LAB SESSION 6

```
import pandas as pd
from sklearn.datasets import load_diabetes

# Load the dataset
diabetes = load_diabetes()
df = pd.DataFrame(data=diabetes.data, columns=diabetes.feature_names)
df['target'] = diabetes.target

# Display the first few rows
print(df.head())
```

```
        age       sex       bmi        bp        s1        s2
s3  \
0  0.038076  0.050680  0.061696  0.021872 -0.044223 -0.034821 -
0.043401
1 -0.001882 -0.044642 -0.051474 -0.026328 -0.008449 -0.019163
0.074412
2  0.085299  0.050680  0.044451 -0.005670 -0.045599 -0.034194 -
0.032356
3 -0.089063 -0.044642 -0.011595 -0.036656  0.012191  0.024991 -
0.036038
4  0.005383 -0.044642 -0.036385  0.021872  0.003935  0.015596
0.008142

         s4        s5        s6  target
0 -0.002592  0.019907 -0.017646   151.0
1 -0.039493 -0.068332 -0.092204    75.0
2 -0.002592  0.002861 -0.025930   141.0
3  0.034309  0.022688 -0.009362   206.0
4 -0.002592 -0.031988 -0.046641   135.0
```

```
# Calculate basic descriptive statistics
print("Mean:\n", df.mean())
print("\nMedian:\n", df.median())
print("\nMode:\n", df.mode().iloc[0])
print("\nStandard Deviation:\n", df.std())
print("\nVariance:\n", df.var())

# Additional descriptive statistics
print("\nRange:\n", df.max() - df.min())
print("\nSkewness:\n", df.skew())
print("\nKurtosis:\n", df.kurt())
```

```
Mean:
 age      -1.444295e-18
sex       2.543215e-18
```

```
bmi      -2.255925e-16
bp       -4.854086e-17
s1       -1.428596e-17
s2        3.898811e-17
s3       -6.028360e-18
s4       -1.788100e-17
s5        9.243486e-17
s6        1.351770e-17
target    1.521335e+02
dtype: float64

Median:
 age         0.005383
sex        -0.044642
bmi        -0.007284
bp         -0.005670
s1         -0.004321
s2         -0.003819
s3         -0.006584
s4         -0.002592
s5         -0.001947
s6         -0.001078
target    140.500000
dtype: float64

Mode:
 age         0.016281
sex        -0.044642
bmi        -0.030996
bp         -0.040099
s1         -0.037344
s2         -0.001001
s3         -0.013948
s4         -0.039493
s5         -0.018114
s6         0.003064
target    72.000000
Name: 0, dtype: float64

Standard Deviation:
 age         0.047619
sex        0.047619
bmi        0.047619
bp         0.047619
s1         0.047619
s2         0.047619
s3         0.047619
s4         0.047619
s5         0.047619
s6         0.047619
```

```
target    77.093005
dtype: float64

Variance:
 age          0.002268
sex          0.002268
bmi          0.002268
bp           0.002268
s1           0.002268
s2           0.002268
s3           0.002268
s4           0.002268
s5           0.002268
s6           0.002268
target    5943.331348
dtype: float64

Range:
 age          0.217952
sex          0.095322
bmi          0.260831
bp           0.244442
s1           0.280694
s2           0.314401
s3           0.283486
s4           0.261629
s5           0.259694
s6           0.273379
target    321.000000
dtype: float64

Skewness:
 age       -0.231382
sex        0.127385
bmi        0.598148
bp         0.290658
s1         0.378108
s2         0.436592
s3         0.799255
s4         0.735374
s5         0.291754
s6         0.207917
target     0.440563
dtype: float64

Kurtosis:
 age       -0.671224
sex       -1.992811
bmi        0.095094
bp        -0.532797
```

```
s1        0.232948
s2        0.601381
s3        0.981507
s4        0.444402
s5       -0.134367
s6        0.236917
target   -0.883057
dtype: float64
```

```
from scipy import stats

# Example data: BMI values
bmi_values = df['bmi']

# Hypothetical population mean for BMI
population_mean = 0.05

# Perform one-sample t-test
t_stat, p_value = stats.ttest_1samp(bmi_values, population_mean)

print(f"T-Statistic: {t_stat}")
print(f"P-Value: {p_value}")
```

```
T-Statistic: -22.074985843710174
P-Value: 2.7634312235044638e-73
```

```
import numpy as np
from scipy import stats

# Sample mean and standard error for BMI
sample_mean = np.mean(bmi_values)
standard_error = stats.sem(bmi_values)

# Compute 95% confidence interval for BMI
confidence_interval = stats.norm.interval(0.95, loc=sample_mean,
scale=standard_error)

print(f"95% Confidence Interval for BMI: {confidence_interval}")
```

```
95% Confidence Interval for BMI: (-0.004439332370169141,
0.0044393323701686915)
```

```
import statsmodels.api as sm

# Define independent variable (add constant for intercept)
X = sm.add_constant(df['bmi'])

# Define dependent variable
y = df['target']

# Fit linear regression model
```

```
model = sm.OLS(y, X).fit()

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

```
                            OLS Regression Results

==============================================================================
Dep. Variable:                 target   R-squared:
0.344
Model:                            OLS   Adj. R-squared:
0.342
Method:                 Least Squares   F-statistic:
230.7
Date:                Thu, 05 Sep 2024   Prob (F-statistic):
3.47e-42
Time:                        20:06:02   Log-Likelihood:
-2454.0
No. Observations:                 442   AIC:
4912.
Df Residuals:                     440   BIC:
4920.
Df Model:                           1

Covariance Type:            nonrobust

==============================================================================
                 coef    std err          t      P>|t|      [0.025
0.975]
------------------------------------------------------------------------------
const        152.1335      2.974     51.162      0.000     146.289
157.978
bmi          949.4353     62.515     15.187      0.000     826.570
1072.301
==============================================================================
Omnibus:                       11.674   Durbin-Watson:
1.848
Prob(Omnibus):                  0.003   Jarque-Bera (JB):
7.310
Skew:                           0.156   Prob(JB):
0.0259
Kurtosis:                       2.453   Cond. No.
21.0
==============================================================================
```

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

EXERCISE 5.1

```
import pandas as pd

# Load dataset from a CSV file (assuming it's already downloaded)
url =
"https://archive.ics.uci.edu/ml/machine-learning-databases/heart-
disease/processed.cleveland.data"
columns = ['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg',
'thalach', 'exang', 'oldpeak', 'slope', 'ca', 'thal', 'target']

# Load the dataset into a DataFrame
df = pd.read_csv(url, names=columns, na_values="?")

# View the first few rows of the dataset
df.head()
```

```
    age  sex   cp  trestbps   chol  fbs  restecg  thalach  exang
oldpeak  \
0  63.0  1.0  1.0     145.0  233.0  1.0      2.0    150.0    0.0
2.3
1  67.0  1.0  4.0     160.0  286.0  0.0      2.0    108.0    1.0
1.5
2  67.0  1.0  4.0     120.0  229.0  0.0      2.0    129.0    1.0
2.6
3  37.0  1.0  3.0     130.0  250.0  0.0      0.0    187.0    0.0
3.5
4  41.0  0.0  2.0     130.0  204.0  0.0      2.0    172.0    0.0
1.4

   slope   ca  thal  target
0    3.0  0.0   6.0       0
1    2.0  3.0   3.0       2
2    2.0  2.0   7.0       1
3    3.0  0.0   3.0       0
4    1.0  0.0   3.0       0
```

```
# Summary statistics for resting blood pressure and cholesterol
mean_bp = df['trestbps'].mean()
median_bp = df['trestbps'].median()
mode_bp = df['trestbps'].mode()[0]
std_bp = df['trestbps'].std()
var_bp = df['trestbps'].var()
```

```
mean_chol = df['chol'].mean()
median_chol = df['chol'].median()
mode_chol = df['chol'].mode()[0]
std_chol = df['chol'].std()
var_chol = df['chol'].var()

print(f"Resting Blood Pressure: mean={mean_bp}, median={median_bp},
mode={mode_bp}, std={std_bp}, var={var_bp}")
print(f"Cholesterol: mean={mean_chol}, median={median_chol},
mode={mode_chol}, std={std_chol}, var={var_chol}")
```

```
Resting Blood Pressure: mean=131.68976897689768, median=130.0,
mode=120.0, std=17.599747729587687, var=309.751120145127
Cholesterol: mean=246.69306930693068, median=241.0, mode=197.0,
std=51.776917542637015, var=2680.8491902170326
```

```
from scipy import stats

# Null hypothesis: The mean cholesterol level is 200 mg/dL
# Alternative hypothesis: The mean cholesterol level is different from
200 mg/dL
population_mean = 200
chol_values = df['chol'].dropna()

t_stat, p_value = stats.ttest_1samp(chol_values, population_mean)

print(f"t-statistic: {t_stat}, p-value: {p_value}")

# If p-value < 0.05, we reject the null hypothesis
if p_value < 0.05:
    print("Reject the null hypothesis: The average cholesterol level
is significantly different from 200 mg/dL.")
else:
    print("Fail to reject the null hypothesis: No significant
difference from 200 mg/dL.")
```

```
t-statistic: 15.697754943543861, p-value: 5.111676087498585e-41
Reject the null hypothesis: The average cholesterol level is
significantly different from 200 mg/dL.
```

```
import numpy as np

# Calculate the mean and standard error of the mean for cholesterol
mean_chol = np.mean(chol_values)
std_error = stats.sem(chol_values)

# Compute the confidence interval
confidence_interval = stats.t.interval(0.95, len(chol_values)-1,
loc=mean_chol, scale=std_error)
```

```
print(f"95% confidence interval for mean cholesterol:
{confidence_interval}")
```

```
95% confidence interval for mean cholesterol: (240.83968661733547,
252.5464519965259)
```

EXERCISE 5.2

```
import pandas as pd
import statsmodels.api as sm
import matplotlib.pyplot as plt
import seaborn as sns

# Load dataset (as shown earlier)
url =
"https://archive.ics.uci.edu/ml/machine-learning-databases/heart-
disease/processed.cleveland.data"
columns = ['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg',
'thalach', 'exang', 'oldpeak', 'slope', 'ca', 'thal', 'target']

# Load the dataset into a DataFrame
df = pd.read_csv(url, names=columns, na_values="?")

# Drop rows with missing values in 'age' or 'trestbps'
df_clean = df[['age', 'trestbps']].dropna()

# Independent variable (age)
X = df_clean['age']

# Dependent variable (resting blood pressure)
y = df_clean['trestbps']

# Add a constant to the independent variable (to account for the
intercept)
X = sm.add_constant(X)

# Fit the linear regression model
model = sm.OLS(y, X).fit()

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

```
                            OLS Regression Results

==============================================================================
========
Dep. Variable:                trestbps   R-squared:
0.081
Model:                            OLS   Adj. R-squared:
```

```
0.078
Method:                 Least Squares   F-statistic:
26.60
Date:                Sun, 08 Sep 2024   Prob (F-statistic):
4.55e-07
Time:                        16:05:25   Log-Likelihood:
-1285.6
No. Observations:                 303   AIC:
2575.
Df Residuals:                     301   BIC:
2583.
Df Model:                           1

Covariance Type:            nonrobust

==============================================================================
                 coef    std err          t      P>|t|      [0.025
0.975]
------------------------------------------------------------------------------
const        101.4851      5.936     17.095      0.000      89.803
113.167
age            0.5548      0.108      5.157      0.000       0.343
0.767
==============================================================================
Omnibus:                       22.051   Durbin-Watson:
1.898
Prob(Omnibus):                  0.000   Jarque-Bera (JB):
27.565
Skew:                           0.570   Prob(JB):
1.03e-06
Kurtosis:                       3.939   Cond. No.
338.
==============================================================================

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

```
# Create a scatter plot with the regression line
plt.figure(figsize=(10,6))
sns.regplot(x='age', y='trestbps', data=df_clean,
line_kws={"color":"red"})

# Label the axes and add a title
plt.xlabel('Age')
plt.ylabel('Resting Blood Pressure (trestbps)')
```

```
plt.title('Age vs Resting Blood Pressure with Regression Line')

# Show the plot
plt.show()
```