statistical_analysis

September 17, 2022

[ ]:
```python
import statistics
import math
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import scipy
from scipy import stats
from IPython.display import Image
import warnings
warnings.filterwarnings('ignore')
```

0.0.1 Statistical Analysis

I. Descriptive Statistics 1. Measure of central tendency - 1.1 Mean - 1.2 Median - 1.3 Mode 2. Measure of variability - 2.1 Min, max and range - 2.2 Percentiles - 2.3 Variance - 2.4 Standard deviation - 2.5 Distribution - 2.6 Skewness - 2.7 Kurtosis

[ ]:
```python
Image(url=r'/home/achmadadyatma/Documents/learncode/my-data-analyst_project/
↪statistical_analysis/capture/descriptive_statistics.png')
```

[ ]:
```
<IPython.core.display.Image object>
```

[ ]:
```python
#Data for descriptive statistics purposes
#1. Numpy Array
speed = np.array([99,86,87,88,111,86,103,87,94,78,77,85,86], dtype=int)
age = np.array([5,31,43,48,50,41,7,11,15,39,80,82,32,2,8,6,25,36,27,61,31],␣
↪dtype=int)

print('speed array:\n', speed.dtype, speed.view())
print('age array :\n', age.dtype, age.view(),'\n', )

#2. Pandas DataFrame
df_cars = pd.read_csv(r'/home/achmadadyatma/Documents/learncode/
↪my-data-analyst_project/statistical_analysis/cars.csv')

print(df_cars.head(10))
```

```
speed array:
 int64 [ 99  86  87  88 111  86 103  87  94  78  77  85  86]
age array :
 int64 [ 5 31 43 48 50 41  7 11 15 39 80 82 32  2  8  6 25 36 27 61 31]

          Car       Model  Volume  Weight  CO2
0      Toyoty        Aygo    1000     790   99
1  Mitsubishi  Space Star    1200    1160   95
2       Skoda      Citigo    1000     929   95
3        Fiat         500     900     865   90
4        Mini      Cooper    1500    1140  105
5          VW         Up!    1000     929  105
6       Skoda       Fabia    1400    1109   90
7    Mercedes     A-Class    1500    1365   92
8        Ford      Fiesta    1500    1112   98
9        Audi          A1    1600    1150   99
```

[ ]:
```
# I. Descriptive statistics
# 1.1 Measures of central tendency - mean
# The mean is the average number, found by adding all data points and dividing␣
 ↪by the number of data points
mean_1_arr = np.mean(speed)
mean_2_arr = np.mean(age)
mean_3_df = df_cars[['Volume', 'Weight','CO2']].mean(axis=0)

print('mean_1_arr(speed):', mean_1_arr)
print('mean_2_arr(age):',mean_2_arr)
print('mean_3_df(cars):\n', mean_3_df)
```

```
mean_1_arr(speed): 89.76923076923077
mean_2_arr(age): 32.38095238095238
mean_3_df(cars):
 Volume    1611.111111
Weight    1292.277778
CO2        102.027778
dtype: float64
```

[ ]:
```
# I. Descriptive statistics
# 1.2 Measures of central tendency - median
# The median is the middle number in a sorted, ascending or descending list of␣
 ↪numbers and can be more descriptive than the average (mean)
median_1_arr = np.median(speed)
median_2_arr = np.median(age)
median_3_df = df_cars[['Volume', 'Weight','CO2']].median(axis=0)

print('median_1_arr(speed):', median_1_arr)
print('median_2_arr(age):', median_2_arr)
```

```
print('median_3_dff(cars):\n', median_3_df)
```

```
median_1_arr(speed): 87.0
median_2_arr(age): 31.0
median_3_dff(cars):
 Volume    1600.0
Weight    1329.0
CO2         99.0
dtype: float64
```

[ ]:
```
# I. Descriptive statistics
# 1.3 Measures of central tendency - mode
# The mode is the value that is repeatedly occuring in a given dataset
mode_1_arr = stats.mode(speed)
mode_2_arr = stats.mode(age)
mode_3_df = df_cars[['Volume', 'Weight', 'CO2']].mode(axis=0)

print('mode_1_arr(speed):\n', mode_1_arr)
print('mode_2_arr(age):\n', mode_2_arr)
print('mode_3_df(cars):\n', mode_3_df)
```

```
mode_1_arr(speed):
 ModeResult(mode=array([86]), count=array([3]))
mode_2_arr(age):
 ModeResult(mode=array([31]), count=array([2]))
mode_3_df(cars):
    Volume  Weight  CO2
0    1600    1365   99
```

[ ]:
```
# I. Descriptive statistics
# 2.1 Measures of variability - min, max and range
# The min is simply the lowest observation, while the max is the highest␣
 ↪observation
# The range is the difference between the largest and smallest values (range =␣
 ↪max() - min())
min_1_arr = np.amax(speed)
max_1_arr = np.amin(speed)
range_1_arr = np.ptp(speed)

print('min_1_arr(speed) :', min_1_arr)
print('max_1_arr(speed) :', max_1_arr)
print('range_1_arr(speed) :', range_1_arr)

min_2_arr = np.amax(age)
max_2_arr = np.amin(age)
range_2_arr = np.ptp(age)
```

```
print('min_2_arr(age) :', min_2_arr)
print('max_2_arr(age) :', max_2_arr)
print('range_2_arr(age) :', range_2_arr)

min_3_df = df_cars[['Volume', 'Weight','CO2']].min(axis=0)
max_3_df = df_cars[['Volume', 'Weight','CO2']].max(axis=0)
range_3_df = df_cars[['Volume', 'Weight','CO2']].max()- df_cars[['Volume',␣
↪'Weight','CO2']].min()

print('min_3_df(cars) :\n', min_3_df)
print('max_3_df(cars) :\n', max_3_df)
print('range_3_df(cars) :\n', range_3_df)
```

```
min_1_arr(speed) : 111
max_1_arr(speed) : 77
range_1_arr(speed) : 34
min_2_arr(age) : 82
max_2_arr(age) : 2
range_2_arr(age) : 80
min_3_df(cars) :
 Volume    900
Weight    790
CO2        90
dtype: int64
max_3_df(cars) :
 Volume    2500
Weight    1746
CO2        120
dtype: int64
range_3_df(cars) :
 Volume    1600
Weight     956
CO2         30
dtype: int64
```

[ ]:
```
# I. Descriptive statistics
# 2.2 Measures of variability - percentiles
# The percentiles is a measure used in statistics indicating the value below␣
↪which a given percentage of observations in a group of observations fall.␣
↪Example, the 20th percentile is the value (or score) below which 20% of the␣
↪observations may be found
perc_1_arr = np.percentile(speed, q=[25, 50, 75, 90])
perc_2_arr =  np.percentile(age, q=[25, 50, 75, 90])
perc_3_df = df_cars[['Volume', 'Weight','CO2']].quantile(axis=0, q=[0.25, 0.50,␣
↪0.75, 0.90])

print('perc_1_arr (speed) :', perc_1_arr)
```

```
print('perc_2_arr (age) :', perc_2_arr)
print('perc_3_df (cars) :\n', perc_3_df)
```

```
perc_1_arr (speed) : [ 86.   87.   94.  102.2]
perc_2_arr (age) : [11. 31. 43. 61.]
perc_3_df (cars) :
       Volume   Weight     CO2
0.25  1475.0  1117.25   97.75
0.50  1600.0  1329.00   99.00
0.75  2000.0  1418.25  105.00
0.90  2050.0  1594.50  114.00
```

[ ]:
```
# I. Descriptive statistics
# 2.3 Measures of variability - variance
# The variance is the average of the squared differences from the mean value
# A large variance indicates that the data is spread out, meanwhile a small␣
 ↪variance indicates that the data is clustered
var_1_arr = statistics.variance(speed)
var_2_arr = statistics.variance(age)
var_3_df = df_cars[['Volume', 'Weight','CO2']].var(axis=0)

print('var_1_arr (speed) :', var_1_arr)
print('var_2_arr (age) :', var_2_arr)
print('var_3_df (cars) :\n', var_3_df)
```

```
var_1_arr (speed) : 92
var_2_arr (age) : 540
var_3_df (cars) :
 Volume    151301.587302
Weight     58623.977778
CO2           55.570635
dtype: float64
```

[ ]:
```
# I. Descriptive statistics
# 2.4 Measures of variability - standard deviation
# The standard deviation is a number that describes how spread out the values␣
 ↪are
# A low standard deviation means that most of the numbers are close to the mean␣
 ↪(average) value, meanwhile a high standard deviation means that the values
std_1_arr = np.std(speed)
std_2_arr = np.std(age)
std_3_df =  df_cars[['Volume', 'Weight','CO2']].std(axis=0)

print('std_1_arr (speed) :', std_1_arr)
print('std_2_arr (age) :', std_2_arr)
print('std_3_df (cars) :\n', std_3_df)
```

```
std_1_arr (speed) : 9.258292301032677
std_2_arr (age) : 22.678868264524073
std_3_df (cars) :
 Volume    388.975047
Weight    242.123889
CO2         7.454571
dtype: float64
```

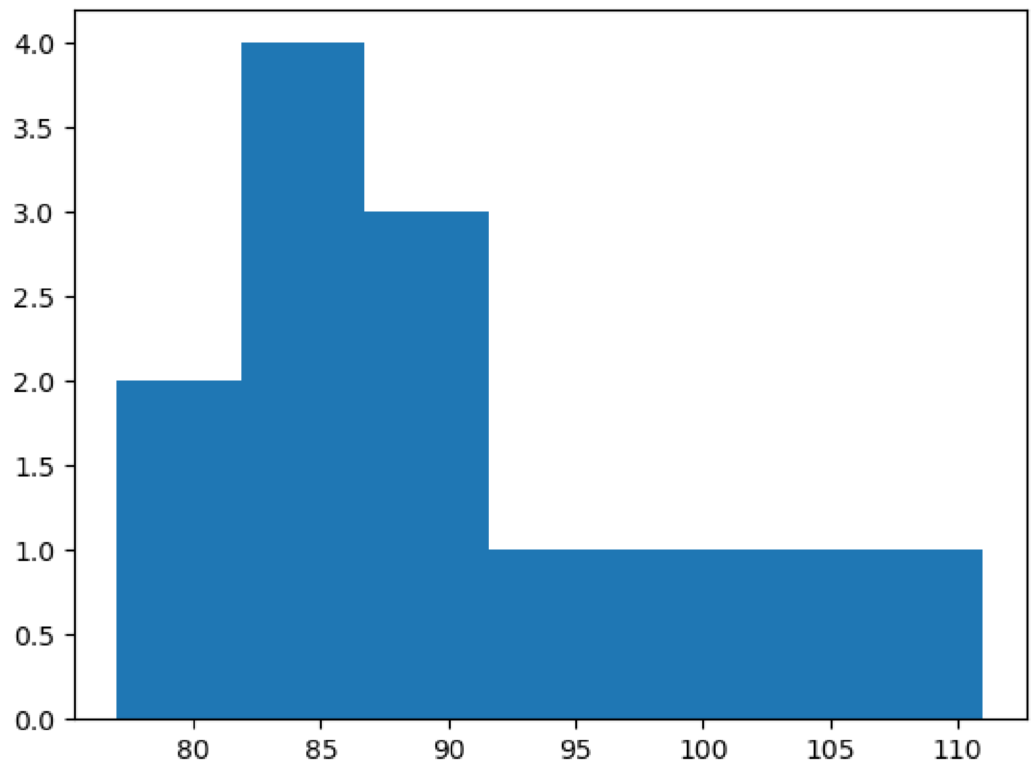
[ ]:
```
# I. Descriptive statistics
# 2.5 Measures of variability - distribution
# The distribution of a statistical dataset is the spread out of the data which
 ↪shows all possible values or intervals of the data and how they occur
# Histogram can be used for determine the distribution of dataset. A histogram
 ↪is a chart that plots the distribution of a numeric variable's values as a
 ↪series of bars
```

[ ]:
```
print('Data Distribution Interpretation')
Image(url=r'/home/achmadadyatma/Documents/learncode/my-data-analyst_project/
 ↪statistical_analysis/capture/histogram-interpretation')
```

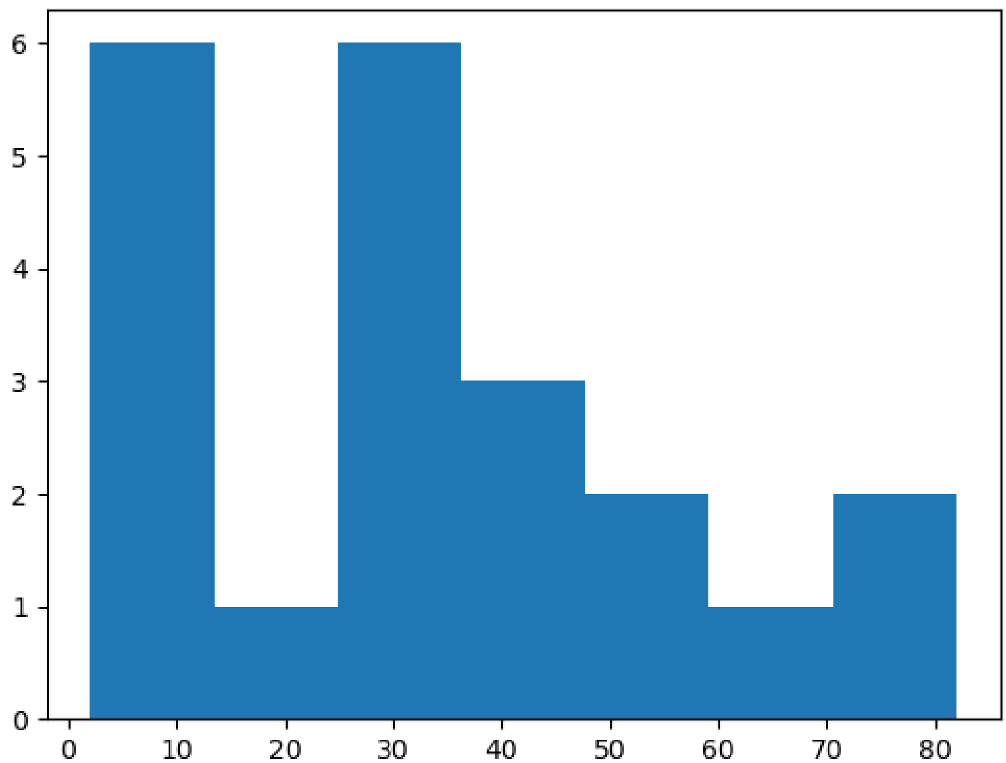
```
Data Distribution Interpretation
```

[ ]: <IPython.core.display.Image object>

[ ]:
```
dis_1_arr = speed
plt.hist(dis_1_arr, bins=7)
plt.show()
```

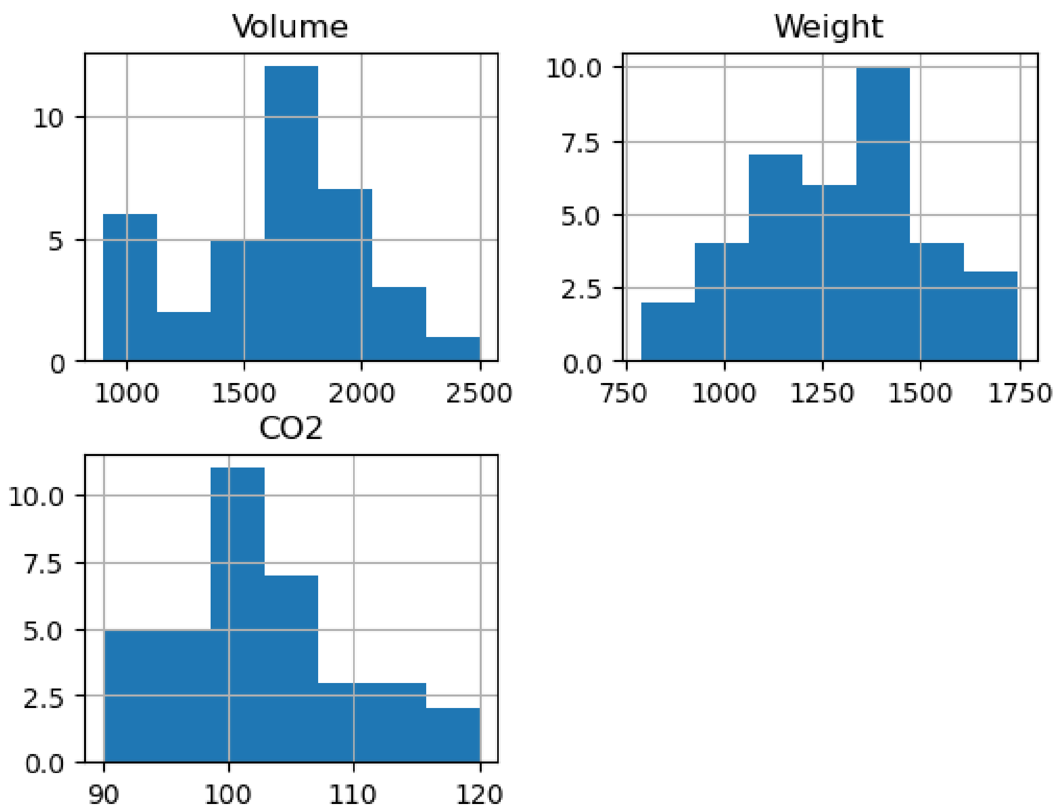


[ ]:
```
dis_2_arr = age
plt.hist(dis_2_arr, bins=7)
plt.show()
```




```
[ ]: df_cars.hist(bins='auto')
```

```
[ ]: array([[<AxesSubplot:title={'center':'Volume'}>,
             <AxesSubplot:title={'center':'Weight'}>],
            [<AxesSubplot:title={'center':'CO2'}>, <AxesSubplot:>]],
           dtype=object)
```



[ ]:
```
# I. Descriptive statistics
# 2.6 Measures of variability - skewness
# The skewness is a measure of the asymmetry of a distribution. A distribution␣
↪can have right (or positive), left (or negative), or zero skewness
# If the skewness is between -0.5 and 0.5, the data are fairly symmetrcal. If␣
↪the skewness is between -1 and -0.5 or between 0.5 and 1, the data are␣
↪moderately skewed. If the skewness is less than -1 or greater than 1, the␣
↪data are highly skewed
```

[ ]:
```
print('Skewness Interpretation')
Image(url=r'/home/achmadadyatma/Documents/learncode/my-data-analyst_project/
↪statistical_analysis/capture/skewness-interpretation.jpeg')
```

```
Skewness Interpretation
```

[ ]: <IPython.core.display.Image object>

[ ]:
```
skew_1_arr = scipy.stats.skew(speed)
skew_2_arr = scipy.stats.skew(age)
skew_3_df =  df_cars[['Volume', 'Weight','CO2']].skew(axis=0)
```

```
print('skew_1_arr (speed) :', skew_1_arr)
print('skew_2_arr (age) :', skew_2_arr)
print('skew_3_df (cars) :\n', skew_3_df)

print('\nInterpretation :')

print('skew_1_arr(speed) : positive moderately skewed')
print('skew_2_arr(age) : positive moderately skewed')
print('skew_3_arr(cars) \nvolume : positive symmetrical \nweight : negative␣
↪symmetrical \nCO2 : positive moderately skewed')
```

```
skew_1_arr (speed) : 0.845105471183182
skew_2_arr (age) : 0.619215361674323
skew_3_df (cars) :
 Volume    0.016157
Weight   -0.116137
CO2       0.656440
dtype: float64

Interpretation :
skew_1_arr(speed) : positive moderately skewed
skew_2_arr(age) : positive moderately skewed
skew_3_arr(cars)
volume : positive symmetrical
weight : negative symmetrical
CO2 : positive moderately skewed
```

[ ]:
```
# I. Descriptive statistics
# 2.7 Measures of variability - kurtosis
# The kurtosis is a measure of the tailedness of a distribution
# Tailedness is how often outliers occur
# If the value is greater than (>0), the distribution is lepto-kurtic. If the␣
↪value is less than (<0) the distribution is platy-kurtic. If the value near␣
↪0 (=0) the distribution is meso-kurtic
```

[ ]:
```
Image(url=r'/home/achmadadyatma/Documents/learncode/my-data-analyst_project/
↪statistical_analysis/capture/kurtosis-interpretation.png')
```

[ ]: <IPython.core.display.Image object>

[ ]:
```
kur_1_arr = scipy.stats.kurtosis(speed)
kur_2_arr = scipy.stats.kurtosis(age)
kur_3_df =  df_cars[['Volume', 'Weight','CO2']].kurtosis(axis=0)

print('kur_1_arr (speed) :', kur_1_arr)
print('kur_2_arr (age) :', kur_2_arr)
print('kur_3_df (cars) :\n', kur_3_df)
```

```
print('\nInterpretation :')

print('kur_1_arr(speed) : Meso-kurtic')
print('kur_2_arr(age) : Platy-kurtic')
print('kur_3_arr(cars) \nvolume : Platy-kurtic \nweight : Platy-kurtic \nCO2 :␣
↪Meso-kurtic')
```

```
kur_1_arr (speed) : 0.01965093187518896
kur_2_arr (age) : -0.2960101885516577
kur_3_df (cars) :
 Volume   -0.399033
Weight   -0.464196
CO2      -0.005159
dtype: float64

Interpretation :
kur_1_arr(speed) : Meso-kurtic
kur_2_arr(age) : Platy-kurtic
kur_3_arr(cars)
volume : Platy-kurtic
weight : Platy-kurtic
CO2 : Meso-kurtic
```

[ ]:
```
# summary to generate descriptive statistics
'''
In NumPy array, describe() method will generate the value of :
1. Number of observation(nobs)
2. Min max
3. Mean
4. Variance
5. Skewness
6. Kurtosis
not included :
1. Median
2. Mode
3. Range
4. Percentiles
5. Standard deviation
6. Distribution
'''
summary_1_arr = stats.describe(speed)
summary_2_arr = stats.describe(age)

print('summary_1_arr (speed) :\n', summary_1_arr)
print('\nsummary_2_arr (age) :\n', summary_2_arr)
```

```
summary_1_arr (speed) :
```

```
 DescribeResult(nobs=13, minmax=(77, 111), mean=89.76923076923077,
variance=92.85897435897435, skewness=0.845105471183182,
kurtosis=0.019650931875188896)

summary_2_arr (age) :
 DescribeResult(nobs=21, minmax=(2, 82), mean=32.38095238095238,
variance=540.047619047619, skewness=0.619215361674323,
kurtosis=-0.2960101885516577)
```

[ ]:
```
'''
In Pandas DataFrame, describe() method will generate the value of :
1. Count
2. Mean
3. Standard deviation
4. Min
5. Max
6. Percentiles (25%, 50%, 75%)
not included :
1. Median
2. Mode
3. Range
4. Variance
5. Distribution
6. Skewness
7. Kurtosis
'''
summary_3_df =  df_cars[['Volume', 'Weight', 'CO2']].describe()

print('summary_3_df (cars) :\n', summary_3_df)
```

```
summary_3_df (cars) :
              Volume       Weight         CO2
count     36.000000    36.000000   36.000000
mean    1611.111111  1292.277778  102.027778
std      388.975047   242.123889    7.454571
min      900.000000   790.000000   90.000000
25%     1475.000000  1117.250000   97.750000
50%     1600.000000  1329.000000   99.000000
75%     2000.000000  1418.250000  105.000000
max     2500.000000  1746.000000  120.000000
```

II. Inferential Statistics

1. Correlation between pairs of data
    - 1.1 Covariance matrix
    - 1.2 Correlation Matrix
    - 1.3 Heatmap
    - 1.4 Correlation coefficient (CC)
        - 1.4.1 Perfect linear relationship (CC = 1)

    - 1.4.2 Perfect negative linear relationship (CC = -1)
    - 1.4.2 No linear relationship (CC = 0)
2. Perform simple linear regression analysis
    - 2.1 Data preparation (extract and rename data from existing Pandas DataFrame)
    - 2.2 Plot the given data points
    - 2.3 Add needed columns (x-x̄), (y-ȳ), (x-x̄)*(y-ȳ), (x-x̄)^2, (y-ȳ)^2
    - 2.4 Sum up (x-x̄)*(y-ȳ), (x-x̄)^2, (y-ȳ)^2 columns
    - 2.5 Calculate Pearson's correlation coefficient (r)
    - 2.6 Calculate standard deviation of X and Y variables
    - 2.7 Calculate slope (b)
    - 2.8 Calculate intercept (a)
    - 2.9 Plot the given data points and fit the regression line
    - 2.10 Predict value

[ ]:
```
Image(url=r'/home/achmadadyatma/Documents/learncode/my-data-analyst_project/
↪statistical_analysis/capture/inferential_statistics.png')
```

[ ]:
```
<IPython.core.display.Image object>
```

[ ]:
```
#Data for inferential statistics purposes
#Pandas DataFrame
df_health_1 = pd.read_csv(r'/home/achmadadyatma/Documents/learncode/
↪my-data-analyst_project/statistical_analysis/health.csv')

df_health_1.head(250)
```

[ ]:
```
     Duration  Average_Pulse  Max_Pulse  Calorie_Burnage  Hours_Work  \
0          60            110        130              409         0.0
1          60            117        145              479         0.0
2          60            103        135              340         8.0
3          45            109        175              282         8.0
4          45            117        148              406         0.0
..        …               …         …                …          …
158        60            105        140              290         7.0
159        60            110        145              300         7.0
160        60            115        145              310         8.0
161        75            120        150              320         0.0
162        75            125        150              330         8.0

     Hours_Sleep
0            8.0
1            8.0
2            7.5
3            8.0
4            6.5
..           …
158          8.0
159          8.0
```

```
160          8.0
161          8.0
162          8.0

[163 rows x 6 columns]
```

[ ]:
```
# only for negative linear relationship analysis
data_health_2 = {'Hours_Work':[10,9,8,7,6,5,4,3,2,1],
                'Calorie_Burnage': [220,240,260,280,300,320,340,360,380,400]}
df_health_2 = pd.DataFrame(data_health_2)
df_health_2.head()
```

[ ]:
```
   Hours_Work  Calorie_Burnage
0          10              220
1           9              240
2           8              260
3           7              280
4           6              300
```

[ ]:
```
# II. Inferential statistics
# 1.1 Correlation between pairs of data - covariance matrix
# The covariance is a statistical tool that is used to determine the␣
↪relationship between the movements of two random variables.
# Covariance gives you a positive number if the variables are positively related.
↪ You'll get a negative number if they are negatively related. A high␣
↪covariance basically indicates there is a strong relationship between the␣
↪variables.
```

[ ]:
```
#Covariance matrix interpretation
Image(url=r'/home/achmadadyatma/Documents/learncode/my-data-analyst_project/
↪statistical_analysis/capture/covariance-interpretation.jpg')
```

[ ]:
```
<IPython.core.display.Image object>
```

[ ]:
```
df_health_1.cov()
```

[ ]:
```
                    Duration  Average_Pulse   Max_Pulse  Calorie_Burnage  \
Duration         1848.528743    -103.969931    0.631107     10470.396122
Average_Pulse    -103.969931     213.892449  188.464288        70.491138
Max_Pulse           0.631107     188.464288  269.090131       881.014694
Calorie_Burnage  10470.396122     70.491138  881.014694     75200.505643
Hours_Work         -19.405059    -15.809305  -17.162349      -150.572162
Hours_Sleep          2.124801      0.331137    1.029652        14.914451

                 Hours_Work  Hours_Sleep
Duration         -19.405059     2.124801
Average_Pulse    -15.809305     0.331137
```

```
Max_Pulse         -17.162349     1.029652
Calorie_Burnage -150.572162     14.914451
Hours_Work         15.395990    -0.375937
Hours_Sleep        -0.375937     0.440809
```

[ ]:
```
# II. Inferential statistics
# 1.2 Correlation between pairs of data - correlation matrix
# The correlation matrix is simply a table which displays the correlation␣
↪coefficients for different variables.
# The correlation coefficient values can fall between -1 and +1.
# Weak positive correlation would be in the range 0.1 to 0.3, moderate positive␣
↪correlation from 0.3 to 0.5, and strong positive correlation from 0.5 to 1.0
# Weak negative correlation would be in the range -0.1 to -0.3, moderate␣
↪negative correlation from -0.3 to -0.5, and strong negative correlation from␣
↪-0.5 to -1.0
```

[ ]:
```
#Correlation matrix interpretation
Image(url=r'/home/achmadadyatma/Documents/learncode/my-data-analyst_project/
↪statistical_analysis/capture/correlation-coefficient-interpretation.png')
```

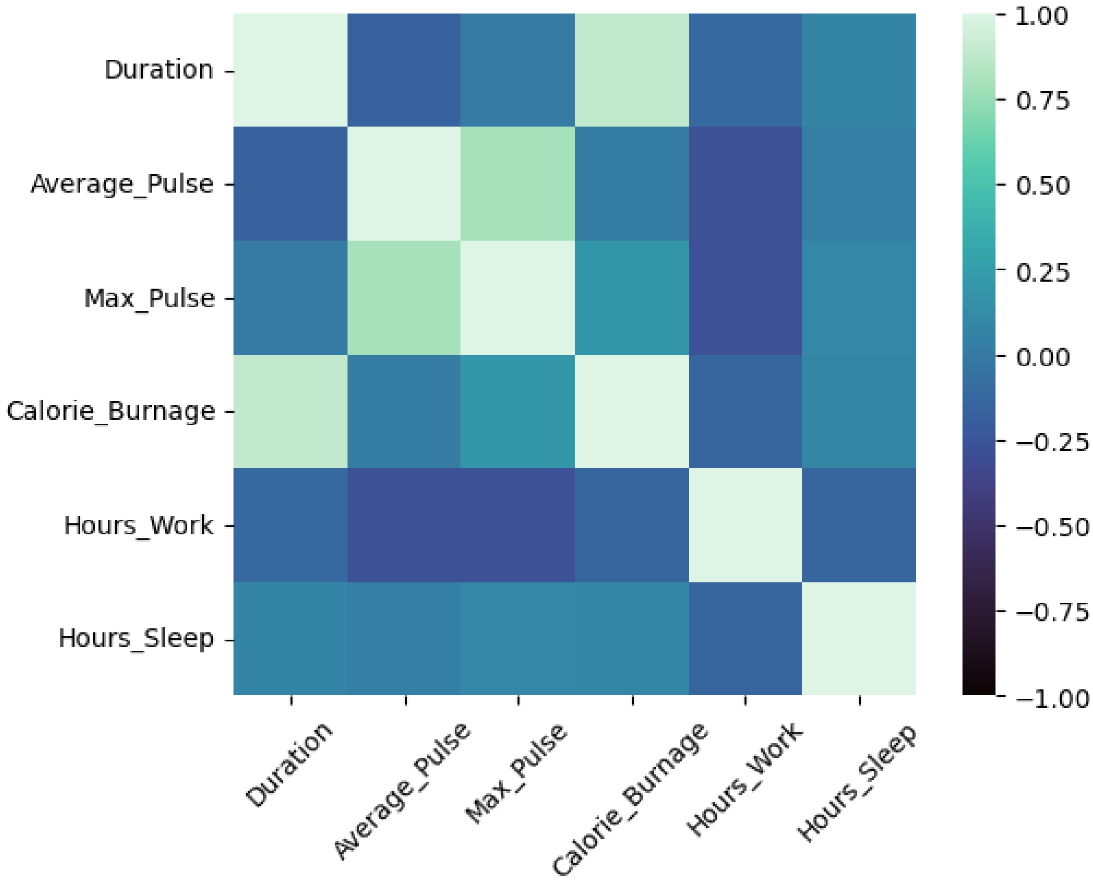
[ ]:
```
<IPython.core.display.Image object>
```

[ ]:
```
df_health_1.corr()
```

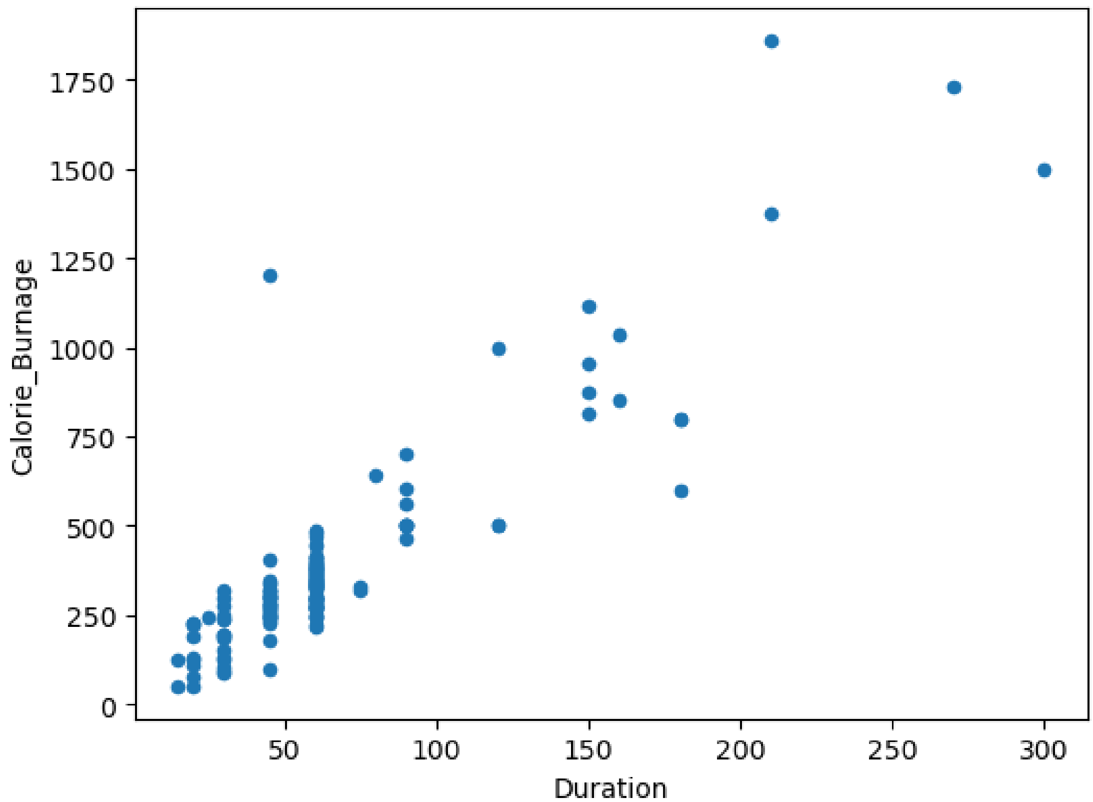
[ ]:
```
                 Duration  Average_Pulse  Max_Pulse  Calorie_Burnage  \
Duration         1.000000      -0.165347   0.000895         0.888055
Average_Pulse   -0.165347       1.000000   0.785566         0.017576
Max_Pulse        0.000895       0.785566   1.000000         0.195850
Calorie_Burnage  0.888055       0.017576   0.195850         1.000000
Hours_Work      -0.115027      -0.275493  -0.266639        -0.139936
Hours_Sleep      0.074435       0.034102   0.094540         0.081917

                 Hours_Work  Hours_Sleep
Duration          -0.115027     0.074435
Average_Pulse     -0.275493     0.034102
Max_Pulse         -0.266639     0.094540
Calorie_Burnage   -0.139936     0.081917
Hours_Work         1.000000    -0.144307
Hours_Sleep       -0.144307     1.000000
```

[ ]:
```
# II. Inferential statistics
# 1.3 Heatmap helps you visualize density. It is a two - dimensional␣
↪representation of data in which values are represented by colors. It can be␣
↪used to visualize correlation matrix of dataset

corr_df_health = df_health_1.corr().round(2)
```

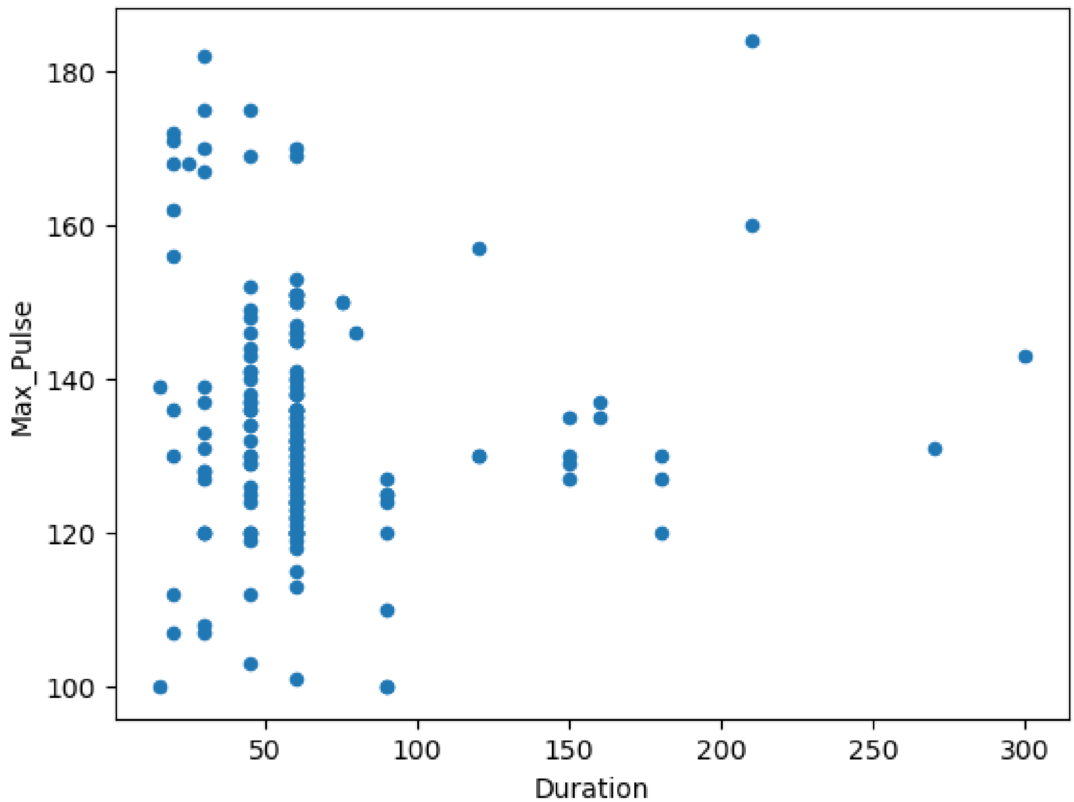
```
axis_corr = sns.heatmap(
    corr_df_health,
    vmin=-1,vmax=1, center=0,
    cmap="mako",
    square=True
)

plt.xticks(rotation=45)
plt.show()
```



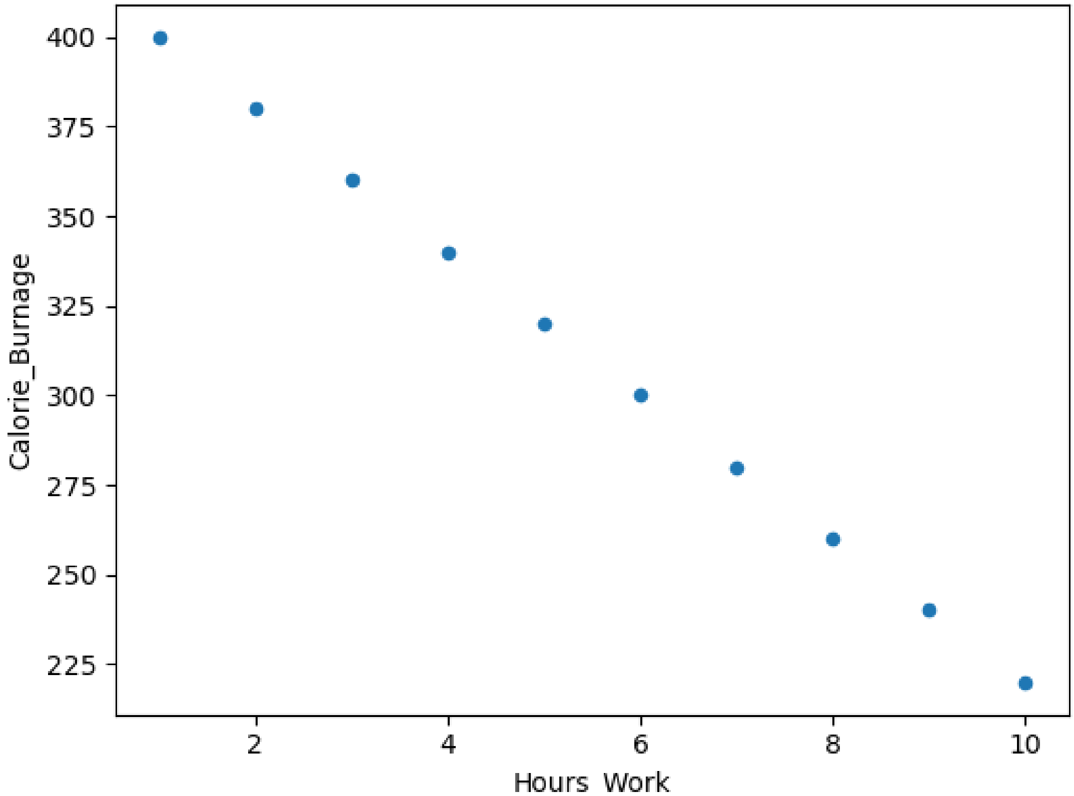
[ ]:
```
# # II. Inferential statistics
# 1.4 Correlation between pairs of data - correlation coefficient (CC)
# Correlation coefficient measures the relationship between two variables. The␣
 ↪correlation coefficient can never be less than -1 or higher than 1
# 1.4.1 Perfect linear relationship (CC = 1)
df_health_1.plot(x ='Duration', y='Calorie_Burnage', kind='scatter')
plt.show()
```



[ ]:
```
# 1.4.2 No linear relationship (CC = 0)
df_health_1.plot(x = 'Duration', y='Max_Pulse', kind='scatter')
plt.show()
```



[ ]:
```
# 1.4.3 Perfect negative linear relationship (CC = -1)
df_health_2.plot(x = 'Hours_Work', y='Calorie_Burnage', kind='scatter')
plt.show()
```




```
[ ]: # II. Inferential statistics
     # 2.1 Simple linear regression
     # It's a form of mathematical regression analysis used to determnine the line␣
      ↪of best fit for a set of data, providing a visual demonstration of the␣
      ↪relationship between the data points. Each point of data represents the␣
      ↪relationship between the data points. Each point of data represents the␣
      ↪relationship between a known independent variable and an unknown dependent␣
      ↪variable
     df_health_1.head()
```

```
[ ]:    Duration  Average_Pulse  Max_Pulse  Calorie_Burnage  Hours_Work  \
     0        60            110        130              409         0.0
     1        60            117        145              479         0.0
     2        60            103        135              340         8.0
     3        45            109        175              282         8.0
     4        45            117        148              406         0.0

        Hours_Sleep
     0          8.0
     1          8.0
     2          7.5
```

```
3           8.0
4           6.5
```

Perform Simple Linear Regression analysis 1. Data preparation (extract and rename data from existing Pandas DataFrame) 2. Plot the given data points 3. Add needed columns (x-x̄), (y-ȳ), (x-x̄)*(y-ȳ), (x-x̄)^2, (y-ȳ)^2 4. Sum up (x-x̄)*(y-ȳ), (x-x̄)^2, (y-ȳ)^2 columns 5. Calculate Pearson's correlation coefficient (r) 6. Calculate standard deviation of X and Y variables 7. Calculate slope (b) 8. Calculate intercept (a) 9. Plot the given data points and fit the regression line 10. Predict value

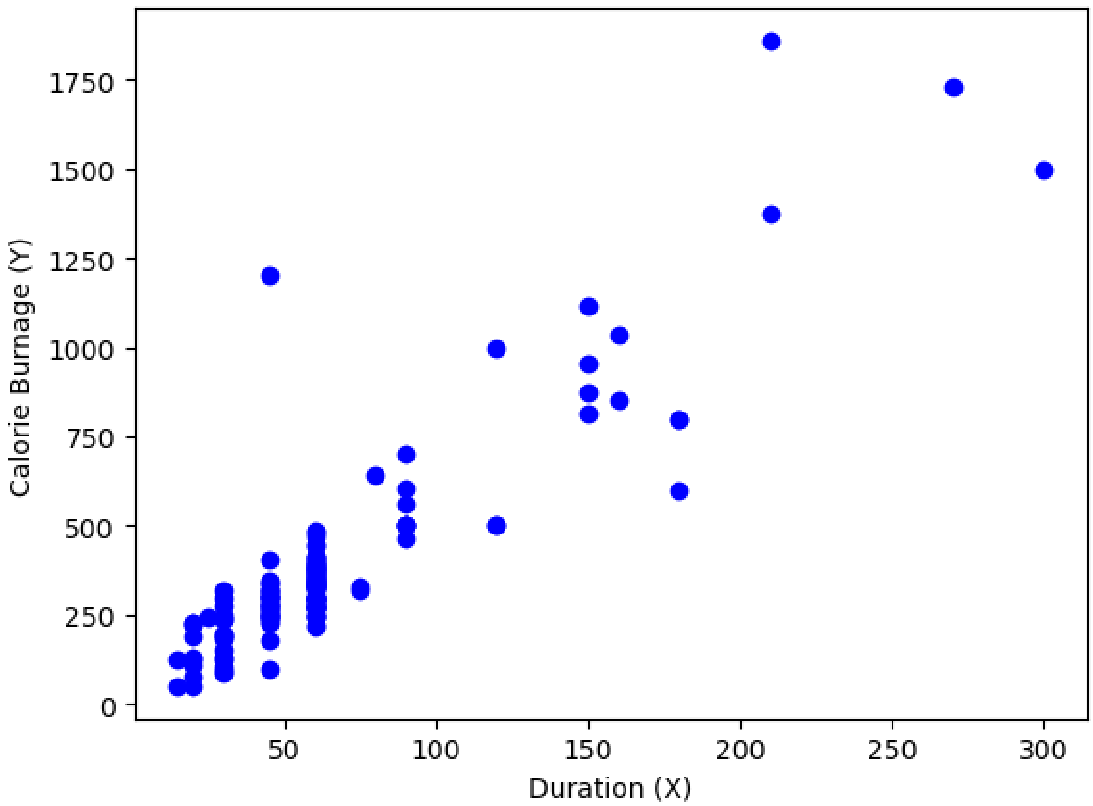
[ ]:
```
# We will analyze the correlation between 'Duration' and 'Calorie_Burnage'␣
↪using least square method
# Duration as independent variable (cause) meanwhile Calorie_Burnage as␣
↪dependent variable (effect)
# Independent variable belongs on the x-axis (horizontal line) of the graph and␣
↪the dependent variable belongs on the y-axis (vertical line)

# extract data from Pandas DataFrame
df_sl_reg = df_health_1[['Duration', 'Calorie_Burnage']]

# rename existing names
df_sl_reg.rename(columns = {'Duration':'Duration (x)', 'Calorie_Burnage':
↪'Calorie_burnage (y)'}, inplace = True)

df_sl_reg.head()
```

[ ]:

	Duration (x)	Calorie_burnage (y)
0	60	409
1	60	479
2	60	340
3	45	282
4	45	406

[ ]:
```
# Plot the given data points
x = df_sl_reg['Duration (x)']
y = df_sl_reg['Calorie_burnage (y)']

plt.scatter(x, y, color='blue')
plt.xlabel('Duration (X)')
plt.ylabel('Calorie Burnage (Y)')
```

[ ]: Text(0, 0.5, 'Calorie Burnage (Y)')



[ ]:

```
# add needed columns (x-x̄), (y-ȳ), (x-x̄)*(y-ȳ), (x-x̄)^2, (y-ȳ)^2
# (x-x̄)
df_sl_reg['(x-x̄)'] = df_sl_reg['Duration (x)'] - df_sl_reg['Duration (x)'].
 ↪mean(axis=0)

# (y-ȳ)
df_sl_reg['(y-ȳ)'] = df_sl_reg['Calorie_burnage (y)'] -␣
 ↪df_sl_reg['Calorie_burnage (y)'].mean(axis=0)

# (x-x̄)*(y-ȳ)
df_sl_reg['(x-x̄)*(y-ȳ)'] = df_sl_reg['(x-x̄)'] * df_sl_reg['(y-ȳ)']

# (x-x̄)^2
df_sl_reg['(x-x̄)^2'] = df_sl_reg['(x-x̄)']**2

# (y-ȳ)^2
df_sl_reg['(y-ȳ)^2'] = df_sl_reg['(y-ȳ)']**2

# sum up (x-x̄)*(y-ȳ), (x-x̄)^2, (y-ȳ)^2 columns
print('Σ(x-x̄)*(y-ȳ):',df_sl_reg['(x-x̄)*(y-ȳ)'].sum(axis=0))
print('Σ(x-x̄)^2:',df_sl_reg['(x-x̄)^2'].sum(axis=0))
```

```
print('Σ(y-ȳ)^2:',df_sl_reg['(y-ȳ)^2'].sum(axis=0))
print('count:',df_sl_reg[['Duration (x)']].count(axis=0))

df_sl_reg.head()
```

```
Σ(x-x̄)*(y-ȳ): 1696204.171779141
Σ(x-x̄)^2: 299461.65644171776
Σ(y-ȳ)^2: 12182481.914110431
count: Duration (x)    163
dtype: int64
```

[ ]:

```
   Duration (x)  Calorie_burnage (y)      (x-x̄)       (y-ȳ)  (x-x̄)*(y-ȳ)  \
0            60                  409  -4.263804   26.631902    -113.553201
1            60                  479  -4.263804   96.631902    -412.019459
2            60                  340  -4.263804  -42.368098     180.649253
3            45                  282 -19.263804 -100.368098    1933.471339
4            45                  406 -19.263804   23.631902    -455.240318

      (x-x̄)^2      (y-ȳ)^2
0   18.180022    709.258196
1   18.180022   9337.724453
2   18.180022   1795.055742
3  371.094132  10073.755128
4  371.094132    558.466785
```

[ ]:

```
print('Pearson\'s Correlation Coefficient formula :')
Image(url=r'/home/achmadadyatma/Documents/learncode/my-data-analyst_project/
 ↪statistical_analysis/capture/Pearsons-correlation-coefficient_formula.png')
```

```
Pearson's Correlation Coefficient formula :
```

[ ]: <IPython.core.display.Image object>

[ ]:

```
# Calculate Pearson's correlation coefficient (r)

r = 1696204.171779141 / math.sqrt(299461.65644171776 * 12182481.914110431)

print(r)
```

```
0.8880545180090471
```

[ ]:

```
Image(url=r'/home/achmadadyatma/Documents/learncode/my-data-analyst_project/
 ↪statistical_analysis/capture/standard-deviation_formula.png')
```

[ ]: <IPython.core.display.Image object>

[ ]:
```
# Calculate standard deviation of X and Y variables

Sx = math.sqrt(299461.65644171776 / (163-1))
Sy = math.sqrt(12182481.914110431 / (163-1))

print(Sx)
print(Sy)
```

```
42.99451992367624
274.22710595901526
```

[ ]:
```
Image(url=r'/home/achmadadyatma/Documents/learncode/my-data-analyst_project/
↪statistical_analysis/capture/slope_formula.png')
```

[ ]: <IPython.core.display.Image object>

[ ]:
```
# Calculate slope (b)
b = r * (Sy / Sx)

print(b)
```

```
5.66417815200078
```

[ ]:
```
Image(url=r'/home/achmadadyatma/Documents/learncode/my-data-analyst_project/
↪statistical_analysis/capture/intercept_formula.png')
```

[ ]: <IPython.core.display.Image object>

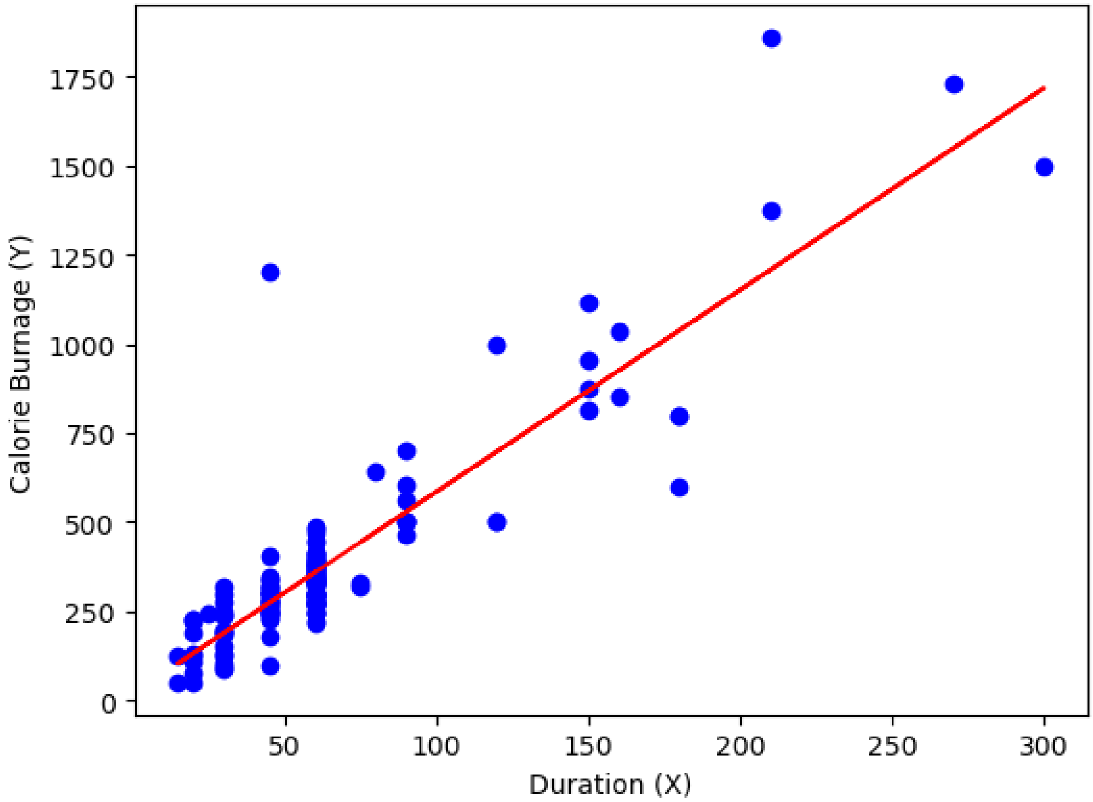
[ ]:
```
# Calculate intercept (a)
a = df_sl_reg['Calorie_burnage (y)'].mean(axis=0) - (b * df_sl_reg['Duration␣
↪(x)'].mean(axis=0))

print(a)
print('result of linear regression formula:\n y = 18.3 + 5.66x')
```

```
18.36646538522598
result of linear regression formula:
 y = 18.3 + 5.66x
```

[ ]:
```
# Plot the given data points and fit the regression line
x = df_sl_reg['Duration (x)']
y = df_sl_reg['Calorie_burnage (y)']
y_pred = b * x + a

plt.scatter(x, y, color='blue')
plt.plot(x, y_pred, color='red')
plt.xlabel('Duration (X)')
```

```
plt.ylabel('Calorie Burnage (Y)')
```

[ ]: Text(0, 0.5, 'Calorie Burnage (Y)')



[ ]:
```
# Predict value using y = 18.3 + 5.66x formula
# we will predict calorie burnage when certain condition of duration

val1 = 200
val2 = 250
val3 = 50

print('Predicted calorie burnage value when duration 200 :', 18.3 + (5.
↪66*(val1)))
print('Predicted calorie burnage value when duration 250 :', 18.3 + (5.
↪66*(val2)))
print('Predicted calorie burnage value when duration 50 :', 18.3 + (5.
↪66*(val3)))
```

```
Predicted calorie burnage value when duration 200 : 1150.3
Predicted calorie burnage value when duration 250 : 1433.3
Predicted calorie burnage value when duration 50 : 301.3
```