SRM VALLIAMMAI ENGINEERING COLLEGE

(An Autonomous Institution)

SRM Nagar, Kattankulathur-603203.

CP3264 - ADVANCED DATA SCIENCE LABORATORY MANUAL

Regulation 2023

I Year (II semester) M.E CSE

2024-2025 (EVEN SEMESTER)

Prepared by

Ms. V. PREMA, AP/CSE

CP3264 ADVANCED DATA SCIENCE LABORATORY

L T P C
0 0 3 1.5

OBJECTIVES

- To develop data analytic code in python
- To be able to use python libraries for handling data
- To develop analytical applications using python
- To perform data visualization using plots

LIST OF EXPERIMENTS

Tools: Python, Numpy, Scipy, Matplotlib, Pandas, statmodels, seaborn, plotly, bokeh
Working with Numpy arrays

1. Working with Pandas data frames
2. Basic plots using Matplotlib
3. Frequency distributions, Averages, Variability
4. Normal curves, Correlation and scatter plots, Correlation coefficient
5. Regression
6. Z-test
7. T-test
8. ANOVA
9. Building and validating linear models
10. Building and validating logistic models
11. Time series analysis

TOTAL: 45 PERIODS

OUTCOMES:

Upon successful completion of this course, students will be able to:

Write python programs to handle data using Numpy and Pandas

- Perform descriptive analytics
- Perform data exploration using Matplotlib
- Perform inferential data analytics
- Build models of predictive analytics

Experiment 1: Working with Pandas DataFrames

Aim:

To understand and implement Working with Pandas DataFrames using Python.

Algorithm:

1. Load or generate necessary data.

2. Apply relevant statistical/machine learning techniques.

3. Interpret results and validate findings.

Source Code:

```python
import pandas as pd


# Creating a sample DataFrame
data = {'Name': ['Alice', 'Bob', 'Charlie'],
        'Age': [25, 30, 35],
        'Score': [90, 85, 88]}


df = pd.DataFrame(data)
print(df)
```

Sample Output:

```
      Name  Age  Score
0    Alice   25     90
1      Bob   30     85
2  Charlie   35     88
```

Experiment 2: Basic plots using Matplotlib

Aim:

To understand and implement Basic plots using Matplotlib using Python.

Algorithm:

1. Load or generate necessary data.

2. Apply relevant statistical/machine learning techniques.

3. Interpret results and validate findings.

Source Code:

```python
import matplotlib.pyplot as plt


# Sample data
x = [1, 2, 3, 4, 5]
y = [10, 20, 25, 30, 40]


# Plotting
plt.plot(x, y, marker='o', linestyle='-')
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.title('Simple Line Plot')
plt.show()
```

Sample Output:

A line plot with points marked along the given X and Y values.

Experiment 3: Frequency distributions, Averages, Variability

Aim:

To understand and implement Frequency distributions, Averages, Variability using Python.

Algorithm:

1. Load or generate necessary data.
2. Apply relevant statistical/machine learning techniques.
3. Interpret results and validate findings.

Source Code:

```python
import numpy as np


# Sample data
data = [10, 20, 20, 30, 30, 30, 40, 40, 50]


# Computing statistics
mean = np.mean(data)
median = np.median(data)
std_dev = np.std(data)


print(f"Mean: {mean}, Median: {median}, Standard Deviation: {std_dev}")
```

Sample Output:

```
Mean: 30.0, Median: 30.0, Standard Deviation: 11.18
```

Experiment 4: Normal curves, Correlation and scatter plots, Correlation coefficient

Aim:

To understand and implement Normal curves, Correlation and scatter plots, Correlation coefficient using Python.

Algorithm:

1. Load or generate necessary data.
2. Apply relevant statistical/machine learning techniques.
3. Interpret results and validate findings.

Source Code:

```python
import numpy as np
import matplotlib.pyplot as plt

# Generate data
np.random.seed(0)
x = np.random.randn(100)
y = x * 2.5 + np.random.randn(100) * 0.5

# Compute correlation
correlation = np.corrcoef(x, y)[0, 1]

# Scatter plot
plt.scatter(x, y, alpha=0.5)
plt.xlabel("X values")
plt.ylabel("Y values")

plt.title(f"Scatter Plot (Correlation: {correlation:.2f})")

plt.show()
```

Sample Output:

A scatter plot with a correlation coefficient value displayed in the title.

Experiment 5: Regression

Aim:

To understand and implement Regression using Python.

Algorithm:

1. Load or generate necessary data.

2. Apply relevant statistical/machine learning techniques.

3. Interpret results and validate findings.

Source Code:

```python
import numpy as np
from sklearn.linear_model import LinearRegression


# Sample data
X = np.array([1, 2, 3, 4, 5]).reshape(-1, 1)
y = np.array([10, 15, 20, 25, 30])
# Fit regression model
model = LinearRegression()
model.fit(X, y)
# Predict
predicted = model.predict(X)
print("Predictions:", predicted)
```

Sample Output:

Predictions: [10. 15. 20. 25. 30.] (Approximated line of best fit)

Experiment 6: Z-test

Aim:

To understand and implement Z-test using Python.

Algorithm:

1. Load or generate necessary data.

2. Apply relevant statistical/machine learning techniques.

3. Interpret results and validate findings.

Source Code:

```python
from statsmodels.stats.weightstats import ztest


# Sample data

data = [100, 102, 98, 101, 99, 97, 103, 105, 96, 104]


# Perform Z-test

z_score, p_value = ztest(data, value=100)

print(f"Z-score: {z_score}, P-value: {p_value}")
```

Sample Output:

Z-score and p-value indicating significance of the test.

Experiment 7: T-test

Aim:

To understand and implement T-test using Python.

Algorithm:

1. Load or generate necessary data.

2. Apply relevant statistical/machine learning techniques.

3. Interpret results and validate findings.

Source Code:

```python
from scipy.stats import ttest_1samp


# Sample data
data = [50, 52, 47, 49, 51, 48, 53, 55, 46, 54]


# Perform T-test
t_stat, p_value = ttest_1samp(data, 50)
print(f"T-statistic: {t_stat}, P-value: {p_value}")
```

Sample Output:

T-statistic and p-value to determine if the sample mean significantly differs from 50.

Experiment 8: ANOVA

Aim:

To understand and implement ANOVA using Python.

Algorithm:

1. Load or generate necessary data.

2. Apply relevant statistical/machine learning techniques.

3. Interpret results and validate findings.

Source Code:

```python
from scipy.stats import f_oneway

# Sample groups
group1 = [10, 12, 14, 15, 16]
group2 = [20, 22, 24, 26, 28]
group3 = [30, 32, 34, 36, 38]

# Perform ANOVA test
f_stat, p_value = f_oneway(group1, group2, group3)
print(f"F-statistic: {f_stat}, P-value: {p_value}")
```

Sample Output:

F-statistic and p-value indicating whether there is a significant difference among groups.

Experiment 9: Building and validating linear models

Aim:

To understand and implement Building and validating linear models using Python.

Algorithm:

1. Load or generate necessary data.

2. Apply relevant statistical/machine learning techniques.

3. Interpret results and validate findings.

Source Code:

```python
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
import numpy as np

# Sample data
X = np.array([1, 2, 3, 4, 5]).reshape(-1, 1)
y = np.array([10, 15, 20, 25, 30])

# Split data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

# Train model
model = LinearRegression()
model.fit(X_train, y_train)
```

```
# Validate model
score = model.score(X_test, y_test)
print(f"Model R^2 Score: {score}")
```

Sample Output:

R-squared score indicating model performance.

Experiment 10: Building and validating logistic models

Aim:

To understand and implement Building and validating logistic models using Python.

Algorithm:

1. Load or generate necessary data.

2. Apply relevant statistical/machine learning techniques.

3. Interpret results and validate findings.

Source Code:

```python
from sklearn.linear_model import LogisticRegression
from sklearn.datasets import make_classification

# Generate sample data
X, y = make_classification(n_samples=100, n_features=2, random_state=0)

# Train logistic regression model
model = LogisticRegression()
model.fit(X, y)

# Predict
predictions = model.predict(X[:5])
print("Sample Predictions:", predictions)
```

Sample Output:

Predicted class labels for the first five samples.

Experiment 11: Time series analysis

Aim:

To understand and implement Time series analysis using Python.

Algorithm:

1. Load or generate necessary data.

2. Apply relevant statistical/machine learning techniques.

3. Interpret results and validate findings.

Source Code:

```python
import pandas as pd
import numpy as np

# Generate sample time series data
dates = pd.date_range(start="2023-01-01", periods=10, freq="D")
values = np.random.randint(50, 100, size=10)

# Create DataFrame
df = pd.DataFrame({'Date': dates, 'Value': values})
df.set_index('Date', inplace=True)
print(df)
```

Sample Output:

Time series data table with dates as the index.