MADRAS INSTITUTE OF TECHNOLOGY
ANNA UNIVERSITY
DEPARTMENT OF INFORMATION TECHNOLOGY

IT5612 DATA ANALYTICS &
CLOUD COMPUTING LABORATORY

LAB RECORD
REGULATION – 2019

NAME : ARUN KISHORE R

REG NO : 2022506053

DEPARTMENT : DEPARTMENT OF INFORMATION TECHNOLOGY

SUBJECT CODE : IT5612

SUBJECT TITLE : DATA ANALYTICS AND CLOUD COMPUTING LABORATORY

ANNA UNIVERSITY

MADRAS INSTITUTE OF TECHNOLOGY CHROMPET, CHENNAI-600 044.

BONAFIDE CERTIFICATE

NAME : ARUN KISHORE R

SUBJECT CODE : IT5612

SUBJECT TITLE : DATA ANALYTICS AND CLOUD COMPUTING LABORATORY

REGISTER NO : 2022506053

Certified that the bonafide record of practical work done by Arun Kishore R in the Laboratory subject code IT5611 during the period JANUARY 2025 - APRIL 2025

DATE: 29-04-2025 **COURSE-IN-CHARGE**

Submitted for the Practical Examination held on 30-04-2025

Examiners

1.
2.

List of Experiments

EXP NO : 1 DATE : 8/1/25	DATA EXTRACTION FROM DIFFERENT FILE TYPES

AIM :

To extract data from different file types like csv , excel using python.

SOURCE CODE :

1. From csv file

```
import pandas as pd
file_path = 'iris_with_headers.csv'  # Adjust the path if necessary
df = pd.read_csv(file_path)
print("Column Headings:", df.columns.tolist())

for index, row in df.iterrows():
    print(row.tolist())
print("\nFull Dataset:")
print(df)

print("\nFirst 5 Rows:")
print(df.head())

print("\nFirst 10 Rows:")
print(df.head(10))

print("\nLast 5 Rows:")
print(df.tail())
```

OUTPUT :

```
Column Headings: ['sepal_length', 'sepal_width', 'petal_length', 'petal_width', 'species']
[5.1, 3.5, 1.4, 0.2, 'Iris-setosa']
[4.9, 3.0, 1.4, 0.2, 'Iris-setosa']
[4.7, 3.2, 1.3, 0.2, 'Iris-setosa']
[4.6, 3.1, 1.5, 0.2, 'Iris-setosa']
[5.0, 3.6, 1.4, 0.2, 'Iris-setosa']
[5.4, 3.9, 1.7, 0.4, 'Iris-setosa']
```

```
Full Dataset:
     sepal_length  sepal_width  petal_length  petal_width         species
0             5.1          3.5           1.4          0.2     Iris-setosa
1             4.9          3.0           1.4          0.2     Iris-setosa
2             4.7          3.2           1.3          0.2     Iris-setosa
3             4.6          3.1           1.5          0.2     Iris-setosa
4             5.0          3.6           1.4          0.2     Iris-setosa
..            ...          ...           ...          ...             ...
145           6.7          3.0           5.2          2.3  Iris-virginica
146           6.3          2.5           5.0          1.9  Iris-virginica
147           6.5          3.0           5.2          2.0  Iris-virginica
148           6.2          3.4           5.4          2.3  Iris-virginica
149           5.9          3.0           5.1          1.8  Iris-virginica

[150 rows x 5 columns]

First 5 Rows:
   sepal_length  sepal_width  petal_length  petal_width      species
0           5.1          3.5           1.4          0.2  Iris-setosa
1           4.9          3.0           1.4          0.2  Iris-setosa
2           4.7          3.2           1.3          0.2  Iris-setosa
3           4.6          3.1           1.5          0.2  Iris-setosa
4           5.0          3.6           1.4          0.2  Iris-setosa
```

```
First 10 Rows:
   sepal_length  sepal_width  petal_length  petal_width      species
0           5.1          3.5           1.4          0.2  Iris-setosa
1           4.9          3.0           1.4          0.2  Iris-setosa
2           4.7          3.2           1.3          0.2  Iris-setosa
3           4.6          3.1           1.5          0.2  Iris-setosa
4           5.0          3.6           1.4          0.2  Iris-setosa
5           5.4          3.9           1.7          0.4  Iris-setosa
6           4.6          3.4           1.4          0.3  Iris-setosa
7           5.0          3.4           1.5          0.2  Iris-setosa
8           4.4          2.9           1.4          0.2  Iris-setosa
9           4.9          3.1           1.5          0.1  Iris-setosa

Last 5 Rows:
     sepal_length  sepal_width  petal_length  petal_width         species
145           6.7          3.0           5.2          2.3  Iris-virginica
146           6.3          2.5           5.0          1.9  Iris-virginica
147           6.5          3.0           5.2          2.0  Iris-virginica
148           6.2          3.4           5.4          2.3  Iris-virginica
149           5.9          3.0           5.1          1.8  Iris-virginica
```

2. From excel file

import pandas as pd

file_path = 'hospital_data.xlsx'

data = pd.read_excel(file_path)

print("Headers:", data.columns)

print("First 5 rows:")

print(data.head())

print("Last 5 rows:")

print(data.tail())

OUTPUT :

```
Headers: Index(['PatientID', 'Date', 'ArrivalTime', 'WaitTimeMinutes', 'VisitType',
       'Complaint', 'StaffAvailable', 'PatientAgeGroup', 'Department',
       'DayOfWeek'],
      dtype='object')
First 5 rows:
   PatientID        Date ArrivalTime  WaitTimeMinutes    VisitType Complaint  StaffAvailable PatientAgeGroup  Department DayOfWeek
0          1  2024-11-01       08:15               45      Walk-In       Yes               5           Adult     General    Monday
1          2  2024-11-01       09:30               30  Appointment        No               6          Senior  Pediatrics    Monday
2          3  2024-11-01       10:00               60    Emergency       Yes               4           Adult   Emergency    Monday
3          4  2024-11-02       11:00               20  Appointment        No               7           Child     General   Tuesday
4          5  2024-11-02       14:00               90      Walk-In       Yes               3          Senior   Emergency   Tuesday
Last 5 rows:
   PatientID        Date ArrivalTime  WaitTimeMinutes    VisitType Complaint  StaffAvailable PatientAgeGroup  Department  DayOfWeek
5          6  2024-11-02       15:30               55    Emergency       Yes               4           Adult   Emergency    Tuesday
6          7  2024-11-03       09:00               25  Appointment        No               6           Adult  Pediatrics  Wednesday
7          8  2024-11-03       13:15               80      Walk-In       Yes               2          Senior     General  Wednesday
8          9  2024-11-04       08:45               50    Emergency       Yes               4           Adult   Emergency   Thursday
9         10  2024-11-04       10:30               15  Appointment        No               8           Child  Pediatrics   Thursday
```

3. From image dataset :

import os

from PIL import Image

import matplotlib.pyplot as plt

dataset_path = 'dataset\\Images\\Train\\audi'

image_files = [f for f in os.listdir(dataset_path) if f.endswith('.jpg') or f.endswith('.png')]

for image_file in image_files:

    image_path = os.path.join(dataset_path, image_file)

    image = Image.open(image_path)

    plt.imshow(image)

```
plt.axis('off')  # Hide the axes
plt.show()
```

OUTPUT :

4. From text file

```
filepath= “text.txt”
with open(filepath,mode=‘r’ )  as file:
        content=file.read()
        print(content)
```

OUTPUT :

```
PS C:\Users\Admin\dacc> python text_file.py
This is Moni
From MIT
Btech in IT
```

RESULT :

Thus extracting data from different file types like csv , excel using python have been done successfully.

EXP NO : 2 DATE : 22/1/25	DESCRIPTIVE ANALYTICS ON UNGROUPED AND GROUPED DATA

AIM :

To perform descriptive analytics on ungrouped and grouped data.

SOURCE CODE :

```
import pandas as pd

import numpy as np

import numpy as np

from scipy import stats

import statistics


df = pd.read_csv("usedcars.csv")

print(df.head())


df[["price"]] = df[["price"]].astype("float")

price_data = df["price"].tolist()

def calculate_mean(data):

    return sum(data) / len(data)

def calculate_median(data):

    sorted_data = sorted(data)

    n = len(sorted_data)

    mid = n // 2

    if n % 2 == 0:

        return (sorted_data[mid - 1] + sorted_data[mid]) / 2
```

```
    else:
        return sorted_data[mid]

def calculate_mode(data):
    frequency = {}
    for value in data:
        frequency[value] = frequency.get(value, 0) + 1
    max_freq = max(frequency.values())
    modes = [key for key, val in frequency.items() if val == max_freq]
    return modes
def calculate_variance(data):
    mean = calculate_mean(data)
    return sum((x - mean) ** 2 for x in data) / len(data)
def calculate_std_dev(data):
    return np.sqrt(calculate_variance(data))

mean_formula = calculate_mean(price_data)
median_formula = calculate_median(price_data)
mode_formula = calculate_mode(price_data)
variance_formula = calculate_variance(price_data)
std_dev_formula = calculate_std_dev(price_data)
mean_builtin = np.mean(price_data)
median_builtin = np.median(price_data)
mode_builtin = statistics.multimode(price_data)  # Using scipy.stats for mode
variance_builtin = np.var(price_data)
std_dev_builtin = np.std(price_data)
```

```
print("Using Formulas:")
print(f"{calculate_mean(price_data)}")
print(f"Median: {calculate_median(price_data)}")
print(f"Mode: {mode_formula}")
print(f"Variance: {variance_formula}")
print(f"Standard Deviation: {std_dev_formula}\n")


print("Using Built-in Functions:")
print(f"Mean: {mean_builtin}")
print(f"Median: {median_builtin}")
print(f"Mode: {mode_builtin}")
print(f"Variance: {variance_builtin}")
print(f"Standard Deviation: {std_dev_builtin}\n")
price_bins = [0, 10000, 20000, 30000, 40000, 50000]  # You can adjust these bins based on your dataset
price_labels = ["0-10000", "10001-20000", "20001-30000", "30001-40000", "40001-50000"]
df["price_group"] = pd.cut(df["price"], bins=price_bins, labels=price_labels, include_lowest=True)
grouped_stats = df.groupby("price_group")["price"].agg(
    ["mean", "median", lambda x: x.mode()[0]]
)
grouped_stats.columns = ["Mean", "Median", "Mode"]
print(grouped_stats)
import pandas as pd
price_bins = [0, 10000, 20000, 30000, 40000, 50000]
price_labels = ["0-10000", "10001-20000", "20001-30000", "30001-40000", "40001-50000"]
df["price_group"] = pd.cut(df["price"], bins=price_bins, labels=price_labels, include_lowest=True)
```

```
price_group_counts = df["price_group"].value_counts()
frequency_stats = pd.DataFrame({
    "Mean": [price_group_counts.mean()],
    "Median": [price_group_counts.median()],
    "Mode": [price_group_counts.mode()[0]]  # Mode will give the most frequent bin
})
print(frequency_stats)
```

OUTPUT :

```
Using Formulas:
Mean: 13285.025906735751
Median: 10245.0
Mode: [16500.0, 5572.0, 7957.0, 6229.0, 6692.0, 7609.0, 8921.0, 7295.0, 8845.0, 8495.0, 9279.0, 13499.0, 18150.0, 7775.0, 7898.0]
Variance: 65094229.48637547
Standard Deviation: 8068.0994965589925

Using Built-in Functions:
Mean: 13285.025906735751
Median: 10245.0
Mode: [16500.0, 5572.0, 7957.0, 6229.0, 6692.0, 7609.0, 8921.0, 7295.0, 8845.0, 8495.0, 9279.0, 13499.0, 18150.0, 7775.0, 7898.0]
Variance: 65094229.48637547
Standard Deviation: 8068.0994965589925
```

```
                     Mean   Median     Mode
price_group
0-10000       7649.789474   7689.0   5572.0
10001-20000  14693.863014  14869.0  13499.0
20001-30000  23735.363636  22625.0  20970.0
30001-40000  34169.454545  34184.0  30760.0
40001-50000  42558.333333  41315.0  40960.0
   Mean  Median  Mode
0  38.6    11.0    11
```

RESULT :

Thus performing descriptive analytics on ungrouped and grouped data have been done successfully.

EXP NO : 3 **DATE : 29/1/25**	**UNIVARIATE AND BIVARIATE ANALYSES**

AIM :

To perform univariate and bivariate analyses on a dataset.

SOURCE CODE :

1. UNIVARIATE ANALYSIS ON A SIMPLE DATASET :

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import skew, kurtosis
from sklearn.datasets import load_iris
# Load the Iris dataset
iris = load_iris()
df = pd.DataFrame(iris.data, columns=iris.feature_names)
# Select the first feature
feature = df.columns[0]
data = df[feature].values

# Manual calculations
n = len(data)
mean_manual = np.sum(data) / n
median_manual = np.median(data)
mode_manual = pd.Series(data).mode()[0]
variance_manual = np.sum((data - mean_manual) ** 2) / (n - 1)
std_dev_manual = np.sqrt(variance_manual)
skewness_manual = np.sum((data - mean_manual) ** 3) / ((n - 1) * std_dev_manual ** 3)
```

```
kurtosis_manual = np.sum((data - mean_manual) ** 4) / ((n - 1) * std_dev_manual ** 4)

# Built-in functions
mean_builtin = df[feature].mean()
median_builtin = df[feature].median()
mode_builtin = df[feature].mode()[0]
variance_builtin = df[feature].var()
std_dev_builtin = df[feature].std()
skewness_builtin = skew(df[feature])
kurtosis_builtin = kurtosis(df[feature])

print(f"Feature: {feature}\n")
print("Manual Calculations:")
print(f"Mean: {mean_manual:.4f}")
print(f"Median: {median_manual:.4f}")
print(f"Mode: {mode_manual:.4f}")
print(f"Variance: {variance_manual:.4f}")
print(f"Standard Deviation: {std_dev_manual:.4f}")
print(f"Skewness: {skewness_manual:.4f}")
print(f"Kurtosis: {kurtosis_manual:.4f}")

print("\nBuilt-in Functions:")
print(f"Mean: {mean_builtin:.4f}")
print(f"Median: {median_builtin:.4f}")
print(f"Mode: {mode_builtin:.4f}")
print(f"Variance: {variance_builtin:.4f}")
print(f"Standard Deviation: {std_dev_builtin:.4f}")
print(f"Skewness: {skewness_builtin:.4f}")
print(f"Kurtosis: {kurtosis_builtin:.4f}")
```

OUTPUT :

```
Feature: sepal length (cm)

Manual Calculations:
Mean: 5.8433
Median: 5.8000
Mode: 5.0000
Variance: 0.6857
Standard Deviation: 0.8281
Skewness: 0.3107
Kurtosis: 2.4103

Built-in Functions:
Mean: 5.8433
Median: 5.8000
Mode: 5.0000
Variance: 0.6857
Standard Deviation: 0.8281
Skewness: 0.3118
Kurtosis: -0.5736
```

2. BIVARIATE ANALYSIS ON A SIMPLE DATASET :

```
import numpy as np
import pandas as pd
from sklearn.datasets import load_iris
iris = load_iris()
df = pd.DataFrame(iris.data, columns=iris.feature_names)

def compute_covariance_matrix(data):
    mean_vector = data.mean()
    centered_data = data - mean_vector
    n = len(data)
    covariance_matrix = (centered_data.T @ centered_data) / (n - 1)
    return covariance_matrix

def compute_correlation_matrix(data):
    covariance_matrix = compute_covariance_matrix(data)
    std_devs = np.sqrt(np.diag(covariance_matrix))
    correlation_matrix = covariance_matrix / np.outer(std_devs, std_devs)
```

```
    return correlation_matrix

cov_whole_manual = compute_covariance_matrix(df)
corr_whole_manual = compute_correlation_matrix(df)
print("Using manual functions : ")
print("Covariance Matrix (Whole Dataset) :\n", cov_whole_manual)

sample_df = df.sample(n=20, random_state=42)

cov_sample_manual = compute_covariance_matrix(sample_df)
corr_sample_manual = compute_correlation_matrix(sample_df)

print("\nCovariance Matrix (Sample) :\n", cov_sample_manual)
print("\nCorrelation Matrix (Whole Dataset)  :\n", corr_whole_manual)
print("\nUsing built in functions ")
cov_whole = df.cov()
print("Covariance Matrix (Whole Dataset):\n", cov_whole)
sample_df = df.sample(n=20, random_state=42)
cov_sample = sample_df.cov()
print("\nCovariance Matrix (Sample):\n", cov_sample)
corr_whole = df.corr()
print("\nCorrelation Matrix (Whole Dataset):\n", corr_whole)
```

OUTPUT :

```
Using manual functions :
Covariance Matrix (Whole Dataset) :
                   sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)
sepal length (cm)           0.685694         -0.042434           1.274315          0.516271
sepal width (cm)           -0.042434          0.189979          -0.329656         -0.121639
petal length (cm)           1.274315         -0.329656           3.116278          1.295609
petal width (cm)            0.516271         -0.121639           1.295609          0.581006

Covariance Matrix (Sample) :
                   sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)
sepal length (cm)           0.512500         -0.106053           1.088026          0.477500
sepal width (cm)           -0.106053          0.166211          -0.425000         -0.153421
petal length (cm)           1.088026         -0.425000           2.981974          1.271447
petal width (cm)            0.477500         -0.153421           1.271447          0.586184

Correlation Matrix (Whole Dataset)  :
                   sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)
sepal length (cm)           1.000000         -0.117570           0.871754          0.817941
sepal width (cm)           -0.117570          1.000000          -0.428440         -0.366126
petal length (cm)           0.871754         -0.428440           1.000000          0.962865
petal width (cm)            0.817941         -0.366126           0.962865          1.000000
```

```
Using built in functions
Covariance Matrix (Whole Dataset):
                   sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)
sepal length (cm)           0.685694         -0.042434           1.274315          0.516271
sepal width (cm)           -0.042434          0.189979          -0.329656         -0.121639
petal length (cm)           1.274315         -0.329656           3.116278          1.295609
petal width (cm)            0.516271         -0.121639           1.295609          0.581006

Covariance Matrix (Sample):
                   sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)
sepal length (cm)           0.512500         -0.106053           1.088026          0.477500
sepal width (cm)           -0.106053          0.166211          -0.425000         -0.153421
petal length (cm)           1.088026         -0.425000           2.981974          1.271447
petal width (cm)            0.477500         -0.153421           1.271447          0.586184

Correlation Matrix (Whole Dataset):
                   sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)
sepal length (cm)           1.000000         -0.117570           0.871754          0.817941
sepal width (cm)           -0.117570          1.000000          -0.428440         -0.366126
petal length (cm)           0.871754         -0.428440           1.000000          0.962865
petal width (cm)            0.817941         -0.366126           0.962865          1.000000
```

3. UNIVARIATE ANALYSIS ON A TIME SERIES DATASET :

```
import numpy as np
import pandas as pd
import scipy.stats as stats
import matplotlib.pyplot as plt


df = pd.read_csv('AAPL.csv', skiprows=2, parse_dates=['Date'], index_col='Date')
```

```
#Select the Closing Price as our Univariate Time Series Data
data = df['Close']
mean_manual = round(sum(data) / len(data), 2)
sorted_data = sorted(data)
n = len(sorted_data)
if n % 2 == 0:
    median_manual = round((sorted_data[n//2 - 1] + sorted_data[n//2]) / 2, 2)
else:
    median_manual = round(sorted_data[n//2], 2)
mode_manual = round(max(set(data), key=list(data).count), 2)
variance_manual = round(sum((x - mean_manual) ** 2 for x in data) / len(data), 2)
std_dev_manual = round(variance_manual ** 0.5, 2)
skew_manual = round(sum((x - mean_manual) ** 3 for x in data) / ((n-1) * (std_dev_manual ** 3)), 2)
kurtosis_manual = round(sum((x - mean_manual) ** 4 for x in data) / ((n-1) * (std_dev_manual ** 4)), 2)

# Display Manual Calculations
print("Manual Calculations:")
print(f"Mean: {mean_manual}")
print(f"Median: {median_manual}")
print(f"Mode: {mode_manual}")
print(f"Variance: {variance_manual}")
print(f"Standard Deviation: {std_dev_manual}")
print(f"Skewness: {skew_manual}")
print(f"Kurtosis: {kurtosis_manual}")

# Compare with Built-in Functions
print("\nBuilt-in Calculations:")
print(f"Mean: {round(data.mean(), 2)}")
```

```
print(f"Median: {round(data.median(), 2)}")
print(f"Mode: {round(data.mode()[0], 2)}")
print(f"Variance: {round(data.var(), 2)}")
print(f"Standard Deviation: {round(data.std(), 2)}")
print(f"Skewness: {round(data.skew(), 2)}")
print(f"Kurtosis: {round(data.kurtosis(), 2)}")
```

OUTPUT :

```
Manual Calculations:
Mean: 138.56
Median: 143.44
Mode: 118.95
Variance: 1121.35
Standard Deviation: 33.49
Skewness: -0.58
Kurtosis: 2.7

Built-in Calculations:
Mean: 138.56
Median: 143.44
Mode: 88.6
Variance: 1122.47
Standard Deviation: 33.5
Skewness: -0.58
Kurtosis: -0.3
```

4. BIVARIATE ANALYSIS ON A TIME SERIES DATASET

```
import numpy as np
import pandas as pd
df = pd.read_csv('AAPL.csv', skiprows=2, parse_dates=['Date'], index_col='Date')
X = df['Close']
Y = df['Volume']
n = len(X)

mean_X = X.mean()
mean_Y = Y.mean()
cov_manual = sum((X.iloc[i] - mean_X) * (Y.iloc[i] - mean_Y) for i in range(n)) / (n - 1)
```

```
std_X = (sum((X.iloc[i] - mean_X) ** 2 for i in range(n)) / (n - 1)) ** 0.5
std_Y = (sum((Y.iloc[i] - mean_Y) ** 2 for i in range(n)) / (n - 1)) ** 0.5
corr_manual = cov_manual / (std_X * std_Y)
cov_builtin = np.cov(X, Y, ddof=1)[0, 1]  # Covariance
corr_builtin = np.corrcoef(X, Y)[0, 1]    # Correlation
print("Manual Calculations:")
print(f"Covariance: {round(cov_manual, 2)}")
print(f"Correlation: {round(corr_manual, 2)}")

print("\nBuilt-in Calculations:")
print(f"Covariance: {round(cov_builtin, 2)}")
print(f"Correlation: {round(corr_builtin, 2)}")
```

OUTPUT :

```
Manual Calculations:
Covariance: -1181239281.66
Correlation: -0.65

Built-in Calculations:
Covariance: -1181239281.66
Correlation: -0.65
```

RESULT :

Thus performing univariate and bivariate analyses on a dataset have been done successfully.

EXP NO : 4 DATE : 5/2/25	VISUALISATION WITH PLOTTING FUNCTIONS

AIM :

To perform data visualization on a dataset using multiple plots like bar chart , pie chart, scatter plot etc.

SOURCE CODE :

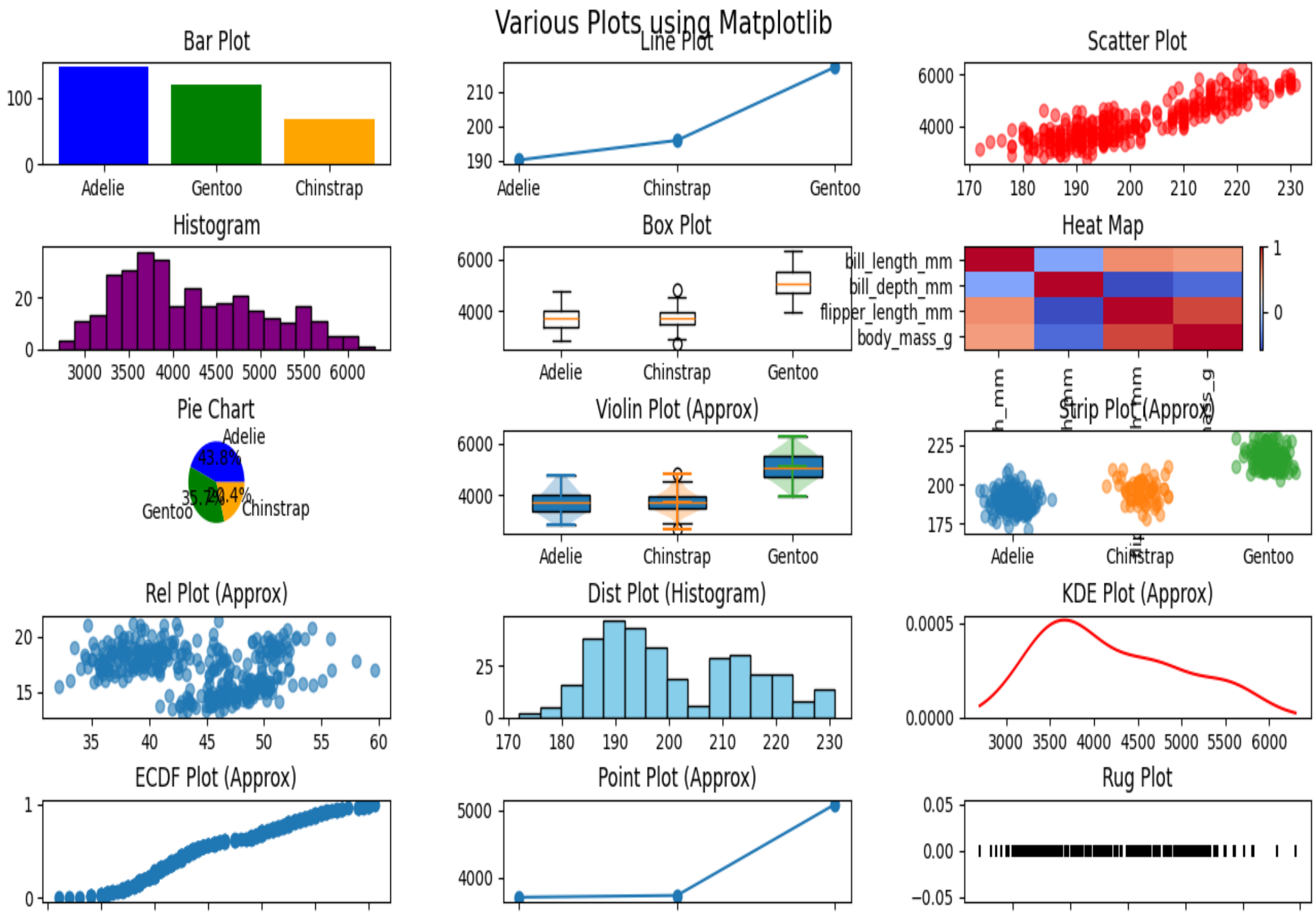
1.Matplotlib

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
df = pd.read_csv("penguins_dataset.csv")
df.dropna(inplace=True)
fig, axes = plt.subplots(5, 3, figsize=(18, 20))
fig.suptitle("Various Plots using Matplotlib", fontsize=16)
# Bar Plot
species_counts = df['species'].value_counts()
axes[0, 0].bar(species_counts.index, species_counts.values, color=['blue', 'green', 'orange'])
axes[0, 0].set_title("Bar Plot")
# Line Plot
axes[0, 1].plot(df.groupby("species")["flipper_length_mm"].mean(), marker='o', linestyle='-')
axes[0, 1].set_title("Line Plot")
# Scatter Plot
axes[0, 2].scatter(df['flipper_length_mm'], df['body_mass_g'], c='r', alpha=0.5)
axes[0, 2].set_title("Scatter Plot")
# Histogram
axes[1, 0].hist(df['body_mass_g'], bins=20, color='purple', edgecolor='black')
axes[1, 0].set_title("Histogram")
```

```
# Box Plot
box_data = [df[df['species'] == sp]['body_mass_g'] for sp in df['species'].unique()]
axes[1, 1].boxplot(box_data, labels=df['species'].unique())
axes[1, 1].set_title("Box Plot")
# Heat Map (Correlation Matrix)
corr_matrix = df.corr(numeric_only=True)
im = axes[1, 2].imshow(corr_matrix, cmap='coolwarm', aspect='auto')
fig.colorbar(im, ax=axes[1, 2])
axes[1, 2].set_xticks(range(len(corr_matrix.columns)))
axes[1, 2].set_yticks(range(len(corr_matrix.columns)))
axes[1, 2].set_xticklabels(corr_matrix.columns, rotation=90)
axes[1, 2].set_yticklabels(corr_matrix.columns)
axes[1, 2].set_title("Heat Map")
# Pie Chart
axes[2, 0].pie(species_counts, labels=species_counts.index, autopct='%1.1f%%', colors=['blue',
'green', 'orange'])
axes[2, 0].set_title("Pie Chart")
# Violin Plot
for i, sp in enumerate(df['species'].unique()):
    species_data = df[df['species'] == sp]['body_mass_g']
    axes[2, 1].boxplot(species_data, positions=[i + 1], widths=0.5, patch_artist=True)
    axes[2, 1].violinplot(species_data, positions=[i + 1], widths=0.5, showmeans=True)
axes[2, 1].set_xticks(range(1, len(df['species'].unique()) + 1))
axes[2, 1].set_xticklabels(df['species'].unique())
axes[2, 1].set_title("Violin Plot (Approx)")
# Strip Plot
for i, sp in enumerate(df['species'].unique()):
    species_data = df[df['species'] == sp]['flipper_length_mm']
    jitter = np.random.normal(0, 0.1, size=len(species_data))
    axes[2, 2].scatter(np.full_like(species_data, i) + jitter, species_data, alpha=0.5)
```

```
axes[2, 2].set_xticks(range(len(df['species'].unique())))
axes[2, 2].set_xticklabels(df['species'].unique())
axes[2, 2].set_title("Strip Plot (Approx)")
# Rel Plot
axes[3, 0].scatter(df['bill_length_mm'], df['bill_depth_mm'], alpha=0.6)
axes[3, 0].set_title("Rel Plot (Approx)")
# Dist Plot
axes[3, 1].hist(df['flipper_length_mm'], bins=15, color='skyblue', edgecolor='black')
axes[3, 1].set_title("Dist Plot (Histogram)")
# KDE Plot
from scipy.stats import gaussian_kde
density = gaussian_kde(df['body_mass_g'])
x_vals = np.linspace(df['body_mass_g'].min(), df['body_mass_g'].max(), 100)
axes[3, 2].plot(x_vals, density(x_vals), color='red')
axes[3, 2].set_title("KDE Plot (Approx)")
# ECDF Plot
sorted_data = np.sort(df['flipper_length_mm'])
y_vals = np.arange(1, len(sorted_data) + 1) / len(sorted_data)
axes[4, 0].plot(sorted_data, y_vals, marker='o', linestyle='none')
axes[4, 0].set_title("ECDF Plot (Approx)")
# Point Plot
axes[4, 1].plot(df.groupby("species")["body_mass_g"].mean(), marker='o', linestyle='-')
axes[4, 1].set_title("Point Plot (Approx)")
# Rug Plot
axes[4, 2].scatter(df['bill_length_mm'], [0] * len(df), marker='|', color='black')
axes[4, 2].set_title("Rug Plot")
plt.tight_layout(rect=[0, 0, 1, 0.96])
plt.show()
```

OUTPUT :



2.SEABORN

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np


df = pd.read_csv("penguins_dataset.csv")
df.dropna(inplace=True)
sns.set(style="whitegrid")
fig, axes = plt.subplots(5, 3, figsize=(18, 25))
fig.suptitle("Penguins Dataset Visualizations using Seaborn", fontsize=16)
```

 # Bar Plot

sns.barplot(x=df["species"].value_counts().index, y=df["species"].value_counts().values, ax=axes[0, 0])

axes[0, 0].set_title("Bar Plot")

 # Line Plot

sns.lineplot(x=df.index, y=df["flipper_length_mm"], hue=df["species"], ax=axes[0, 1])

axes[0, 1].set_title("Line Plot")

 # Scatter Plot

sns.scatterplot(x=df["bill_length_mm"], y=df["bill_depth_mm"], hue=df["species"], ax=axes[0, 2])

axes[0, 2].set_title("Scatter Plot")

 # Histogram

sns.histplot(df["flipper_length_mm"], bins=20, kde=True, ax=axes[1, 0])

axes[1, 0].set_title("Histogram")

 # Box Plot

sns.boxplot(x=df["species"], y=df["flipper_length_mm"], ax=axes[1, 1])

axes[1, 1].set_title("Box Plot")

 # Heat Map

corr = df.select_dtypes(include=['number']).corr()

sns.heatmap(corr, annot=True, cmap="coolwarm", ax=axes[1, 2])

axes[1, 2].set_title("Heat Map")

 # Pie Chart

sns.barplot(x=df["species"].value_counts().index, y=df["species"].value_counts().values, ax=axes[2, 0])

axes[2, 0].set_title("Pie Chart (Bar Representation)")

 # Violin Plot

sns.violinplot(x=df["species"], y=df["flipper_length_mm"], ax=axes[2, 1])

axes[2, 1].set_title("Violin Plot")

 # Strip Plot

sns.stripplot(x=df["species"], y=df["flipper_length_mm"], ax=axes[2, 2])

axes[2, 2].set_title("Strip Plot")

 # Rel Plot

sns.scatterplot(x=df.index, y=df["flipper_length_mm"], hue=df["species"], ax=axes[3, 0])

axes[3, 0].set_title("Rel Plot")

 # Dist Plot

sns.histplot(df["bill_length_mm"], bins=20, kde=True, ax=axes[3, 1])

axes[3, 1].set_title("Dist Plot")

 # KDE Plot

sns.kdeplot(df["bill_length_mm"], ax=axes[3, 2])

axes[3, 2].set_title("KDE Plot")

 # ECDF Plot

sns.ecdfplot(df["flipper_length_mm"], ax=axes[4, 0])

axes[4, 0].set_title("ECDF Plot")

 # Point Plot

sns.pointplot(x=df["species"], y=df["flipper_length_mm"], ax=axes[4, 1])

axes[4, 1].set_title("Point Plot")

 # Rug Plot

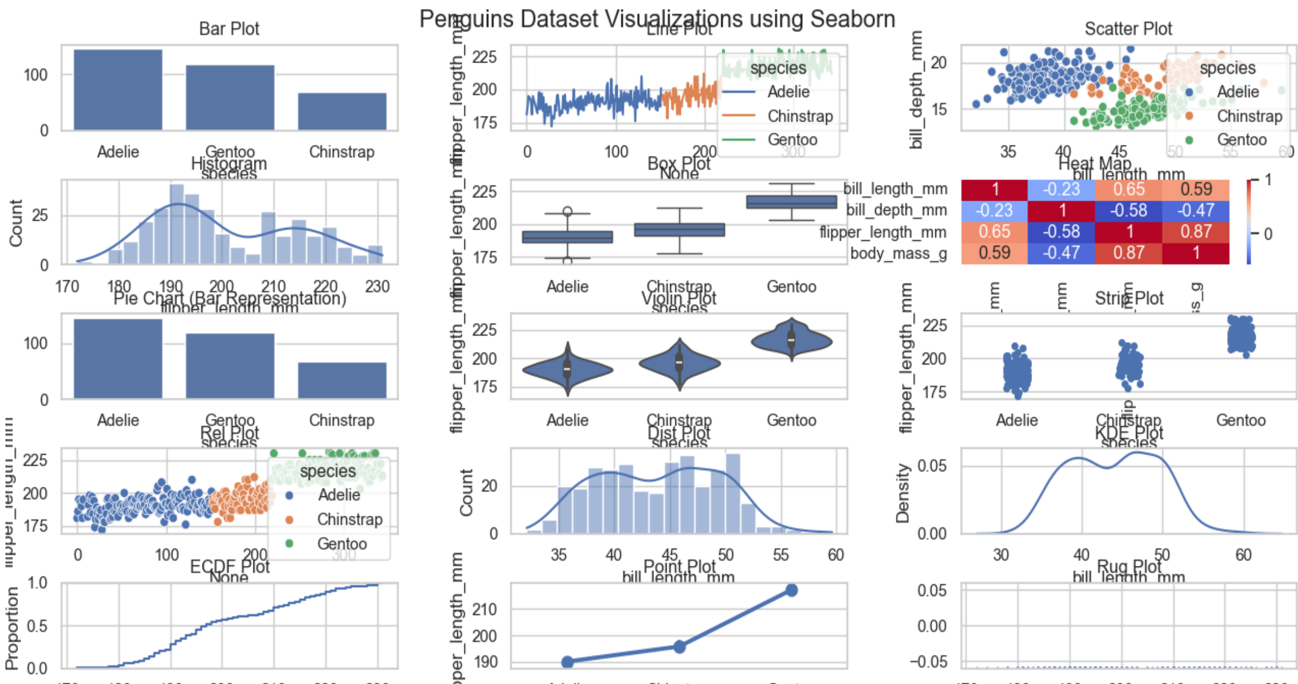
sns.rugplot(df["flipper_length_mm"], ax=axes[4, 2])

axes[4, 2].set_title("Rug Plot")

plt.tight_layout(rect=[0, 0, 1, 0.96])

plt.show()

OUTPUT :



3.Bokeh

```
import pandas as pd
from bokeh.layouts import gridplot
from bokeh.plotting import figure, show
from bokeh.io import output_file
from bokeh.models import ColumnDataSource, FactorRange, LinearColorMapper, ColorBar
from bokeh.transform import factor_cmap, linear_cmap
import numpy as np
from math import pi
from scipy.stats import gaussian_kde
from statsmodels.distributions.empirical_distribution import ECDF
df = pd.read_csv("penguins_dataset.csv")
df.dropna(inplace=True)
output_file("penguins_visualization.html")
df["species"] = df["species"].astype(str)
```

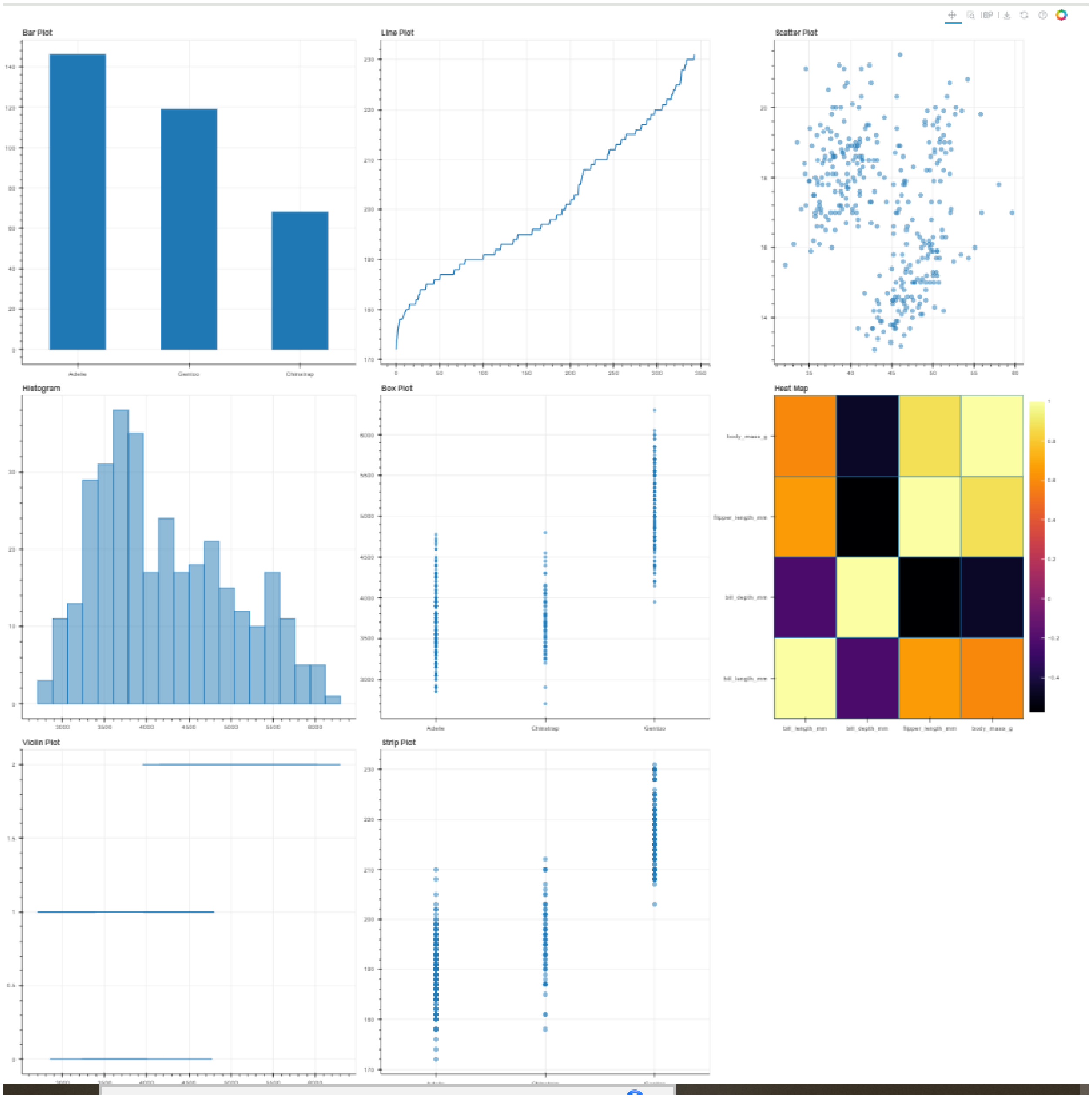
```
source = ColumnDataSource(df)
# Bar Plot
species_counts = df["species"].value_counts()
bar_source = ColumnDataSource(data=dict(species=species_counts.index,
counts=species_counts.values))
bar_plot = figure(x_range=FactorRange(*species_counts.index), title="Bar Plot")
bar_plot.vbar(x='species', top='counts', source=bar_source, width=0.5)
# Line Plot
line_plot = figure(title="Line Plot")
line_plot.line(df.index, df["flipper_length_mm"], line_width=2)
# Scatter Plot
scatter_plot = figure(title="Scatter Plot")
scatter_plot.scatter(df["bill_length_mm"], df["bill_depth_mm"], size=7, alpha=0.5)
# Histogram
hist, edges = np.histogram(df["body_mass_g"], bins=20)
hist_source = ColumnDataSource(data=dict(left=edges[:-1], right=edges[1:], top=hist))
hist_plot = figure(title="Histogram")
hist_plot.quad(top='top', bottom=0, left='left', right='right', source=hist_source, fill_alpha=0.5)
# Box Plot (using vbars)
box_plot = figure(x_range=FactorRange(*df["species"].unique()), title="Box Plot")
for species in df["species"].unique():
    subset = df[df["species"] == species]["body_mass_g"]
    q1, q2, q3 = np.percentile(subset, [25, 50, 75])
    iqr = q3 - q1
    lower = max(subset.min(), q1 - 1.5 * iqr)
    upper = min(subset.max(), q3 + 1.5 * iqr)
    box_plot.segment(species, lower, species, upper, line_width=2)
    box_plot.vbar(x=species, top=q3, bottom=q1, width=0.5)
    box_plot.scatter([species] * len(subset), subset, size=5, alpha=0.5)
```

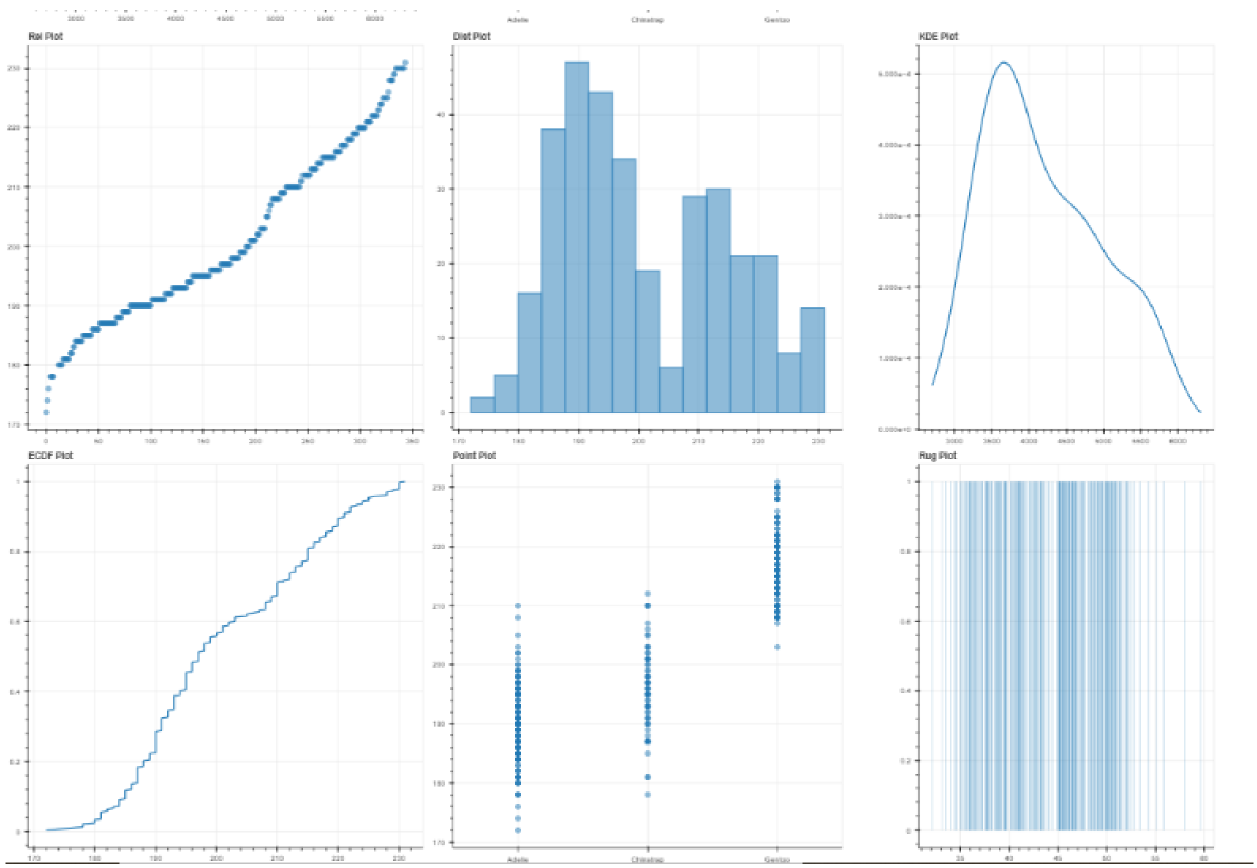
```python
# Heat Map (correlation matrix)
corr = df.select_dtypes(include=['number']).corr()
x_names, y_names, values = [], [], []
for i, col1 in enumerate(corr.columns):
    for j, col2 in enumerate(corr.columns):
        x_names.append(col1)
        y_names.append(col2)
        values.append(corr.iloc[i, j])
heat_source = ColumnDataSource(data=dict(x=x_names, y=y_names, value=values))
mapper = LinearColorMapper(palette="Inferno256", low=min(values), high=max(values))
heatmap = figure(title="Heat Map", x_range=list(corr.columns), y_range=list(corr.columns))
heatmap.rect(x="x", y="y", width=1, height=1, source=heat_source,
fill_color=linear_cmap("value", "Inferno256", min(values), max(values)))
color_bar = ColorBar(color_mapper=mapper)
heatmap.add_layout(color_bar, 'right')
# Violin Plot (using KDE)
violin_plot = figure(title="Violin Plot")
for i, species in enumerate(df["species"].unique()):
    subset = df[df["species"] == species]["body_mass_g"]
    kde = gaussian_kde(subset)
    x = np.linspace(subset.min(), subset.max(), 100)
    y = kde(x)
    violin_plot.line(x, y + i, line_width=2)
# Strip Plot
strip_plot = figure(title="Strip Plot", x_range=FactorRange(*df["species"].unique()))
strip_plot.scatter(x="species", y="flipper_length_mm", source=source, size=7, alpha=0.5)
# Rel Plot (approximate with scatter)
rel_plot = figure(title="Rel Plot")
rel_plot.scatter(df.index, df["flipper_length_mm"], size=7, alpha=0.5)
```

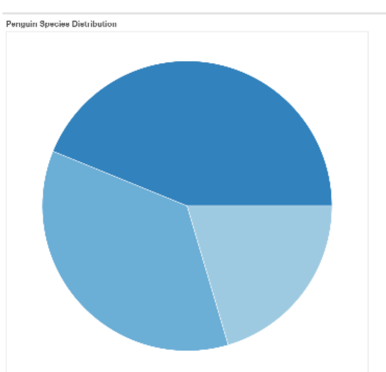
```
# Dist Plot (Histogram)
dist_hist, dist_edges = np.histogram(df["flipper_length_mm"], bins=15)
dist_source = ColumnDataSource(data=dict(left=dist_edges[:-1], right=dist_edges[1:],
top=dist_hist))
dist_plot = figure(title="Dist Plot")
dist_plot.quad(top='top', bottom=0, left='left', right='right', source=dist_source, fill_alpha=0.5)
# KDE Plot
kde_plot = figure(title="KDE Plot")
kde = gaussian_kde(df["body_mass_g"])
x = np.linspace(df["body_mass_g"].min(), df["body_mass_g"].max(), 100)
y = kde(x)
kde_plot.line(x, y, line_width=2)
# ECDF Plot
ecdf = ECDF(df["flipper_length_mm"])
ecdf_plot = figure(title="ECDF Plot")
ecdf_plot.line(ecdf.x, ecdf.y, line_width=2)
# Point Plot
point_plot = figure(title="Point Plot", x_range=FactorRange(*df["species"].unique()))
point_plot.scatter(x="species", y="flipper_length_mm", source=source, size=7, alpha=0.5)
# Rug Plot
rug_plot = figure(title="Rug Plot")
rug_plot.segment(df["bill_length_mm"], 0, df["bill_length_mm"], 1, alpha=0.3)
# Arrange all plots in a grid
grid = gridplot([
    [bar_plot, line_plot, scatter_plot],
    [hist_plot, box_plot, heatmap],
    [ violin_plot, strip_plot],
    [rel_plot, dist_plot, kde_plot],
    [ecdf_plot, point_plot, rug_plot]
])
```

show(grid)

OUTPUT :







4.Plotly

import pandas as pd

import numpy as np

import plotly.graph_objects as go

import plotly.express as px

from scipy.stats import gaussian_kde

from statsmodels.distributions.empirical_distribution import ECDF df = pd.read_csv("penguins_dataset.csv")

df.dropna(inplace=True)

 # Bar Plot

bar_plot = px.bar(df, x="species", title="Bar Plot")

 # Line Plot

line_plot = px.line(df, x=df.index, y="flipper_length_mm", title="Line Plot")

 # Scatter Plot

scatter_plot = px.scatter(df, x="bill_length_mm", y="bill_depth_mm", title="Scatter Plot")

 # Histogram

hist_plot = px.histogram(df, x="body_mass_g", nbins=20, title="Histogram")

 # Box Plot

box_plot = px.box(df, x="species", y="body_mass_g", title="Box Plot")

 # Heat Map (correlation matrix)

corr = df.select_dtypes(include=['number']).corr()

heatmap = px.imshow(corr, text_auto=True, title="Heatmap")

 # Pie Chart

pie_plot = px.pie(df, names="species", title="Pie Chart")

 # Violin Plot

violin_plot = px.violin(df, x="species", y="body_mass_g", box=True, title="Violin Plot")

 # Strip Plot

strip_plot = px.strip(df, x="species", y="flipper_length_mm", title="Strip Plot")

 # Rel Plot

rel_plot = px.scatter(df, x=df.index, y="flipper_length_mm", title="Rel Plot")

 # Dist Plot

dist_plot = px.histogram(df, x="flipper_length_mm", nbins=15, title="Dist Plot")

 # KDE Plot

kde = gaussian_kde(df["body_mass_g"])

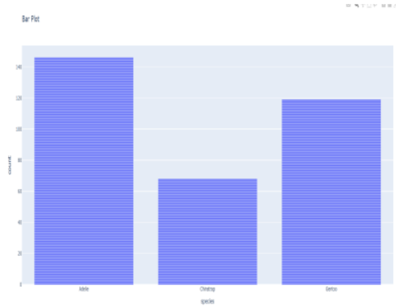
x = np.linspace(df["body_mass_g"].min(), df["body_mass_g"].max(), 100)

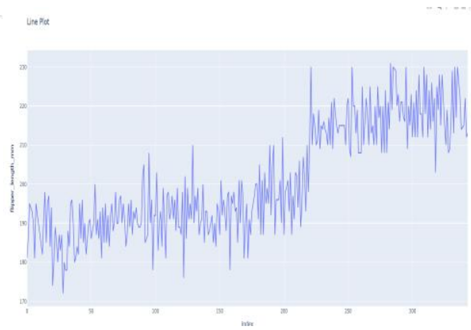
y = kde(x)

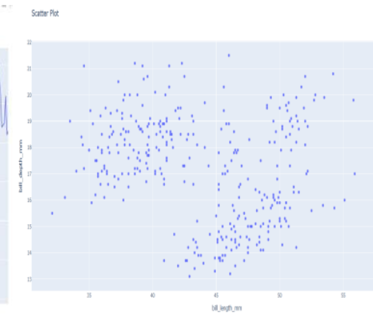
kde_plot = go.Figure()

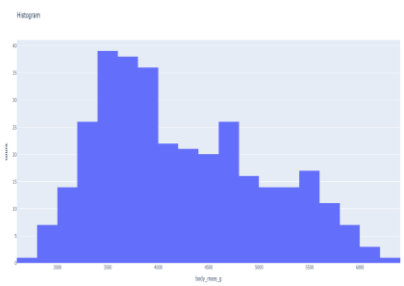
```
kde_plot.add_trace(go.Scatter(x=x, y=y, mode="lines", name="KDE Plot"))
kde_plot.update_layout(title="KDE Plot")
# ECDF Plot
ecdf = ECDF(df["flipper_length_mm"])
ecdf_plot = go.Figure()
ecdf_plot.add_trace(go.Scatter(x=ecdf.x, y=ecdf.y, mode="lines", name="ECDF Plot"))
ecdf_plot.update_layout(title="ECDF Plot")
# Point Plot
point_plot = px.scatter(df, x="species", y="flipper_length_mm", title="Point Plot")
# Rug Plot
rug_plot = go.Figure()
rug_plot.add_trace(go.Scatter(x=df["bill_length_mm"], y=np.zeros_like(df["bill_length_mm"]),
mode="markers", name="Rug Plot"))
rug_plot.update_layout(title="Rug Plot")
bar_plot.show()
line_plot.show()
scatter_plot.show()
hist_plot.show()
box_plot.show()
heatmap.show()
pie_plot.show()
violin_plot.show()
strip_plot.show()
rel_plot.show()
dist_plot.show()
kde_plot.show()
ecdf_plot.show()
point_plot.show()
rug_plot.show()
```

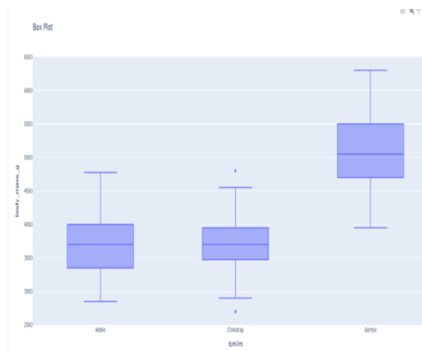
OUTPUT :

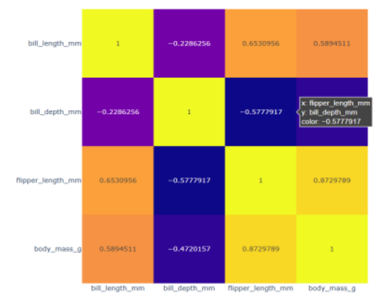

Bar Plot

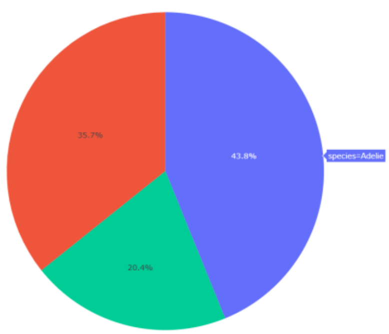

Line Plot

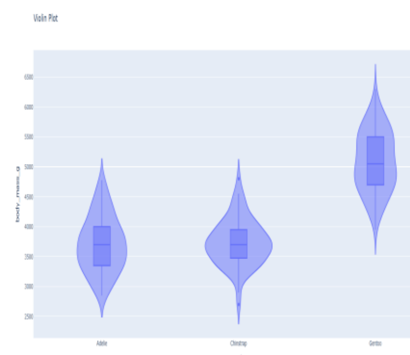

Scatter Plot

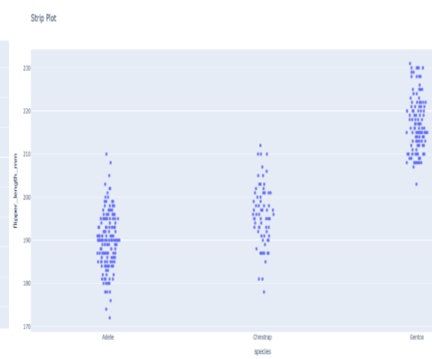

Histogram

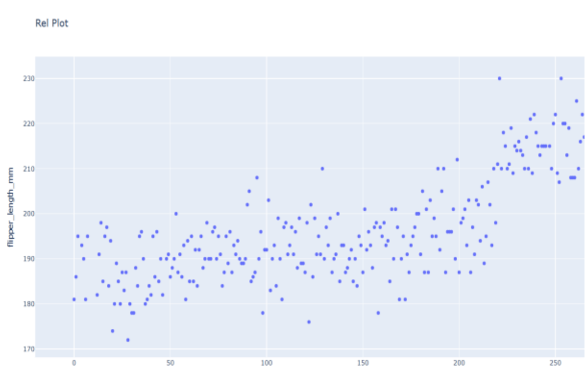

Box Plot

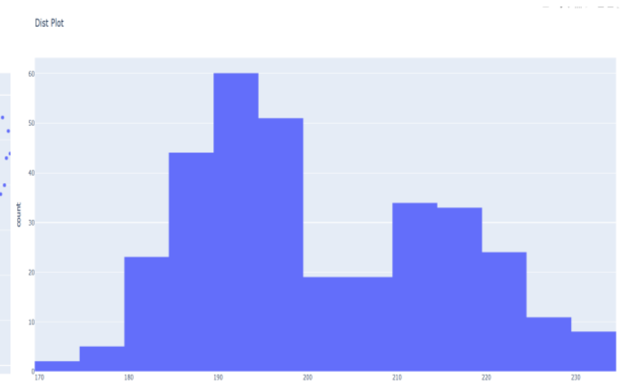

bill_length_mm
bill_depth_mm
flipper_length_mm
body_mass_g
1
−0.2286256
0.6530956
0.5894511
−0.2286256
1
−0.5777917
x: flipper_length_mm
y: bill_depth_mm
color: −0.5777917
0.6530956
−0.5777917
1
0.8729789
0.5894511
−0.4720157
0.8729789
1
bill_length_mm
bill_depth_mm
flipper_length_mm
body_mass_g

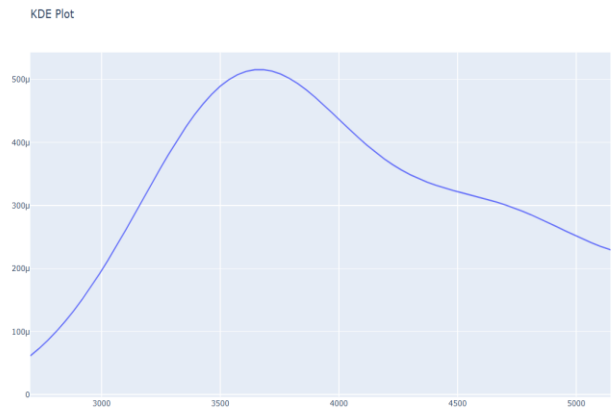

35.7%
43.8%
species=Adelie
20.4%

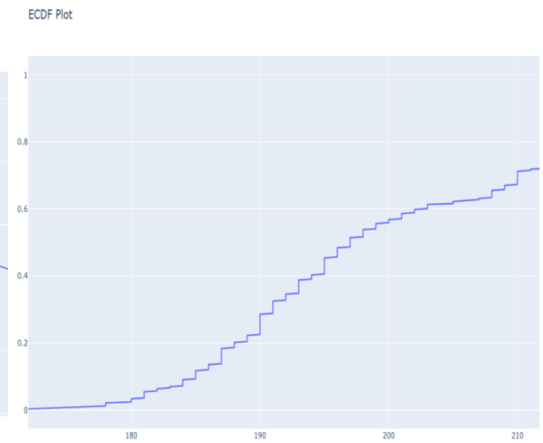

Violin Plot

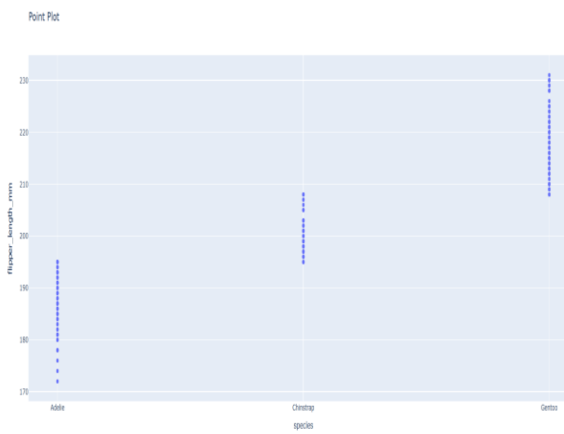

Strip Plot

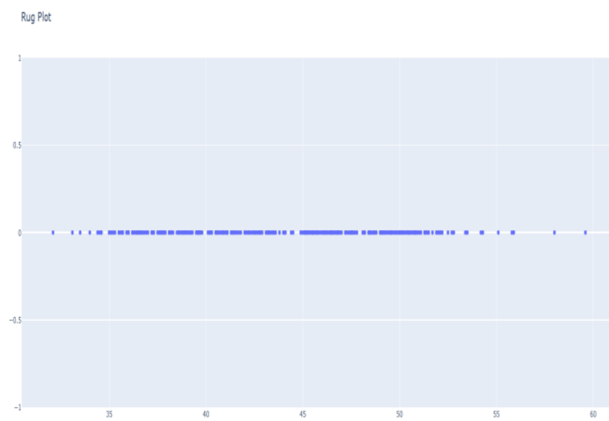

Rel Plot
flipper_length_mm
index


Dist Plot









RESULT :

Thus performing data visualization on a dataset using multiple plots like bar chart , pie chart, scatter plot etc. have been done successfully.

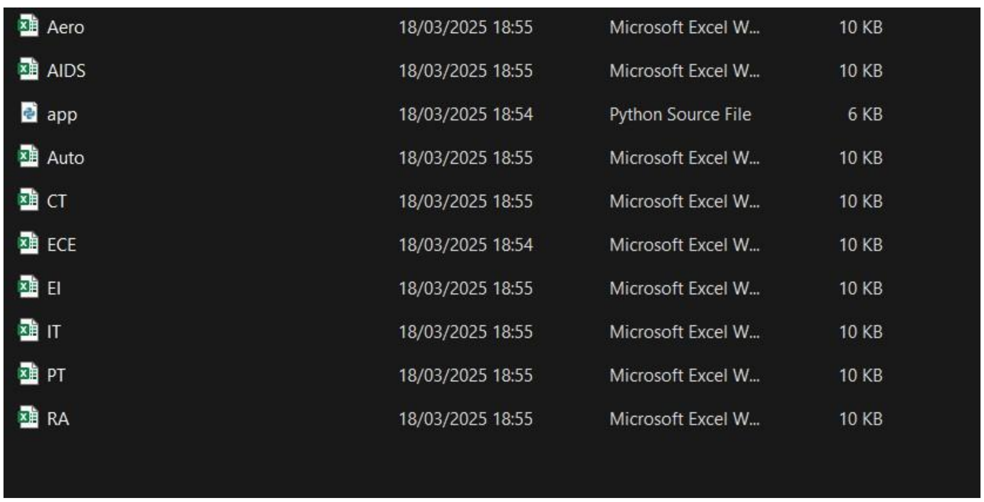
EXP NO : 5 DATE : 12/2/25	EXTRACTING AND VISUALISING FROM MULTIPLE DATA FILES

AIM :

To extract data from multiple datasets and construct an interactive webpage to know details of students enrolled in different Open elective subjects.

DATASETS :

List of OE subjects



Aero	18/03/2025 18:55	Microsoft Excel W...	10 KB
AIDS	18/03/2025 18:55	Microsoft Excel W...	10 KB
app	18/03/2025 18:54	Python Source File	6 KB
Auto	18/03/2025 18:55	Microsoft Excel W...	10 KB
CT	18/03/2025 18:55	Microsoft Excel W...	10 KB
ECE	18/03/2025 18:54	Microsoft Excel W...	10 KB
EI	18/03/2025 18:55	Microsoft Excel W...	10 KB
IT	18/03/2025 18:55	Microsoft Excel W...	10 KB
PT	18/03/2025 18:55	Microsoft Excel W...	10 KB
RA	18/03/2025 18:55	Microsoft Excel W...	10 KB

Each dept – total 50 students

Total 9 depts

we have Auto.xlsx , EI.xlsx , ECE.xlsx, IT.xlsx, RA.xlsx , PT.xlsx,Aero.xlsx,AIDS.xlsx,CT.xlsx

SOURCE CODE :

app.py :

```
import os
import pandas as pd
from flask import Flask, render_template, request
```

```
app = Flask(_name_)

# Path for the Excel files
department_files = {
    'EI': 'EI.xlsx',
    'IT': 'IT.xlsx',
    'AERO': 'Aero.xlsx',
    'CT': 'CT.xlsx',
    'RA': 'RA.xlsx',
    'AIDS': 'AIDS.xlsx',
    'ECE': 'ECE.xlsx',
    'PT': 'PT.xlsx',
    'AUTO': 'AUTO.xlsx',
}

# Function to load and process each department's data
def load_department_data(file_path):
    df = pd.read_excel(file_path)
    # Calculate total marks if empty
    df['Total Marks'] = df['Total Marks'].fillna(((df['Assess 1'] + df['Assess 2']) * 0.4) + df['End
Sem Marks'] * 0.6)
    return df

# Function to combine all department data into one DataFrame
def combine_all_data():
    combined_data = pd.DataFrame()

    # Collecting the performance statistics by department
    department_performance = {}
```

```
    for dept, file_path in department_files.items():
        dept_data = load_department_data(file_path)
        dept_data['Department'] = dept  # Add department info to each row
        combined_data = pd.concat([combined_data, dept_data], ignore_index=True)

        # Calculate pass percentage for department
        total_students = len(dept_data)
        passing_students = len(dept_data[dept_data['Total Marks'] >= 50])  # Assuming 50 as pass
mark
        pass_percentage = round((passing_students / total_students) * 100, 1)  # Round to one
decimal place
        department_performance[dept] = {
            'total_students': total_students,
            'passing_students': passing_students,
            'pass_percentage': pass_percentage
        }

    return combined_data, department_performance

# Function to filter students based on selected OE
def filter_students_by_oe(oe_selected):
    filtered_data = student_df[student_df['OE Opted'] == oe_selected]
    return filtered_data

# Function to calculate department-wise performance for selected OE
def department_performance_for_oe(oe_selected, students):
    department_performance_for_oe = {}

    # Grouping students by department
```

```
    grouped_by_dept = students.groupby('Department')

    for dept, group in grouped_by_dept:
        total_students = len(group)
        passing_students = len(group[group['Total Marks'] >= 50])  # Assuming 50 as pass mark
        pass_percentage = round((passing_students / total_students) * 100, 1)  # Round to one
decimal place

        department_performance_for_oe[dept] = {
            'total_students': total_students,
            'passing_students': passing_students,
            'pass_percentage': pass_percentage
        }

    return department_performance_for_oe

# Function to calculate highest, lowest, and average marks for the selected OE
def calculate_marks_statistics(oe_selected, students):
    highest_mark = students['Total Marks'].max()
    lowest_mark = students['Total Marks'].min()
    average_mark = round(students['Total Marks'].mean(), 1)

    return highest_mark, lowest_mark, average_mark

@app.route('/', methods=['GET', 'POST'])
def index():
    global student_df
    student_df, department_performance = combine_all_data()  # Load all data initially

    oe_selected = request.form.get('oe_opted', None)
```

```
    performance_stats = None
    students_list = []
    department_performance_for_selected_oe = None
    marks_statistics = None

    if oe_selected:
        # Filter students for the selected OE
        filtered_data = filter_students_by_oe(oe_selected)

        if not filtered_data.empty:
            # Calculate overall OE performance stats
            total_students = len(filtered_data)
            passing_students = len(filtered_data[filtered_data['Total Marks'] >= 50])  # Assuming 50
as pass mark
            pass_percentage = round((passing_students / total_students) * 100, 1)  # Round to one
decimal place

            performance_stats = {
                'oe_selected': oe_selected,
                'total_students': total_students,
                'passing_students': passing_students,
                'pass_percentage': pass_percentage
            }

            # Collect student details
            students_list = filtered_data[['Name', 'Register Number', 'Total Marks',
'Department']].to_dict(orient='records')

            # Calculate department performance for selected OE
            department_performance_for_selected_oe =
department_performance_for_oe(oe_selected, filtered_data)
```

```
            # Calculate marks statistics (highest, lowest, and average)
            highest_mark, lowest_mark, average_mark = calculate_marks_statistics(oe_selected,
filtered_data)
            marks_statistics = {
                'highest_mark': highest_mark,
                'lowest_mark': lowest_mark,
                'average_mark': average_mark
            }

    return render_template('index.html',
                           performance_stats=performance_stats,
                           students_list=students_list,

department_performance_for_selected_oe=department_performance_for_selected_oe,
                           department_performance=department_performance,
                           marks_statistics=marks_statistics)


if _name_ == "_main_":
    app.run(debug=True)
```

index.html :

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Student Performance and OE</title>
    <link rel="stylesheet" href="{{ url_for('static', filename='css/style.css') }}">
</head>
```

```html
<body>
    <div class="container">
        <h1>Student Performance on OE </h1>

        <!-- Form to select OE -->
        <form method="POST">
            <label for="oe_opted">Select an OE:</label>
            <select name="oe_opted" id="oe_opted" required>
                <option value="">--Select OE--</option>
                <option value="Microcontroller">Microcontroller</option>
                <option value="Data Structures">Data Structures</option>
                <option value="Biomimetic">Biomimetic</option>
                <option value="Product Design">Product Design</option>
                <option value="Principles of Flight">Principles of Flight</option>
                <option value="Engines">Engines</option>
                <option value="Polymer Properties">Polymer Properties</option>
                <option value="Instrumentation">Instrumentation</option>
                <option value="Communication">Communication</option>
            </select>
            <button type="submit">Show Performance</button>
        </form>

        <div class="main-content">
            <!-- Side Table for Department-wise Pass Percentage -->
            <div class="side-table">
                <h2>Department-wise Pass Percentage for {{ performance_stats['oe_selected']
}}</h2>
                <table>
                    <thead>
                        <tr>
```

```
                <th>Department</th>
                <th>Total Students</th>
                <th>Passing Students</th>
                <th>Pass Percentage</th>
              </tr>
            </thead>
            <tbody>
              {% if department_performance_for_selected_oe %}
                {% for dept, stats in department_performance_for_selected_oe.items() %}
                  <tr>
                    <td>{{ dept }}</td>
                    <td>{{ stats['total_students'] }}</td>
                    <td>{{ stats['passing_students'] }}</td>
                    <td>{{ stats['pass_percentage'] }}%</td>
                  </tr>
                {% endfor %}
              {% else %}
                <tr>
                  <td colspan="4">No students found for the selected OE.</td>
                </tr>
              {% endif %}
            </tbody>
          </table>
        </div>

        <!-- Main Table for OE Performance -->
        <div class="performance-table">
          {% if performance_stats %}
            <h2>Performance for {{ performance_stats['oe_selected'] }}</h2>
            <p>Total Students: {{ performance_stats['total_students'] }}</p>
```

```
<p>Passing Students: {{ performance_stats['passing_students'] }}</p>
<p>Pass Percentage: {{ performance_stats['pass_percentage'] }}%</p>

<!-- Display Highest, Lowest, and Average Marks -->
<div class="marks-stats">
  <p><strong>Highest Mark:</strong> {{ marks_statistics['highest_mark'] }}</p>
  <p><strong>Lowest Mark:</strong> {{ marks_statistics['lowest_mark'] }}</p>
  <p><strong>Average Mark:</strong> {{ marks_statistics['average_mark'] }}</p>
</div>

<h3>Student List</h3>
<table>
  <thead>
    <tr>
      <th>Department</th>
      <th>Name</th>
      <th>Register Number</th>
      <th>Total Marks</th>
    </tr>
  </thead>
  <tbody>
    {% if students_list %}
      {% for student in students_list %}
        <tr>
          <td>{{ student['Department'] }}</td>
          <td>{{ student['Name'] }}</td>
          <td>{{ student['Register Number'] }}</td>
          <td>{{ student['Total Marks'] }}</td>
        </tr>
      {% endfor %}
```

```
                {% else %}
                    <tr>
                        <td colspan="4">No students found for this OE.</td>
                    </tr>
                {% endif %}
            </tbody>
        </table>
    {% else %}
        <p>Please select an OE to see performance details.</p>
    {% endif %}
  </div>
 </div>
</div>
</body>
</html>
```

OUTPUT :

```
PS F:\sem6\DS\dslab\multixlui> python app.py
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 247-135-448
```

Student Performance on OE

Select an OE: --Select OE-- Show Performance

Department-wise Pass Percentage for

Department	Total Students	Passing Students	Pass Percentage
No students found for the selected OE.			

Please select an OE to see performance details.

Student Performance on OE

Select an OE: --Select OE-- Show Performance

Department-wise Pass Percentage for Data Structures

Department	Total Students	Passing Students	Pass Percentage
AERO	3	3	100.0%
AIDS	3	3	100.0%
AUTO	3	3	100.0%
ECE	3	3	100.0%
EI	3	3	100.0%
IT	3	3	100.0%
RA	3	3	100.0%

Performance for Data Structures

Total Students: 21

Passing Students: 21

Passing Students: 21

Pass Percentage: 100.0%

Highest Mark: 95.2

Lowest Mark: 61.400000000000006

Average Mark: 78.7

Student List

Department	Name	Register Number	Total Marks
EI	Emily Davis	2022503002	76.6
EI	Benjamin Garcia	2022503012	81.6
EI	James Smith	2022503019	72.4
IT	John Harris	2022506001	61.400000000000006
IT	Benjamin Lee	2022506012	81.6
IT	Benjamin Thomas	2022506018	76.0
AERO	Sarah Lee	2022501007	80.8
AERO	Isabella Perez	2022501011	73.6
AERO	Amelia Martin	2022501020	74.80000000000001
RA	Rishi Patel	2022511001	76.0
RA	Arvind Joshi	2022511010	89.4

RESULT :

Thus extracting data from multiple datasets and constructing an interactive webpage to know details of students enrolled in different Open elective subjecs have been implemented successfully.

EXP NO : 6 DATE : 19/2/25	DATA CLEANUP AND PREPROCESSING

AIM :

To perform data cleaning and preprocessing on a dataset.

SOURCE CODE :

1.Count the missing values and no of samples per label

```
import pandas as pd
df = pd.read_csv('iris_with_headers.csv')
missing_values = df.isnull().sum()
print("Missing Values:\n", missing_values)
zero_values = (df == 0).sum()
print("\nAttributes with Zeros:\n", zero_values)
duplicates = df.duplicated().sum()
print(f"\nNumber of Duplicated Rows: {duplicates}")
label_counts = df['species'].value_counts()
print("\nNumber of Samples per Label:\n", label_counts)
```

OUTPUT :

```
Missing Values:
 sepal_length    0
sepal_width     0
petal_length    0
petal_width     0
species         0
dtype: int64

Attributes with Zeros:
 sepal_length    0
sepal_width     0
petal_length    0
petal_width     0
species         0
dtype: int64

Number of Duplicated Rows: 3

Number of Samples per Label:
 species
Iris-setosa        50
Iris-versicolor    50
Iris-virginica     50
Name: count, dtype: int64
```

2. Check for duplicates and print them

```
import pandas as pd
# Load the dataset
df = pd.read_csv('iris_with_headers.csv')
duplicates = df[df.duplicated()]
if not duplicates.empty:
    print("\nDuplicate Rows in the Dataset:")
    print(duplicates)
else:
    print("\nNo duplicate rows found.")
```

OUTPUT :

```
Duplicate Rows in the Dataset:
     sepal_length  sepal_width  petal_length  petal_width         species
34            4.9          3.1           1.5          0.1     Iris-setosa
37            4.9          3.1           1.5          0.1     Iris-setosa
142           5.8          2.7           5.1          1.9  Iris-virginica
```

3.Delete duplicate rows from the dataset

```
import pandas as pd
# Load the dataset
df = pd.read_csv('iris_with_headers.csv')
duplicates = df[df.duplicated()]
num_duplicates = len(duplicates)
print(f"\nNumber of duplicate rows in the dataset: {num_duplicates}")
if not duplicates.empty:
    print("\nDuplicate Rows:")
    print(duplicates)
df_cleaned = df.drop_duplicates()
df_cleaned.to_csv('iris_cleaned.csv', index=False)
print("\nDuplicate rows removed. Cleaned dataset saved as 'iris_cleaned.csv'.")
```

OUTPUT :

```
Number of duplicate rows in the dataset: 3

Duplicate Rows:
     sepal_length  sepal_width  petal_length  petal_width          species
34            4.9          3.1           1.5          0.1      Iris-setosa
37            4.9          3.1           1.5          0.1      Iris-setosa
142           5.8          2.7           5.1          1.9   Iris-virginica

Duplicate rows removed. Cleaned dataset saved as 'iris_cleaned.csv'.
```

4. Check for samples with 4 or more missing values and remove them

import pandas as pd

df = pd.read_csv('iris_with_headers.csv')

num_duplicates = df.duplicated().sum()

print(f"\nNumber of duplicate rows in the dataset: {num_duplicates}")

df_cleaned = df.drop_duplicates()

missing_threshold = 4 # Define threshold for missing values

num_missing_samples = (df_cleaned.isna().sum(axis=1) >= missing_threshold).sum()

print(f"\nNumber of samples with {missing_threshold} or more missing values: {num_missing_samples}")

df_cleaned = df_cleaned.dropna(thresh=df_cleaned.shape[1] - missing_threshold)

df_cleaned.to_csv('iris_cleaned.csv', index=False)

print("\nData cleaning complete. Cleaned dataset saved as 'iris_cleaned.csv'.")

OUTPUT :

```
Number of duplicate rows in the dataset: 3

Number of samples with 4 or more missing values: 0

Data cleaning complete. Cleaned dataset saved as 'iris_cleaned.csv'.
```

RESULT :

Thus performiing data cleaning and preprocessing on a dataset have been done successfully.

EXP NO : 7 DATE : 26/2/25	DIMENSIONALITY REDUCTION USING PCA

AIM :

To perform dimensionality reduction on a dataset using PCA technique.

SOURCE CODE :

```
import numpy as np
import pandas as pd
from sklearn import preprocessing
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression
from sklearn.metrics import accuracy_score
df = pd.read_csv('iris_with_headers.csv')
label_encoder = preprocessing.LabelEncoder()
label = label_encoder.fit_transform(df['species'])
df = df.select_dtypes(include=[np.number])
data = np.array(df)

# Calculate mean using for loop
num_samples = len(data)
num_features = len(data[0])
mean = [0] * num_features
for j in range(num_features):
    feature_sum = 0
    for i in range(num_samples):
        feature_sum += data[i][j]
```

```
    mean[j] = feature_sum / num_samples

# Centering the data
centered_data = []
for i in range(num_samples):
    centered_sample = []
    for j in range(num_features):
        centered_sample.append(data[i][j] - mean[j])
    centered_data.append(centered_sample)

# Calculate covariance matrix manually
cov_matrix = [[0] * num_features for _ in range(num_features)]
n = len(data) - 1  # Using n-1 for unbiased estimation
for i in range(num_features):
    for j in range(num_features):
        covariance_sum = 0
        for k in range(len(data)):
            covariance_sum += centered_data[k][i] * centered_data[k][j]
        cov_matrix[i][j] = covariance_sum / n

# Compute eigenvalues and eigenvectors
eigenvalues, eigenvectors = np.linalg.eig(np.array(cov_matrix))

# Sort eigenvalues and eigenvectors in descending order
eigen_pairs = sorted(zip(eigenvalues, eigenvectors.T), key=lambda x: x[0], reverse=True)
eigenvalues_sorted, eigenvectors_sorted = zip(*eigen_pairs)
eigenvectors_sorted = [list(vec) for vec in eigenvectors_sorted]

# Selecting top k principal components
num_components = 3
```

```
projection_matrix = [list(col) for col in zip(*eigenvectors_sorted[:num_components])]

# Transforming data
transformed_data = []
for i in range(num_samples):
    transformed_sample = []
    for k in range(num_components):
        transformed_value = 0
        for j in range(num_features):
            transformed_value += centered_data[i][j] * projection_matrix[j][k]
        transformed_sample.append(transformed_value)
    transformed_data.append(transformed_sample)

print("Transformed Data:")
for row in transformed_data:
    print(row)

# Classification on PCA-transformed data
X = transformed_data  # Features
y = label  # Labels (0, 1, 2)
# Split data into training and testing sets (80% train, 20% test)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Standardize features (recommended for SVM)
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# 1. Linear Regression (not meant for classification but included for demonstration)
lin_reg = LinearRegression()
```

```
lin_reg.fit(X_train, y_train)
y_pred_lin = np.round(lin_reg.predict(X_test))  # Rounding to get class labels
print("Linear Regression Accuracy:", accuracy_score(y_test, y_pred_lin))

# Compare PCA-transformed vs. original data in classification
X_orig = df.values  # Original dataset before PCA
X_train_orig, X_test_orig, y_train_orig, y_test_orig = train_test_split(X_orig, y, test_size=0.2, random_state=42)
X_train_orig = scaler.fit_transform(X_train_orig)
X_test_orig = scaler.transform(X_test_orig)

lin_reg_orig = LinearRegression()
lin_reg_orig.fit(X_train_orig, y_train_orig)
y_pred_lin_orig = np.round(lin_reg_orig.predict(X_test_orig))
print("Linear Regression Accuracy (Original Data):", accuracy_score(y_test_orig, y_pred_lin_orig))
```

OUTPUT :

```
Transformed Data:
[np.float64(-2.6842071251039514), np.float64(-0.32660731476438687), np.float64(-0.021511837001962908)]
[np.float64(-2.715390615634133б), np.float64(0.16955684755602735), np.float64(-0.20352142500549064)]
[np.float64(-2.8898195396179194), np.float64(0.1373456096050289), np.float64(0.024709240998956883)]
[np.float64(-2.7464371973087376), np.float64(0.3111243157519928), np.float64(0.03767197528530075)]
[np.float64(-2.7285929818313175), np.float64(-0.33392456356845374), np.float64(0.0962296997746086)]
[np.float64(-2.2798973610096), np.float64(-0.7477827132251326), np.float64(0.17432561901640226)]
```

```
Linear Regression Accuracy: 1.0
Linear Regression Accuracy (Original Data): 1.0
```

RESULT :

Thus dimensionality reduction using PCA has been performed successfully.

EXP NO : 8 DATE : 5/3/25	DIMENSIONALITY REDUCTION USING LDA

AIM :

To perform dimensionality reduction on a dataset using LDA technique.

SOURCE CODE :

App.py:

```
import pandas as pd
from sklearn.decomposition import PCA
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis as LDA
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
import numpy as np
from flask import Flask, request, render_template
import joblib

app = Flask(_name_)

# Load dataset
file_path = "F:\\sem6\\DS\\dslab\\weather_forecast_data.csv"
df = pd.read_csv(file_path)

# Extract numerical features and target variable
```

```
X = df.drop(columns=["Rain"])  # Features (numerical only)
y = df["Rain"]  # Target variable

# Encode categorical labels (rain -> 1, no rain -> 0)
label_encoder = LabelEncoder()
y_encoded = label_encoder.fit_transform(y)

# Standardize the numerical data
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Apply LDA (better suited for classification tasks)
lda = LDA(n_components=1)
X_lda = lda.fit_transform(X_scaled, y_encoded)

# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X_lda, y_encoded, test_size=0.2,
random_state=42)

# Train a classifier (Random Forest)
clf = RandomForestClassifier(n_estimators=100, random_state=42)
clf.fit(X_train, y_train)

# Save the model and transformers
joblib.dump(clf, 'rain_model.pkl')
joblib.dump(scaler, 'scaler.pkl')
joblib.dump(lda, 'lda.pkl')
```

```
joblib.dump(label_encoder, 'label_encoder.pkl')
joblib.dump(X.columns.tolist(), 'feature_names.pkl')

# Predict route
@app.route('/', methods=['GET', 'POST'])
def index():
    prediction = None
    if request.method == 'POST':
        try:
            # Load models and transformers
            clf = joblib.load('rain_model.pkl')
            scaler = joblib.load('scaler.pkl')
            lda = joblib.load('lda.pkl')
            label_encoder = joblib.load('label_encoder.pkl')
            feature_names = joblib.load('feature_names.pkl')

            # Get input from form
            input_data = [
                float(request.form['Temperature']),
                float(request.form['Humidity']),
                float(request.form['WindSpeed']),
                float(request.form['CloudCover']),
                float(request.form['Pressure'])
            ]

            # Create DataFrame
            input_df = pd.DataFrame([input_data], columns=feature_names)
```

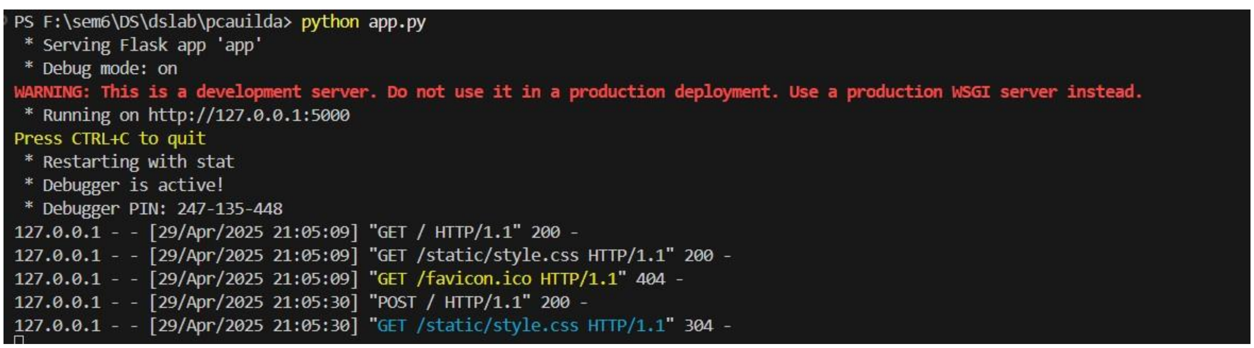
```
            input_scaled = scaler.transform(input_df)
            input_lda = lda.transform(input_scaled)
            prediction_label = clf.predict(input_lda)
            prediction = label_encoder.inverse_transform(prediction_label)[0]
        except Exception as e:
            prediction = f"Error: {e}"

    return render_template('index.html', prediction=prediction)


if _name_ == '_main_':
    app.run(debug=True)
```

index.html:

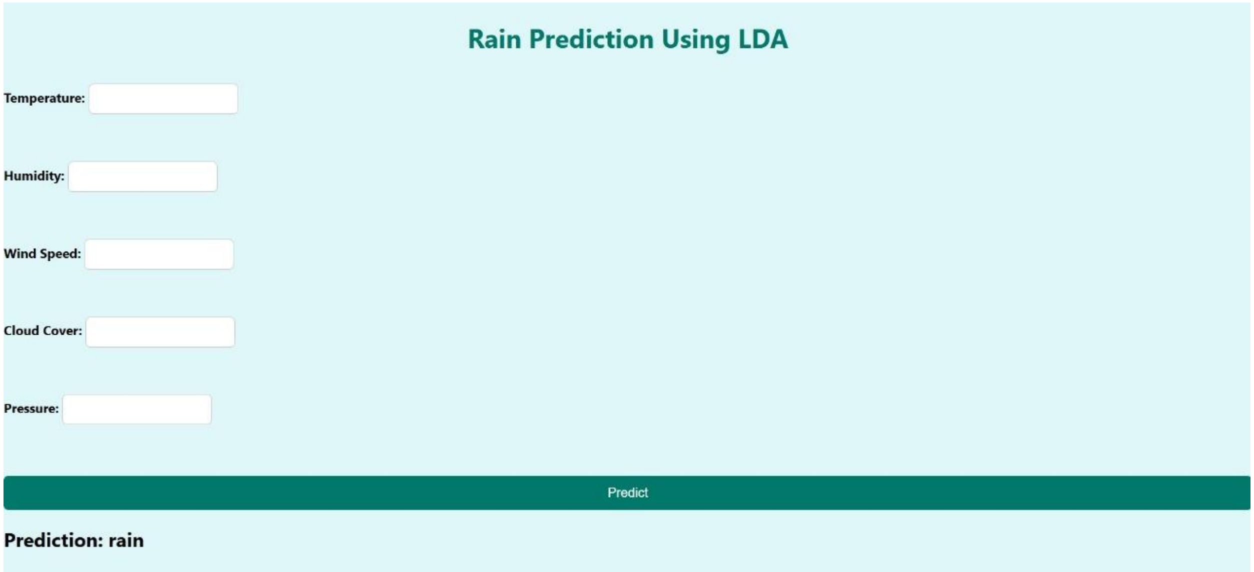
```
<!-- templates/index.html -->
<!DOCTYPE html>
<html>
<head>
    <title>Rain Prediction</title>
    <link rel="stylesheet" href="{{ url_for('static', filename='style.css') }}">
</head>
<body>
    <h1>Rain Prediction Using LDA</h1>
    <form method="post">
        <label>Temperature: <input type="number" step="any" name="Temperature"
required></label><br><br>
        <label>Humidity: <input type="number" step="any" name="Humidity"
required></label><br><br>
```

```
        <label>Wind Speed: <input type="number" step="any" name="WindSpeed" required></label><br><br>

        <label>Cloud Cover: <input type="number" step="any" name="CloudCover" required></label><br><br>

        <label>Pressure: <input type="number" step="any" name="Pressure" required></label><br><br>

        <button type="submit">Predict</button>

    </form>


    {% if prediction %}

        <h2>Prediction: {{ prediction }}</h2>

    {% endif %}

</body>

</html>
```

OUTPUT :



```
PS F:\sem6\DS\dslab\pcauilda> python app.py
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 247-135-448
127.0.0.1 - - [29/Apr/2025 21:05:09] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [29/Apr/2025 21:05:09] "GET /static/style.css HTTP/1.1" 200 -
127.0.0.1 - - [29/Apr/2025 21:05:09] "GET /favicon.ico HTTP/1.1" 404 -
127.0.0.1 - - [29/Apr/2025 21:05:30] "POST / HTTP/1.1" 200 -
127.0.0.1 - - [29/Apr/2025 21:05:30] "GET /static/style.css HTTP/1.1" 304 -
```



RESULT :

Thus dimensionality reduction using LDA has been performed successfully.

EXP NO : 9 DATE : 12/3/25	IMPLEMENTATION OF REGRESSION AND CLASSIFICATION ALGORITHMS

AIM :

To implement linear regression and various classifiers like naïve bayes,SVM.

SOURCE CODE :

Pca.py

```
import numpy as np
import pandas as pd
from sklearn import preprocessing
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression, LogisticRegression
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, classification_report, ConfusionMatrixDisplay

df=pd.read_csv('iris_with_headers.csv')
label_encoder = preprocessing.LabelEncoder()
label=label_encoder.fit_transform(df['species'])
df=df.select_dtypes(include=[np.number])
data=np.array(df)

# Calculate mean using for loop
num_samples = len(data)
num_features = len(data[0])
mean = [0] * num_features
```

```
for j in range(num_features):
    feature_sum = 0
    for i in range(num_samples):
        feature_sum += data[i][j]
    mean[j] = feature_sum / num_samples
# Centering the data
centered_data = []
for i in range(num_samples):
    centered_sample = []
    for j in range(num_features):
        centered_sample.append(data[i][j] - mean[j])
    centered_data.append(centered_sample)

# Calculate covariance matrix manually
cov_matrix = [[0] * num_features for _ in range(num_features)]
n = len(data) - 1  # Using n-1 for unbiased estimation

for i in range(num_features):
    for j in range(num_features):
        covariance_sum = 0
        for k in range(len(data)):
            covariance_sum += centered_data[k][i] * centered_data[k][j]
        cov_matrix[i][j] = covariance_sum / n

# Compute eigenvalues and eigenvectors
eigenvalues, eigenvectors = np.linalg.eig(np.array(cov_matrix))

# Sort eigenvalues and eigenvectors in descending order
eigen_pairs = sorted(zip(eigenvalues, eigenvectors.T), key=lambda x: x[0], reverse=True)
eigenvalues_sorted, eigenvectors_sorted = zip(*eigen_pairs)
```

```
eigenvectors_sorted = [list(vec) for vec in eigenvectors_sorted]

# Selecting top k principal components
num_components = 3
projection_matrix = [list(col) for col in zip(*eigenvectors_sorted[:num_components])]

# Transforming data
transformed_data = []
for i in range(num_samples):
    transformed_sample = []
    for k in range(num_components):
        transformed_value = 0
        for j in range(num_features):
            transformed_value += centered_data[i][j] * projection_matrix[j][k]
        transformed_sample.append(transformed_value)
    transformed_data.append(transformed_sample)

print("Transformed Data:")
for row in transformed_data:
    print(row)

X = transformed_data # Features
y = label # Labels (0, 1, 2)
# Split data into training and testing sets (80% train, 20% test)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Standardize features (recommended for SVM)
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

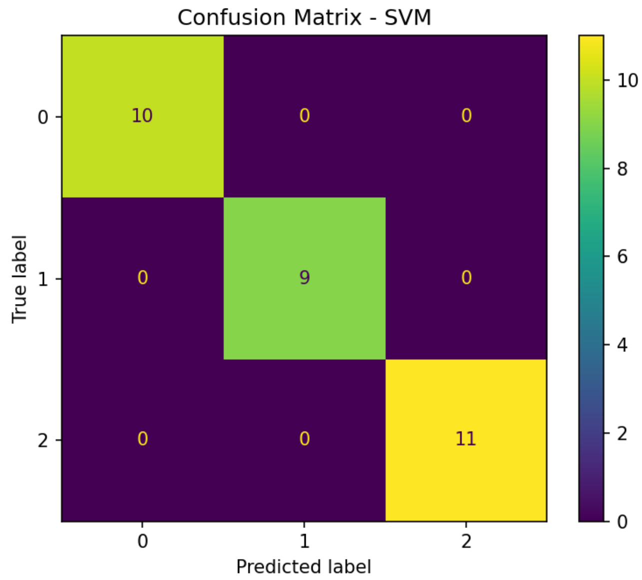
```
# 1. Linear Regression (not meant for classification but included for demonstration)
lin_reg = LinearRegression()
lin_reg.fit(X_train, y_train)
y_pred_lin = np.round(lin_reg.predict(X_test))  # Rounding to get class labels
print("Linear Regression Accuracy:", accuracy_score(y_test, y_pred_lin))

# 2. Logistic Regression
log_reg = LogisticRegression()
log_reg.fit(X_train, y_train)
y_pred_log = log_reg.predict(X_test)
print("\nLogistic Regression Accuracy:", accuracy_score(y_test, y_pred_log))

# 3. Naïve Bayes (Gaussian)
nb = GaussianNB()
nb.fit(X_train, y_train)
y_pred_nb = nb.predict(X_test)
print("\nNaïve Bayes Accuracy:", accuracy_score(y_test, y_pred_nb))

# 4. Support Vector Machine (SVM)
svm = SVC(kernel='linear')  # Using a linear kernel
svm.fit(X_train, y_train)
y_pred_svm = svm.predict(X_test)
print("\nSVM Accuracy:", accuracy_score(y_test, y_pred_svm))

# Print classification reports
print("\nClassification Report (SVM):\n", classification_report(y_test, y_pred_svm))

# Confusion Matrix
ConfusionMatrixDisplay.from_estimator(svm, X_test, y_test)
```

```
plt.title("Confusion Matrix - SVM")
plt.show()
```

OUTPUT :

```
Transformed Data:
[np.float64(-2.6842071251039514), np.float64(-0.32660731476438687), np.float64(-0.021511837001962908)]
[np.float64(-2.7153906156341336), np.float64(0.16955684755602735), np.float64(-0.20352142500549064)]
[np.float64(-2.8898195396179194), np.float64(0.1373456096050289), np.float64(0.024709240998956883)]
[np.float64(-2.7464371973087376), np.float64(0.3111243157519928), np.float64(0.03767197528530075)]
[np.float64(-2.7285929818313175), np.float64(-0.33392456356845374), np.float64(0.0962296997746086)]
[np.float64(-2.2798973610096), np.float64(-0.7477827132251326), np.float64(0.17432561901640226)]
[np.float64(-2.8208906821806328), np.float64(0.08210451102468182), np.float64(0.2642510851906958)]
```

```
Linear Regression Accuracy: 1.0

Logistic Regression Accuracy: 1.0


Naïve Bayes Accuracy: 0.9666666666666667

SVM Accuracy: 1.0

Classification Report (SVM):
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        10
           1       1.00      1.00      1.00         9
           2       1.00      1.00      1.00        11

    accuracy                           1.00        30
   macro avg       1.00      1.00      1.00        30
weighted avg       1.00      1.00      1.00        30
```



RESULT :

Thus implementation of regression and various classifiers have been done successfully.

EXP NO : 10 DATE : 19/3/25	MODEL TESTING AND PREDICTION WITH A USER INTERFACE

AIM :

To perform model testing and prediction with a user interface on a dataset.

SOURCE CODE :

app.py :

```
import pandas as pd
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.model_selection import StratifiedKFold
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
import numpy as np
import joblib
from flask import Flask, request, render_template

app = Flask(_name_)

# Load dataset
file_path = "F:\\sem6\\DS\\dslab\\weather_forecast_data.csv"
df = pd.read_csv(file_path)

# Extract numerical features and target variable
X = df.drop(columns=["Rain"])
y = df["Rain"]
```

```
# Encode target labels
label_encoder = LabelEncoder()
y_encoded = label_encoder.fit_transform(y)

# Standardize features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Apply PCA
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X_scaled)

# Save PCA transformed data (first 5 samples) for frontend use
pca_sample = pd.DataFrame(X_pca[:5], columns=['PC1', 'PC2'])
pca_sample.to_csv('pca_sample_display.csv', index=False)

# Print PCA-transformed values (first 5 for display)
print("\nPCA Transformed Sample (first 5 rows):")
for i in range(5):
    print(f"Sample {i+1}: PC1 = {X_pca[i,0]:.4f}, PC2 = {X_pca[i,1]:.4f}")

# K-Fold Cross Validation
k = 5
skf = StratifiedKFold(n_splits=k, shuffle=True, random_state=42)
accuracies = []

for fold, (train_index, test_index) in enumerate(skf.split(X_pca, y_encoded)):
```

```
    X_train, X_test = X_pca[train_index], X_pca[test_index]
    y_train, y_test = y_encoded[train_index], y_encoded[test_index]

    clf = RandomForestClassifier(n_estimators=100, random_state=42)
    clf.fit(X_train, y_train)
    y_pred = clf.predict(X_test)

    acc = accuracy_score(y_test, y_pred)
    accuracies.append(acc)

print(f"\nAverage Accuracy over {k} folds: {np.mean(accuracies)*100:.2f}%")

# Save models and transformers for Flask
joblib.dump(clf, 'rain_model.pkl')
joblib.dump(scaler, 'scaler.pkl')
joblib.dump(pca, 'pca.pkl')
joblib.dump(label_encoder, 'label_encoder.pkl')
joblib.dump(X.columns.tolist(), 'feature_names.pkl')

# Flask route for prediction
@app.route('/', methods=['GET', 'POST'])
def index():
    prediction = None
    pca_data = None
    if request.method == 'POST':
        try:
            clf = joblib.load('rain_model.pkl')
```

```
        scaler = joblib.load('scaler.pkl')
        pca = joblib.load('pca.pkl')
        label_encoder = joblib.load('label_encoder.pkl')
        feature_names = joblib.load('feature_names.pkl')

        input_data = [
            float(request.form['Temperature']),
            float(request.form['Humidity']),
            float(request.form['WindSpeed']),
            float(request.form['CloudCover']),
            float(request.form['Pressure'])
        ]

        input_df = pd.DataFrame([input_data], columns=feature_names)
        input_scaled = scaler.transform(input_df)
        input_pca = pca.transform(input_scaled)
        prediction_label = clf.predict(input_pca)
        prediction = label_encoder.inverse_transform(prediction_label)[0]

        pca_data = {'PC1': round(input_pca[0][0], 4), 'PC2': round(input_pca[0][1], 4)}

    except Exception as e:
        prediction = f"Error: {e}"

    return render_template('index.html', prediction=prediction, pca_data=pca_data)

if _name_ == '_main_':
```

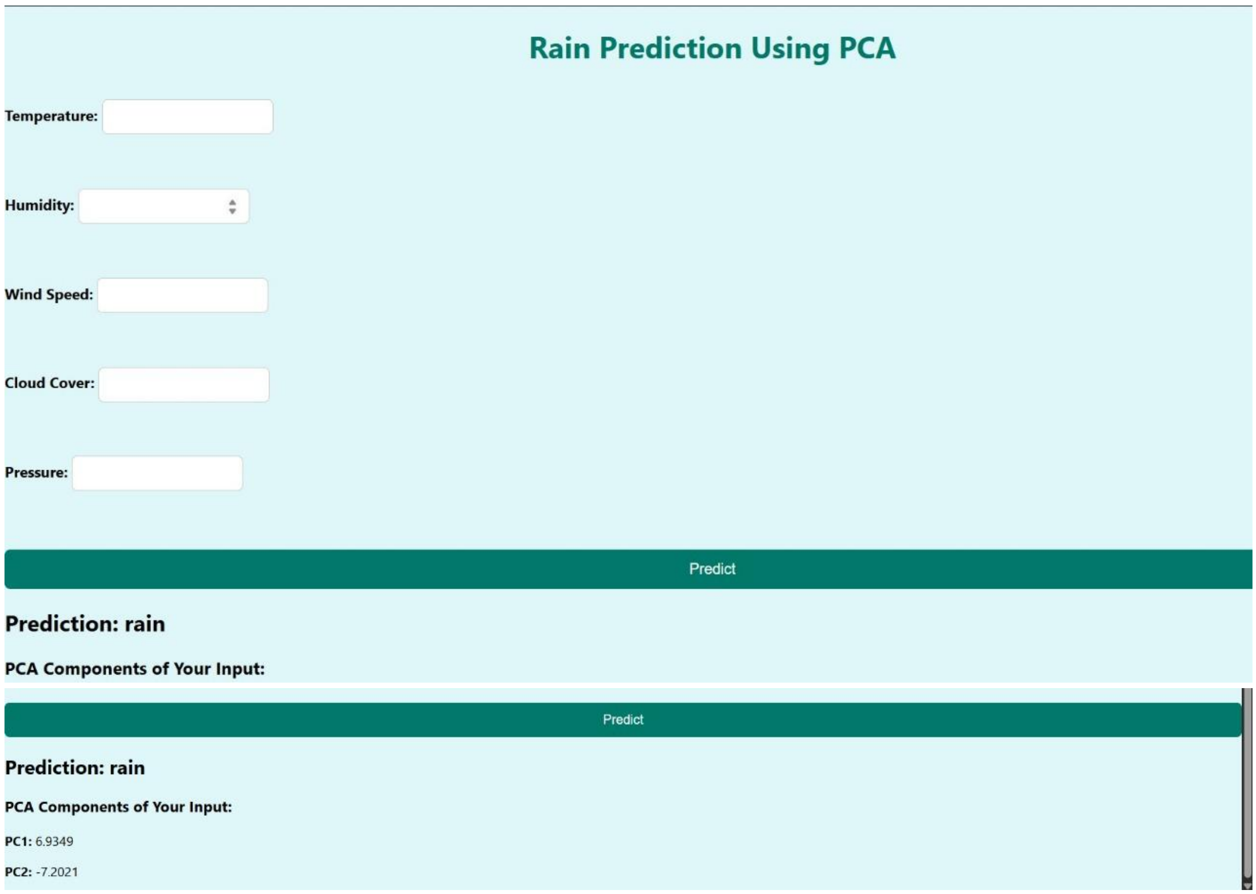
```python
    app.run(debug=True)
```

templates/index.html :

```html
<!DOCTYPE html>
<html>
<head>
    <title>Rain Prediction</title>
    <link rel="stylesheet" href="{{ url_for('static', filename='style.css') }}">
</head>
<body>
    <h1>Rain Prediction Using PCA</h1>

    <form method="post">
        <label>Temperature: <input type="number" step="any" name="Temperature"
required></label><br><br>
        <label>Humidity: <input type="number" step="any" name="Humidity"
required></label><br><br>
        <label>Wind Speed: <input type="number" step="any" name="WindSpeed"
required></label><br><br>
        <label>Cloud Cover: <input type="number" step="any" name="CloudCover"
required></label><br><br>
        <label>Pressure: <input type="number" step="any" name="Pressure"
required></label><br><br>
        <button type="submit">Predict</button>
    </form>

    {% if prediction %}
        <h2>Prediction: {{ prediction }}</h2>
    {% endif %}
```

```
    {% if pca_data %}
      <h3>PCA Components of Your Input:</h3>
      <p><strong>PC1:</strong> {{ pca_data['PC1'] }}</p>
      <p><strong>PC2:</strong> {{ pca_data['PC2'] }}</p>
    {% endif %}

    {% if pca_samples %}
      <h3>Sample PCA-Transformed Data (First 5 Rows):</h3>
      <table border="1">
        <tr>
          <th>Sample</th>
          <th>PC1</th>
          <th>PC2</th>
        </tr>
        {% for row in pca_samples %}
        <tr>
          <td>{{ loop.index }}</td>
          <td>{{ row['PC1'] }}</td>
          <td>{{ row['PC2'] }}</td>
        </tr>
        {% endfor %}
      </table>
    {% endif %}
  </body>
  </html>
```

OUTPUT :



RESULT :

Thus performing model testing and prediction with a user interface on penguins dataset have been done successfully.

Exp. No: 11	Installation of OpenStack
26/3/25	

Aim

To install OpenStack, a free and standard open source cloud computing infrastructure software project.

OpenStack Installation Procedure – DevStack

DevStack is a series of extensible scripts used to quickly bring up a complete OpenStack environment based on the latest versions of everything from git master.

1) Perform Updates in WSL/VM

sudo apt update

sudo apt -y upgrade

sudo apt -y dist-upgrade

2) Add OpenStack User

sudo useradd -s /bin/bash -d /opt/stack -m stack

echo "stack ALL=(ALL) NOPASSWD: ALL" | sudo tee /etc/sudoers.d/stack

```
yuvaraj@Yuvaraj-Laptop:~$ sudo useradd -s /bin/bash -d /opt/stack -m stack
yuvaraj@Yuvaraj-Laptop:~$ echo "stack ALL=(ALL) NOPASSWD: ALL" | sudo tee /etc/sudoers.d/stack
stack ALL=(ALL) NOPASSWD: ALL
```

3) Navigate to Stack and Root

sudo su - stack

sudo su -

su – stack

```
yuvaraj@Yuvaraj-Laptop:~$ sudo su - stack
stack@Yuvaraj-Laptop:~$ sudo su -
root@Yuvaraj-Laptop:~# su - stack
stack@Yuvaraj-Laptop:~$
```

4) Install Git

sudo apt -y install git

5) Clone DevsStack Repository

git clone https://github.com/openstack-dev/devstack.git

```
stack@Yuvaraj-Laptop:~$ git clone https://github.com/openstack-dev/devstack.git
Cloning into 'devstack'...
remote: Enumerating objects: 51577, done.
remote: Counting objects: 100% (932/932), done.
remote: Compressing objects: 100% (433/433), done.
remote: Total 51577 (delta 666), reused 526 (delta 499), pack-reused 50645 (from 5)
Receiving objects: 100% (51577/51577), 16.62 MiB | 2.49 MiB/s, done.
Resolving deltas: 100% (35929/35929), done.
stack@Yuvaraj-Laptop:~$
```

6) DevStack Configuration

cd devstack

vi local.conf

```
stack@Yuvaraj-Laptop:~$ cd devstack
stack@Yuvaraj-Laptop:~/devstack$ vi local.conf
```

In local.conf,

```
[[local|localrc]]
HOST_IP=127.0.0.1 # IP Address of Device
ADMIN_PASSWORD=password
DATABASE_PASSWORD=$ADMIN_PASSWORD
RABBIT_PASSWORD=$ADMIN_PASSWORD
SERVICE_PASSWORD=$ADMIN_PASSWORD
```

```
:wq
```

7) Final Installation

```
./stack.sh
```

Takes 1 to 2 hours, largely depending on the speed of your internet connection. Many git trees and packages are installed during this process.

RESULT :

Hence, the installation of OpenStack was successfully completed through DevStack.

Exp. No: 12	Installation and Setup of Hadoop
26/3/25	

Aim

To install and configure Hadoop in pseudo-distributed mode.

Apache Hadoop Installation and Setup

Hadoop is an open-source software framework designed for storing and processing large datasets across clusters of computers, enabling distributed computing and handling big data analytics.

1) Download JDK

- Apache Hadoop 3.3 and upper supports Java 8 and Java 11 (runtime only).
- So, we need to compile Hadoop with Java 8. Compiling Hadoop with Java 11 is not supported.
- Installation of Java SE Development Kit 8u202 is done in Oracle.
- The Windows x64 exe file is downloaded.
- Java installation is done.
- Java Environment Variables are set.

User Variable | JAVA_HOME: C:\Java\jdk-1.8\bin

System Variable Path | C:\Java\jdk-1.8\bin

2) Download Hadoop

- Download any version of Hadoop that is above 3.3.
- The latest available version is 3.4.1, so download the associated tar file through a binary mirror.
- Extract the tar file.

3) Edit Hadoop Files

- The following XML and CMD Script files in C:/hadoop-3.3.0/etc/hadoop have to be edited.
    1. core-site.xml
    2. mapred-site.xml
    3. hdfs-site.xml
    4. yarn-site.xml
    5. hadoop-env.cmd

core-site.xml

```
<configuration>
 <property>
 <name>fs.default.name</name>
 <value>hdfs://localhost:9000</value>
 </property>
</configuration>
```

mapred-site.xml

```
<configuration>
 <property>
 <name>mapreduce.framework.name</name>
 <value>yarn</value>
 </property>
</configuration>
```

hdfs-site.xml

// Create folder "data" under "C:\Hadoop-3.3.0"

// Create folder "datanode" under "C:\Hadoop-3.3.0\data"

// Create folder "namenode" under "C:\Hadoop-3.3.0\data"

```
<configuration>
 <property>
<name>dfs.replication</name>
 <value>1</value>
 </property>
 <property>
 <name>dfs.namenode.name.dir</name>
 <value>file:///C:/hadoop-3.3.0/data/namenode</value>
 </property>
 <property>
 <name>dfs.datanode.data.dir</name>
 <value>/C:/hadoop-3.3.0/data/datanode</value>
 </property>

</configuration>
```

yarn-site.xml

```
<configuration>
 <property>
```

```
 <name>yarn.nodemanager.aux-services</name>
 <value>mapreduce_shuffle</value>
 </property>
 <property>
 <name>yarn.nodemanager.auxservices.mapreduce.shuffle.class</name>
<value>org.apache.hadoop.mapred.ShuffleHandler</value>
 </property>
</configuration>
```

hadoop-env.cmd

set JAVA_HOME=C:\Java\jdk-1.8

Install Visual C++ Redistributable Packages for Visual Studio 2013 if not available.

It is available if msvcr120.dll is present in C:\Windows\System32.

4) Hadoop Environment Variables

User Variable	HADOOP_HOME: C:\hadoop-3.4.1
System Variable Path 1	C:\hadoop-3.4.1\bin
System Variable Path 2	C:\hadoop-3.4.1\sbin

Then, add the additional configuration files in the bin folder.

5) Setting Up Hadoop in CMD

Run these commands in CMD with Administrator privileges.

Delete the tmp file that is created in C drive.

Ensure that the hadoop-3.4.1 folder is NOT read-only.

hdfs namenode -format

```
C:\Users\TC>hdfs namenode -format
2025-03-18 21:44:34,541 INFO namenode.NameNode: STARTUP_MSG:
/************************************************************
STARTUP_MSG: Starting NameNode
STARTUP_MSG:   host = Yuvaraj-Laptop/192.168.1.26
STARTUP_MSG:   args = [-format]
STARTUP_MSG:   version = 3.4.1
```

…

```
Re-format filesystem in Storage Directory root= C:\hadoop-3.4.1\data\namenode; location= null ? (Y or N) Y
2025-03-18 21:44:40,567 INFO namenode.FSImage: Allocated new BlockPoolId: BP-775212812-192.168.1.26-1742314480551
2025-03-18 21:44:40,567 INFO common.Storage: Will remove files: [C:\hadoop-3.4.1\data\namenode\current\fsimage_000000000
0000000000, C:\hadoop-3.4.1\data\namenode\current\fsimage_0000000000000000000.md5, C:\hadoop-3.4.1\data\namenode\current
\seen_txid, C:\hadoop-3.4.1\data\namenode\current\VERSION]
2025-03-18 21:44:40,591 INFO common.Storage: Storage directory C:\hadoop-3.4.1\data\namenode has been successfully forma
tted.
2025-03-18 21:44:40,619 INFO namenode.FSImageFormatProtobuf: Saving image file C:\hadoop-3.4.1\data\namenode\current\fsi
mage.ckpt_0000000000000000000 using no compression
2025-03-18 21:44:40,692 INFO namenode.FSImageFormatProtobuf: Image file C:\hadoop-3.4.1\data\namenode\current\fsimage.ck
pt_0000000000000000000 of size 397 bytes saved in 0 seconds .
2025-03-18 21:44:40,694 INFO namenode.NNStorageRetentionManager: Going to retain 1 images with txid >= 0
2025-03-18 21:44:40,710 INFO blockmanagement.DatanodeManager: Slow peers collection thread shutdown
2025-03-18 21:44:40,710 INFO namenode.FSNamesystem: Stopping services started for active state
2025-03-18 21:44:40,710 INFO namenode.FSNamesystem: Stopping services started for standby state
2025-03-18 21:44:40,710 INFO namenode.FSImage: FSImageSaver clean checkpoint: txid=0 when meet shutdown.
2025-03-18 21:44:40,710 INFO namenode.NameNode: SHUTDOWN_MSG:
/************************************************************
SHUTDOWN_MSG: Shutting down NameNode at Yuvaraj-Laptop/192.168.1.26
************************************************************/
```

In C:\hadoop-3.4.1\sbin, to start daemon services

start-dfs

jps

```
C:\hadoop-3.4.1\sbin>start-dfs

C:\hadoop-3.4.1\sbin>jps
11248 DataNode
11472 Jps
1604 NameNode

C:\hadoop-3.4.1\sbin>
```

start-yarn

```
C:\hadoop-3.4.1\sbin>start-yarn
starting yarn daemons

C:\hadoop-3.4.1\sbin>jps
11968 ResourceManager
16836 Jps
8500 NameNode
17448 DataNode
2664 NodeManager
```

jps is used to view all processes/services and their IDs.

6) Hadoop Overview in Browser

localhost:9870

Overview 'localhost:9000' (✔active)

Started:	Tue Mar 18 23:05:10 +0530 2025
Version:	3.4.1, r4d7825309348956336b8f06a08322b78422849b1
Compiled:	Wed Oct 09 20:27:00 +0530 2024 by mthakur from branch-3.4.1
Cluster ID:	CID-1527dbd4-f394-476a-8293-e1aaea39712f
Block Pool ID:	BP-1682476646-192.168.1.26-1742319300670

Summary

Security is off.

Safemode is off.

1 files and directories, 0 blocks (0 replicated blocks, 0 erasure coded block groups) = 1 total filesystem object(s).

Heap Memory used 186.48 MB of 334.5 MB Heap Memory. Max Heap Memory is 889 MB.

Non Heap Memory used 56.52 MB of 57.92 MB Commited Non Heap Memory. Max Non Heap Memory is <unbounded>.

Configured Capacity:	153.71 GB
Configured Remote Capacity:	0 B
DFS Used:	149 B (0%)
Non DFS Used:	77.04 GB
DFS Remaining:	76.67 GB (49.88%)
Block Pool Used:	149 B (0%)
DataNodes usages% (Min/Median/Max/stdDev):	0.00% / 0.00% / 0.00% / 0.00%
Live Nodes	1 (Decommissioned: 0, In Maintenance: 0)
Dead Nodes	0 (Decommissioned: 0, In Maintenance: 0)
Decommissioning Nodes	0
Entering Maintenance Nodes	0
Total Datanode Volume Failures	0 (0 B)
Number of Under-Replicated Blocks	0
Number of Blocks Pending Deletion (including replicas)	0
Block Deletion Start Time	Tue Mar 18 23:05:10 +0530 2025
Last Checkpoint Time	Tue Mar 18 23:05:00 +0530 2025
Last HA Transition Time	Never
Enabled Erasure Coding Policies	RS-6-3-1024k

7) Reset

To stop all services, kill browser overview, to ensure smooth restart of Hadoop services

stop-dfs

stop-yarn

In PowerShell,

netstat -ano | findstr :9870

taskkill /PID <PID> /F

In PowerShell,

1) To check all running Java processes

Get-Process java -ErrorAction SilentlyContinue

2) To kill Java processes

Stop-Process -Name java -Force

To reset datanode and namenode, simply remove all the contents in those 2 folders.

Close all Hadoop CMDs.

Follow ALL of these steps before restarting Hadoop.

RESULT :

Hence, Hadoop was successfully installed and configured in the pseudo-distributed mode.

Exp. No: 13	File Management Tasks in Hadoop
2/4/25	

Aim

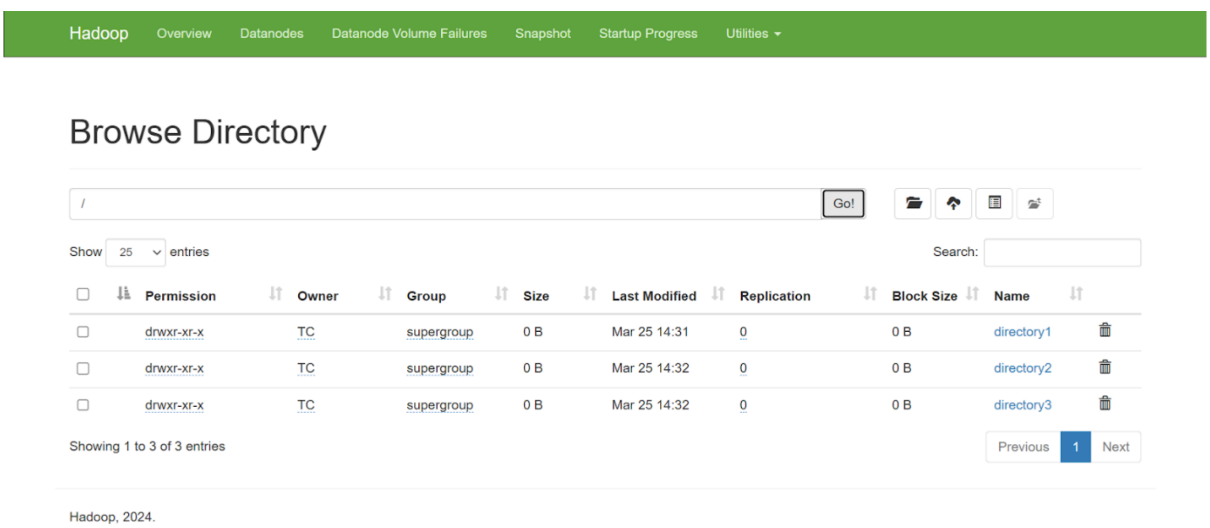
To implement different file management tasks in Hadoop, such as adding directories, files, viewing and deleting files.

Adding Directories

hadoop fs -mkdir /directory1

hadoop fs -mkdir /directory2

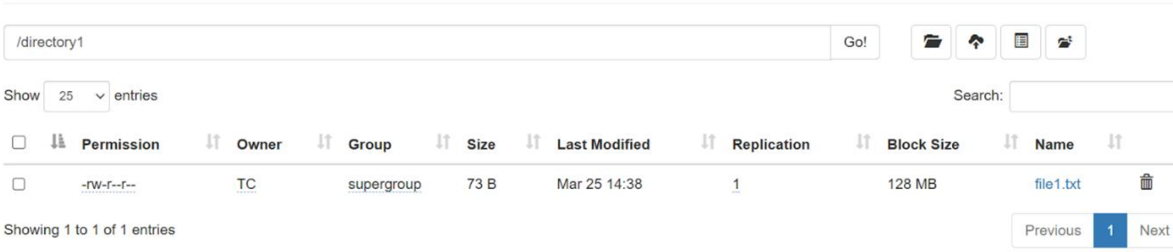
hadoop fs -mkdir /directory3



Adding Files

notepad file1.txt

hadoop fs -put file1.txt /directory1



/directory1 Go!

Show 25 entries Search:

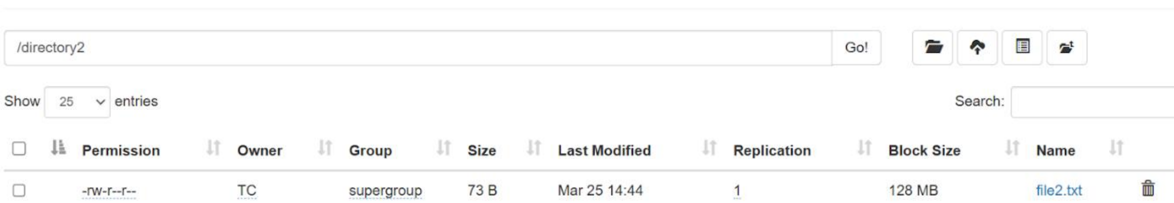
	Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
☐	-rw-r--r--	TC	supergroup	73 B	Mar 25 14:38	1	128 MB	file1.txt

Showing 1 to 1 of 1 entries

Previous 1 Next

notepad file2.txt

hadoop fs -put file2.txt /directory2



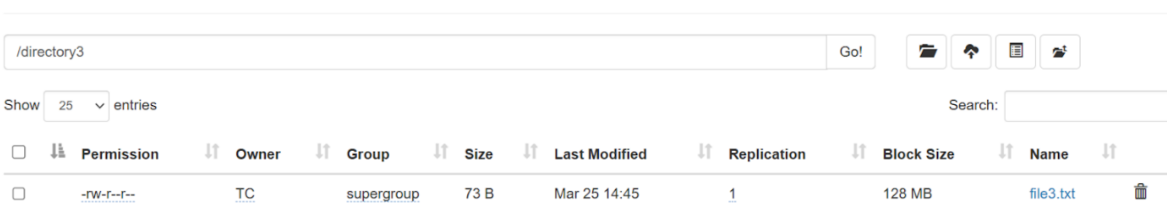
/directory2 Go!

Show 25 entries Search:

	Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
☐	-rw-r--r--	TC	supergroup	73 B	Mar 25 14:44	1	128 MB	file2.txt

notepad file3.txt

hadoop fs -put file3.txt /directory3



/directory3 Go!

Show 25 entries Search:

	Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
☐	-rw-r--r--	TC	supergroup	73 B	Mar 25 14:45	1	128 MB	file3.txt

Retrieve File Contents

hadoop fs -cat /directory1/file1.txt

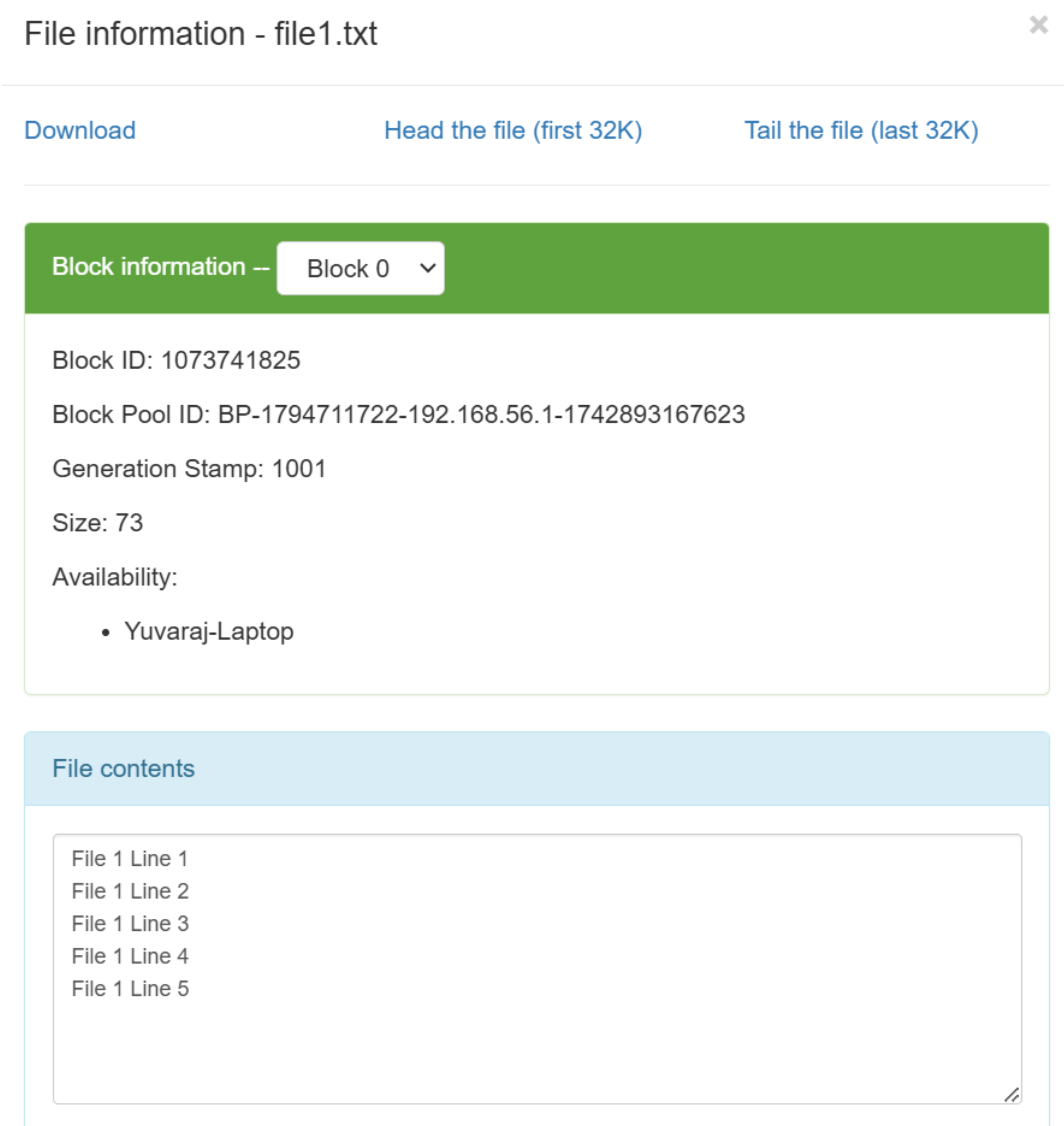
hadoop fs -cat /directory2/file2.txt

hadoop fs -cat /directory3/file3.txt

```
C:\Windows\System32>hadoop fs -cat /directory1/file1.txt
File 1 Line 1
File 1 Line 2
File 1 Line 3
File 1 Line 4
File 1 Line 5
C:\Windows\System32>hadoop fs -cat /directory2/file2.txt
File 2 Line 1
File 2 Line 2
File 2 Line 3
File 2 Line 4
File 2 Line 5
C:\Windows\System32>hadoop fs -cat /directory3/file3.txt
File 3 Line 1
File 3 Line 2
File 3 Line 3
File 3 Line 4
File 3 Line 5
C:\Windows\System32>
```

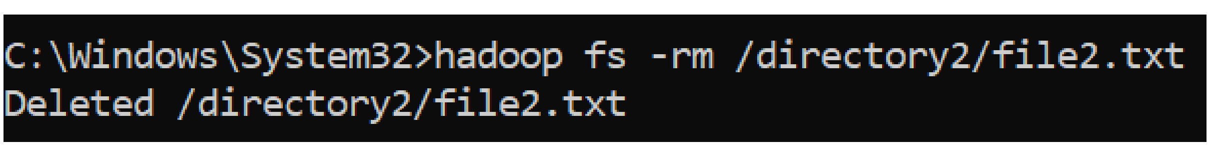
View File Contents

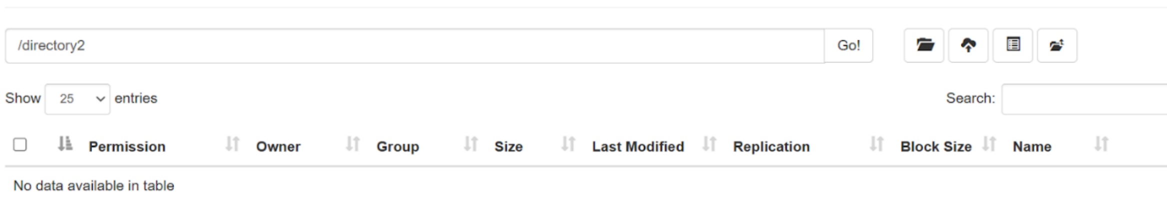
- The contents of a particular file can be viewed through the Hadoop Overview window in the browser.
- It provides options to head the file, tail the file, or download the file.
- We can head and tail the first 32K and last 32K lines respectively, which is mostly sufficient enough for a majority of text files.



Delete File

hadoop fs -rm /directory2/file2.txt





RESULT :

Hence, different file management tasks were successfully implemented in Hadoop.

EXP NO : 14 DATE : 16/4/25	CRUD OPERATIONS IN MONGODB

AIM :

To perform basic CRUD operations in MONGODB.

COMMANDS :

To create and use a database :

use dbname

```
test> use sample
switched to db sample
```

To know the current database :

db

```
sample> db
sample
```

To show the available list of databases :

show dbs

```
sample> show dbs
admin    40.00 KiB
config   12.00 KiB
local    40.00 KiB
```

Create operations :

To insert element into a database :

db.collection_name.insert()

```
sample> db.student.insert({"name":"Abc"})
DeprecationWarning: Collection.insert() is deprecated. Use insertOne, insertMany, or bulkWrite.
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('67ed3f025c568cf3936b140b') }
}
```

```
sample> show dbs
admin    40.00 KiB
config   12.00 KiB
local    40.00 KiB
sample   40.00 KiB
```

To list the documents in a collection :

db.collection_name.find({})

```
sample> db.student.find({})
[ { _id: ObjectId('67ed3f025c568cf3936b140b'), name: 'Abc' } ]
sample> db.student.insertOne({ name: "John Doe", age: 21, course: "Computer Science" });
{
  acknowledged: true,
  insertedId: ObjectId('67ed3ff15c568cf3936b140c')
}
```

To insert many records at a single time :

db.collection_name.insertMany({ … list of documents ..})

```
sample> db.student.insertMany([
...   { name: "Alice", age: 22, course: "Mathematics" },
...   { name: "Bob", age: 23, course: "Physics" }
... ]);
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('67ed400c5c568cf3936b140d'),
    '1': ObjectId('67ed400c5c568cf3936b140e')
  }
}
```

After inserting , listing out the documents to check :

```
sample> db.student.find({});
...
[
  { _id: ObjectId('67ed3f025c568cf3936b140b'), name: 'Abc' },
  {
    _id: ObjectId('67ed3ff15c568cf3936b140c'),
    name: 'John Doe',
    age: 21,
    course: 'Computer Science'
  },
  {
    _id: ObjectId('67ed400c5c568cf3936b140d'),
    name: 'Alice',
    age: 22,
    course: 'Mathematics'
  },
  {
    _id: ObjectId('67ed400c5c568cf3936b140e'),
    name: 'Bob',
    age: 23,
    course: 'Physics'
  }
]
```

Read operations :

```
sample> db.student.find({}).pretty()
[
  { _id: ObjectId('67ed3f025c568cf3936b140b'), name: 'Abc' },
  {
    _id: ObjectId('67ed3ff15c568cf3936b140c'),
    name: 'John Doe',
    age: 21,
    course: 'Computer Science'
  },
  {
    _id: ObjectId('67ed400c5c568cf3936b140d'),
    name: 'Alice',
    age: 22,
    course: 'Mathematics'
  },
  {
    _id: ObjectId('67ed400c5c568cf3936b140e'),
    name: 'Bob',
    age: 23,
    course: 'Physics'
  }
]
```

To find documents with conditions :

gt – greater than

db.collection_name.find({ key : {$gt : value } })

```
sample> db.student.find({ age: { $gt: 21 } })
[
  {
    _id: ObjectId('67ed400c5c568cf3936b140d'),
    name: 'Alice',
    age: 22,
    course: 'Mathematics'
  },
  {
    _id: ObjectId('67ed400c5c568cf3936b140e'),
    name: 'Bob',
    age: 23,
    course: 'Physics'
  }
]
```

To simply find using a key :

```
sample> db.student.find({ name: "Alice" })
[
  {
    _id: ObjectId('67ed400c5c568cf3936b140d'),
    name: 'Alice',
    age: 22,
    course: 'Mathematics'
  }
]
```

Update operations :

db.collection_name.updateOne( )

```
sample> db.student.updateOne({ name: "John Doe" }, { $set: { age: 22 } });
...
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
sample> db.student.find({});
[
  { _id: ObjectId('67ed3f025c568cf3936b140b'), name: 'Abc' },
  {
    _id: ObjectId('67ed3ff15c568cf3936b140c'),
    name: 'John Doe',
    age: 22,
    course: 'Computer Science'
  },
  {
    _id: ObjectId('67ed400c5c568cf3936b140d'),
    name: 'Alice',
    age: 22,
    course: 'Mathematics'
  },
  {
    _id: ObjectId('67ed400c5c568cf3936b140e'),
    name: 'Bob',
    age: 23,
    course: 'Physics'
  }
]
```

db.collection_name.updateMany({})

```
]
sample> db.student.updateMany({}, { $set: { status: "active" } });
...
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 4,
  modifiedCount: 4,
  upsertedCount: 0
}
sample> db.student.find({});
[
  {
    _id: ObjectId('67ed3f025c568cf3936b140b'),
    name: 'Abc',
    status: 'active'
  },
  {
    _id: ObjectId('67ed3ff15c568cf3936b140c'),
    name: 'John Doe',
    age: 22,
    course: 'Computer Science',
    status: 'active'
  },
  {
    _id: ObjectId('67ed400c5c568cf3936b140d'),
    name: 'Alice',
    age: 22,
    course: 'Mathematics',
    status: 'active'
  },
  {
    _id: ObjectId('67ed400c5c568cf3936b140e'),
    name: 'Bob',
    age: 23,
    course: 'Physics',
    status: 'active'
  }
]
```

db.collection_name.replaceOne({})

```
sample> db.student.replaceOne({ name: "Alice" }, { name: "Alice", age: 23, course: "Statistics" });
...
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
sample> db.student.find({});
[
  {
    _id: ObjectId('67ed3f025c568cf3936b140b'),
    name: 'Abc',
    status: 'active'
  },
  {
    _id: ObjectId('67ed3ff15c568cf3936b140c'),
    name: 'John Doe',
    age: 22,
    course: 'Computer Science',
    status: 'active'
  },
  {
    _id: ObjectId('67ed400c5c568cf3936b140d'),
    name: 'Alice',
    age: 23,
    course: 'Statistics'
  },
  {
    _id: ObjectId('67ed400c5c568cf3936b140e'),
    name: 'Bob',
    age: 23,
    course: 'Physics',
    status: 'active'
  }
]
```

DELETE OPERATIONS :

db.collection_name.deleteOne( )

```
sample> db.student.deleteOne({ name: "John Doe" });
...
{ acknowledged: true, deletedCount: 1 }
sample> db.student.find({});
[
  {
    _id: ObjectId('67ed3f025c568cf3936b140b'),
    name: 'Abc',
    status: 'active'
  },
  {
    _id: ObjectId('67ed400c5c568cf3936b140d'),
    name: 'Alice',
    age: 23,
    course: 'Statistics'
  },
  {
    _id: ObjectId('67ed400c5c568cf3936b140e'),
    name: 'Bob',
    age: 23,
    course: 'Physics',
    status: 'active'
  }
]
```

db.collection_name.deleteMany()

```
sample> db.student.deleteMany({ age: { $gt: 22 } });
...
{ acknowledged: true, deletedCount: 2 }
sample> db.student.find({});
[
  {
    _id: ObjectId('67ed3f025c568cf3936b140b'),
    name: 'Abc',
    status: 'active'
  }
]
```

db.collection_name.drop()

```
sample> db.student.drop();
...
true
sample> db.student.find({});
```

RESULT :

Thus CRUD operations in MONGODB have been performed successfully.