib/python3.10/dist-
packages (1.9.3)
```

```
Requirement already satisfied: numpy>=1.6.1 in /usr/local/lib/python3.10/dist-
packages (from wordcloud) (1.25.2)
Requirement already satisfied: pillow in /usr/local/lib/python3.10/dist-packages
(from wordcloud) (9.4.0)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.10/dist-
packages (from wordcloud) (3.7.1)
Requirement already satisfied: contourpy>=1.0.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib->wordcloud) (1.2.1)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-
packages (from matplotlib->wordcloud) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in
/usr/local/lib/python3.10/dist-packages (from matplotlib->wordcloud) (4.53.1)
Requirement already satisfied: kiwisolver>=1.0.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib->wordcloud) (1.4.5)
Requirement already satisfied: packaging>=20.0 in
/usr/local/lib/python3.10/dist-packages (from matplotlib->wordcloud) (24.1)
Requirement already satisfied: pyparsing>=2.3.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib->wordcloud) (3.1.2)
Requirement already satisfied: python-dateutil>=2.7 in
/usr/local/lib/python3.10/dist-packages (from matplotlib->wordcloud) (2.8.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-
packages (from python-dateutil>=2.7->matplotlib->wordcloud) (1.16.0)
```

```
[nltk_data] Downloading package punkt to /root/nltk_data…
[nltk_data]   Unzipping tokenizers/punkt.zip.
[nltk_data] Downloading package stopwords to /root/nltk_data…
[nltk_data]   Unzipping corpora/stopwords.zip.
```

```
Most Frequent Words:
jug: 8
water: 6
crow: 4
pebbles: 3
flew: 2
looking: 2
could: 2
find: 2
suddenly: 2
saw: 2
```


water
jug
crow
pebbles
flew
find
looking
fields
found
straight
idea
tilt
lost
sadly
problem
high
soon
tree
thirsty
see
hot
neck
hope
think
long
saw
dropping
solution
rising
narrow
filled
tried
thought
started
weak
enough
level
almost
yes
kept
time
good
day
worked
hard
inside
suddenly
one
drink
plan
push
moral
work
heavy
may
felt
around
failed
head
flow
picking