

Experiment 1

Implement following operations on python data structures

1. Write a Python program to reverse a list.
2. How do you remove duplicates from a list in Python?
3. Explain the difference between a list and a tuple in Python.
4. Write a Python program to count the occurrences of an element in a tuple.
5. Write a Python program to merge two dictionaries.
6. Write a program to sort a dictionary by its keys or values.
7. Write a Python program to find the intersection two sets?
8. Write a Python program to find the union of two sets.

Code:

```
list1=[1,2,3,4,5]
```

```
list1[::-1]
```

```
list_ex = [1, 2, 3]
```

```
tuple_ex = (1, 2, 3)
```

```
my_tuple = (1, 2, 3, 2, 4, 2, 5)
```

```
count = my_tuple.count(3)
```

```
print("Occurrences of 2:", count)
```

```
dict1 = {'a': 1, 'b': 2}
```

```
dict2 = {'c': 3, 'd': 4}
```

```
dict1.update(dict2)
```

```
print("Merged dictionary:", dict1)
```

```
my_dict = {'b': 3, 'a': 1, 'c': 2}
```

```
sorted_by_keys = dict(sorted(my_dict.items()))
```

```
print("Sorted by keys:", sorted_by_keys)
```

```
set1 = {1, 2, 3, 4}
```

```
set2 = {3, 4, 5, 6}

intersection = set1 & set2 # or set1.intersection(set2)

print("Intersection:", intersection)
```

```
set1 = {1, 2, 3, 4}

set2 = {3, 4, 5, 6}

union = set1 | set2 # or set1.union(set2)

print("Union:", union)
```

Experiment 2

Introduction to pandas : take Sports dataset that shows the results from NCAA basketball games from 1985 to 2016. Execute following questions using pandas

- 1. Display sample records using head() and tail() function.**
- 2. Display number of records in dataset**
- 3. Display the number of missing values in each column.**
- 4. Display maximum value present in each column**
- 5. Display unique values present in each column**
- 6. Display number of unique values present in each column**
- 7. Display number of times unique values appeared in each column.**

```
import pandas as pd

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

print(df.head())

print(df.tail())

print("Rows:", df.shape[0])

print("Columns:", df.shape[1])


print(df.isnull().sum())

print(df.max(numeric_only=True))


for col in df.columns:

    print(f"\nColumn: {col}")

    print(df[col].unique())
```

```
print(df.nunique())
```

```
for col in df.columns:
```

```
    print(f"\nColumn: {col}")
```

```
    print(df[col].value_counts())
```

Experiment 3

Execute Basic operations and functionalities of the NumPy library in Python.

- 1. Create a 1D NumPy array with values ranging from 10 to 49.**
- 2. Reshape the array into a 3x5 matrix.**
- 3. Extract the elements that are divisible by 3 from the original array.**
- 4. Create two 3x3 NumPy arrays with random integers.**
- 5. Perform element-wise addition, subtraction, multiplication, and division.**

```
import numpy as np
```

```
arr = np.arange(10, 50)
```

```
print("1D Array:\n", arr)
```

```
matrix = arr[:15].reshape(3, 5)
```

```
print("\n3x5 Matrix:\n", matrix)
```

```
div3 = arr[arr % 3 == 0]
```

```
print("\nElements divisible by 3:\n", div3)
```

```
np.random.seed(0)
```

```
a = np.random.randint(0, 10, (3,3))
```

```
b = np.random.randint(0, 10, (3,3))
```

```
print("\nArray a:\n", a)
```

```
print("\nArray b:\n", b)
```

```
print("\nAddition:\n", a + b)
```

```
print("\nSubtraction:\n", a - b)
```

```
print("\nMultiplication:\n", a * b)
```

```
print("\nDivision:\n", a / b)
```

Experiment 4

Array Manipulations and Boolean Indexing

1. Create a 1D NumPy array with 15 evenly spaced numbers between 1 and 100.
2. Create a 4x4 array with random integers between 1 and 50 and replace all even numbers with -1.
3. Create a 5x5 matrix of random integers between 1 and 100 and retrieve all elements in second row and last column
4. Create a NumPy array with integers from 1 to 50 and extract all the odd numbers.
5. Create a 3x3 array with random floating-point values and Calculate its mean

```
arr1 = np.linspace(1, 100, 15)
print("1D Array with 15 evenly spaced numbers:\n", arr1)

np.random.seed(0)
arr2 = np.random.randint(1, 51, (4, 4))
arr2[arr2 % 2 == 0] = -1
print("\n4x4 Array with even numbers replaced by -1:\n", arr2)

arr3 = np.random.randint(1, 101, (5, 5))
second_row = arr3[1, :] # second row (index 1)
last_column = arr3[:, -1] # last column
print("\n5x5 Random Matrix:\n", arr3)
print("\nSecond row:\n", second_row)
print("Last column:\n", last_column)

arr4 = np.arange(1, 51)
odd_numbers = arr4[arr4 % 2 == 1]
print("\nOdd numbers from 1 to 50:\n", odd_numbers)

arr5 = np.random.rand(3, 3) # random floats between 0 and 1
mean_val = arr5.mean()
print("\n3x3 Random Float Array:\n", arr5)
print("Mean of the array:", mean_val)
```

```
arr1 = np.arange(1, 17).reshape(4, 4)
print("4x4 Matrix:\n", arr1)
```

```
arr5 = np.random.rand(3, 3) # random floats between 0 and 1
mean_val = arr5.mean()
print("\n3x3 Random Float Array:\n", arr5)
print("Mean of the array:", mean_val)
```

Experiment 5

Reshaping, Transposing, and Element-wise Operations

1. Create a 1D NumPy array with values from 1 to 16 and reshape it into a 4x4 matrix.
2. Create a 3x4 matrix with random integers between 10 and 99, transpose it, and print its shape.
3. Create a 1D array with 10 elements (random integers) and compute the square of each element.
4. Create a 2D NumPy array filled with zeros of shape (4, 5).
5. Create a NumPy array of integers from 1 to 20 and extract all elements that are greater than 5.

```
arr1 = np.arange(1, 17).reshape(4, 4)
print("4x4 Matrix:\n", arr1)
```

```
np.random.seed(0)
arr2 = np.random.randint(10, 100, (3, 4))
arr2_transposed = arr2.T
print("\nOriginal 3x4 Matrix:\n", arr2)
print("Transposed Matrix:\n", arr2_transposed)
print("Shape of Transposed Matrix:", arr2_transposed.shape)
```

```
arr3 = np.random.randint(1, 20, 10)
arr3_squared = arr3 ** 2
print("\n1D Array:", arr3)
print("Squared Elements:", arr3_squared)
```

```
zeros_array = np.zeros((4, 5))
print("\n2D Array of Zeros:\n", zeros_array)
```

```
arr4 = np.arange(1, 21)
greater_than_5 = arr4[arr4 > 5]
print("\nElements greater than 5:", greater_than_5)
```

Experiment 6

Data scraping techniques (web scraping, API integration) :Apply data scraping techniques by extracting data from a web page (url: https://en.wikipedia.org/wiki/World_population) . Retrieve tables from the Wikipedia page on "World Population" using web scraping methods and display the data in Pandas DataFrames for further analysis

```
import pandas as pd

url = https://en.wikipedia.org/wiki/World\_population

tables = pd.read_html(url)

print("Total Tables Found on Page:", len(tables))
print("\nFirst Table:")
print(tables[0].head())

if len(tables) > 1:
    print("\nSecond Table:")
    print(tables[1].head())
```

Experiment 7

Data Cleaning and Preprocessing with the Titanic Dataset: Handling Missing Values and Outliers.

- 1. Understand data using head, tail, shape, column name etc**
- 2. Describe function used to find the total count, mean, standard deviation, minimum value, maximum value, 25,50,75 percentage of the dataset value**
- 3. how many nulls present in each attribute of dataset**
- 4. Handle the missing values of Age and Cabin attribute as : Fill missing values of age attribute by mean of age. Fill missing values of Cabin attribute by value "Unkown".**
- 5. Detect outliers present in attributes(e.g. for Fare attribute). Use matplotlib to detect it. Replace outliers with some values within range.**

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

df=pd.read_csv("/content/Titanic-Dataset.csv")
df.columns
df.describe()
df.info()
df.isnull().sum()

mean1=df['Age'].mean()
df['Age'].fillna(mean1,inplace=True)

df['Cabin'].fillna("Unknown")

plt.figure(figsize=(5,5))
sns.boxplot(df['Fare'])
plt.title("Fare Boxplot")
plt.show()
Q1=df["Fare"].quantile(0.25)
Q3=df["Fare"].quantile(0.75)

IQR=Q3-Q1

lower_bound=Q1-(1.5*IQR)
upper_bound=Q3+(1.5*IQR)

print("Lower Bound: ",lower_bound)
print("Upper Bound: ",upper_bound)
```

```
median=df['Fare'].median()

median

df["Fare"]=df["Fare"].apply(lambda x:upper_bound if(x<lower_bound) or(x>upper_bound)
else x)

plt.figure(figsize=(5,5))

sns.boxplot(df['Fare'])

plt.title("Fare Boxplot")

plt.show()
```

Experiment 8

Exploratory Data Analysis (EDA): Analyze the performance of students in various subjects using a dataset of student grades. Perform Exploratory Data Analysis (EDA) to identify patterns, correlations, and trends in the data. Visualize the distribution of grades, the correlation between subjects, and the impact of attendance on grades using Matplotlib and Seaborn. (Use dataset from Kaggle or UCI Machine Learning Repository)

[Student Performance - UCI Machine Learning Repository](#)

[Students' Academic Performance Dataset \(kaggle.com\)](#)

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

df=pd.read_csv("/content/student_info.csv")

df.shape

df.head()

df.info()

df.describe()

df.isnull().sum()
```

```
plt.figure(figsize=(6,6))
sns.histplot(df['attendance_rate'],bins=30,kde=True,color='skyblue',edgecolor="black")
plt.title("Attendance Rate Distribution")
plt.show()
```

```
plt.figure(figsize=(8,6))
sns.countplot(x="extra_activities",data=df,palette="Set2")
plt.title("Extra-Curricular Activities")
plt.show()
```

```
plt.figure(figsize=(6,6))
sns.countplot(x="final_result",data=df,order=['Pass','Fail'],palette="Set2")
plt.title("Final Result of Students")
plt.show()
```

```
sns.barplot(x='gender', y='math_score', data=df, ci=None, palette='Set2')
plt.title("Average Math Score by Gender")
plt.show()

sns.barplot(x='parent_education', y='math_score', data=df, ci=None, palette='coolwarm')
plt.title("Average Math Score vs Parent Education")
plt.xticks(rotation=45)
plt.show()
```

```
sns.scatterplot(x='attendance_rate',
                y='math_score',
                data=df,
                hue='gender',
                palette='coolwarm')
```

```

corr=df[['age','grade_level','math_score', 'reading_score','writing_score', 'attendance_rate', 'study_
hours']].corr()

plt.figure(figsize=(10,6))

sns.heatmap(corr,annot=True,cmap="coolwarm")

plt.title("Correlation Heatmap")

plt.show()

```

Experiment 9

Dataset Link: House Sales in King County, USA - Kaggle

1. Load the dataset and display summary statistics using describe(). What insights can you derive about house prices?
2. Check for missing values and propose methods to handle any missing data in the dataset.
3. Visualize the distribution of house prices using a histogram. Are there any significant outliers?
4. Use a scatter plot to examine the relationship between sqft_living (living area) and price. Add a regression line to visualize the trend.

Group the data by zipcode and calculate the average house price for each zip code. Visualize this using a bar chart

```

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

```

```

df=pd.read_csv("/content/Copy of House Sales in King County, USA.csv")

df.head()

df.info()

df.describe()

df.shape

df.isnull().sum()

```

```
sns.histplot(df['price'],kde=True,color="green",bins=10)
plt.show()
```

```
sns.scatterplot(x=df['sqft_living'],y=df['price'],color="royalblue",alpha=0.7)
plt.show()
```

```
import numpy as np
```

```
x = df['sqft_living']
y = df['price']
m, b = np.polyfit(x, y, 1) # slope and intercept
```

```
plt.scatter(x, y, color='royalblue', alpha=0.7)
plt.plot(x, m*x + b, color='red') # regression line
plt.title("sqft_living vs price with manual regression line")
plt.xlabel("Living Area (sqft)")
plt.ylabel("Price ($)")
plt.show()
```

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

```
x = df['sqft_living']
y = df['price']
```

```
# Regression coefficients
m, b = np.polyfit(x, y, 1)
```

```
# Scatter plot
```

```

sns.scatterplot(x=x, y=y, color='royalblue', alpha=0.6)

# Dotted regression line
plt.plot(x, m*x + b, color='red', linewidth=2, linestyle='--') # <-- dashed line
# OR plt.plot(x, m*x + b, color='red', linewidth=2, linestyle=':') # dotted line

plt.title("Living Area vs Price (with Dotted Regression Line)")
plt.xlabel("Living Area (sqft)")
plt.ylabel("Price ($)")
plt.show()

df['zipcode'].unique()

avg_price_zipcode=df.groupby("zipcode")["price"].mean().sort_values(ascending=False)
print(avg_price_zipcode)

top_zip=avg_price_zipcode.head(15)
plt.figure(figsize=(5,5))
sns.barplot(x=top_zip.index.astype(str),y=top_zip.values,palette="viridis")
plt.title("Top 15 Zipcodes with Highest Average Price")
plt.xlabel("Zipcode")
plt.ylabel("Average Price")
plt.xticks(rotation=45)
plt.show()

```

Experiment 10

Dataset Link: Titanic Dataset on Kaggle

1. Load the dataset and display the first 10 rows using head(). What are the column names and their data types?
2. Check for missing values in the dataset and identify the columns with the most missing data.
3. Analyze the survival rate across different age groups using a bar plot.

4. **Examine the survival rate based on the passenger class (Pclass) and visualize it using a bar chart.**
5. **Create a heatmap of the correlation matrix for the numerical features. What are the most correlated features?**

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

df=pd.read_csv("/content/Copy of Titanic-Dataset.csv")
sns.barplot(x='Pclass', y='Survived', data=df, palette='coolwarm')

plt.title("Survival Rate by Passenger Class")
plt.xlabel("Passenger Class (1=Upper, 3=Lower)")
plt.ylabel("Average Survival Rate")
plt.show()

# Create age groups (bins)
bins = [0, 12, 18, 35, 60, 100]
labels = ['Child', 'Teen', 'Adult', 'Middle-Aged', 'Senior']
df['AgeGroup'] = pd.cut(df['Age'], bins=bins, labels=labels)##VVVP command

# Bar plot of survival rate by AgeGroup
sns.barplot(x='AgeGroup', y='Survived', data=df, palette='Set2')

plt.title("Survival Rate by Age Group")
plt.xlabel("Age Group")
plt.ylabel("Average Survival Rate")
plt.show()
```

Experiment 11

Perform Exploratory Data Analysis (EDA) to answer questions Take Sports dataset that shows the results from NCAA basketball games from 1985 to 2016

1. Display all object columns in dataset(iterate for loop on columns)
2. find all of the games where the winning team scored more than 150 points
3. find out when the winning team scores more than 150 points and when the losing team scores below 100.
4. How many games were played in each season?
5. What is the average winning score for each team?
6. How many games were won at home, away, and at neutral locations?

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
df=pd.read_csv("/content/MRegular.csv")

df.head()

for i in df.columns:
    if df[i].dtype=="object":
        print(i)

counter=0
for i in df['WScore']:
    if i>120:
        print(i)
        counter+=1
print(counter)

count=(df["WScore"]>120).sum()
count

df['WTeamID'].unique().sum()

avg_win_score =
df.groupby('WTeamID')['WScore'].mean().sort_values(ascending=False)
print(avg_win_score)
```

Experiment 12

You are provided with a dataset containing demographic information, including the ages of individuals from different regions. Your task is to perform a data distribution analysis to understand the spread and central tendency of ages in the population. Analyze the distribution to identify patterns, skewness, kurtosis, and the presence of outliers. Visualize the distribution using histograms, box plots, and density plots to effectively communicate your findings.

Dataset Link: <https://archive.ics.uci.edu/ml/datasets/adult>

- 1. What is the central tendency of the age distribution?**
- 2. Is the age distribution skewed or symmetric?**
- 3. Are there any outliers or unusual patterns in the age distribution?**
- 4. How does the age distribution vary across different regions or groups?**

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from scipy.stats import skew,kurtosis

df=pd.read_csv("/content/adult - adult.csv")

df.head()

#Central tendency

mean=df['age'].mean()

median=df['age'].median()

mode=df['age'].mode()


# Visualization

plt.figure(figsize=(7,5))

sns.histplot(df['age'], kde=True, bins=20, color='skyblue')

plt.title("Age Distribution - Histogram & Density Plot")

plt.show()


sns.boxplot(x=df['age'], color='lightgreen')

plt.title("Boxplot - Age Outliers")

plt.show()


plt.figure(figsize=(10,5))

sns.boxplot(x='workclass', y='age', data=df, palette='Set3')

plt.title("Age Distribution Across Workclasses")
```

```
plt.xticks(rotation=45)
plt.show()
df['workclass'].value_counts()
sns.boxplot(x='gender', y='age', data=df, palette='coolwarm')
plt.title("Age Distribution by Gender")
plt.show()
```

Experiment 13

Apply Naive bays classifier on IRIS inbuild dataset

- 1. Load and preprocess a dataset**
- 2. Split data into training and testing sets.**
- 3. Train and evaluate models using various metrics.**
- 4. Visualize the predicted vs. actual values.**

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

from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, confusion_matrix

iris=load_iris(as_frame=True)
df=iris.frame

df.head()
df.isnull().sum()

plt.figure(figsize=(5,5))
sns.boxplot(data=df[["sepal length (cm)","sepal width (cm)","petal length (cm)","petal width (cm)"]])
```

```
plt.show()
```

```
x=df.drop(columns=['target'])
```

```
y=df['target']
```

```
model=GaussianNB()
```

```
model.fit(x_train,y_train)
```

```
y_pred=model.predict(x_test)
```

```
df['target'].value_counts()
```

```
acc=accuracy_score(y_test,y_pred)
```

```
prec=precision_score(y_test,y_pred,average="macro")
```

```
recall_score(y_test,y_pred,average="macro")
```

```
f1=f1_score(y_test,y_pred,average="macro")
```

```
print(acc,prec,f1)
```

```
cm=confusion_matrix(y_test,y_pred)
```

```
plt.figure(figsize=(5,5))
```

```
sns.heatmap(cm,annot=True,cmap="Blues")
```

```
plt.show()
```

Experiment 14

1. Implement Linear regression and Logistic regression on IRIS inbuild dataset
2. Load and preprocess a dataset
3. Split data into training and testing sets.
4. Train and evaluate models using various metrics.
5. Visualize the predicted vs. actual values.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.datasets import load_iris
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score

iris=load_iris(as_frame=True)
df=iris.frame
df.head()
df.isnull().sum()

x=df.drop(columns=['target'])
y=df['target']
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2)

model=LinearRegression()
model.fit(x_train,y_train)

y_pred=model.predict(x_test)

mae=mean_absolute_error(y_test,y_pred)
mse=mean_squared_error(y_test,y_pred)
r2=r2_score(y_test,y_pred)
print(mae,mse,r2)

plt.figure(figsize=(5,5))
sns.scatterplot(x=y_test,y=y_pred)
plt.xlabel('actual')
plt.ylabel('predicted')
```

```
plt.title('actual vs predicted')
plt.plot([y_test.min(),y_test.max()], [y_test.min(),y_test.max()], 'r--')
plt.show()
```

Experiment 15

Dataset: Titanic (from seaborn library or Kaggle)

- 1. Load the Titanic dataset and preprocess it to handle missing values, encode categorical variables, and scale numerical features.**
- 2. Split the dataset into training and testing sets.**
- 3. Train a logistic regression model to predict survival based on available features.**
- 4. Evaluate the model using accuracy, precision, recall, F1 score, and ROC-AUC metrics.**
- 5. Visualize the feature importance or coefficients of the trained model.**

```
import pandas as pd
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, roc_auc_score,
confusion_matrix, classification_report
import matplotlib.pyplot as plt

df = sns.load_dataset('titanic')
df.head()

df['age'].fillna(df['age'].median(), inplace=True)
df['embarked'].fillna(df['embarked'].mode()[0], inplace=True)
df['deck'].fillna('Unknown', inplace=True)
```

```

le = LabelEncoder()
for col in ['sex', 'embarked', 'class', 'who', 'adult_male', 'alone']:
    df[col] = le.fit_transform(df[col].astype(str))

X = df[['pclass', 'sex', 'age', 'sibsp', 'parch', 'fare', 'embarked', 'alone']]
y = df['survived']

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42)

model = LogisticRegression()
model.fit(X_train, y_train)

y_pred = model.predict(X_test)

print("Accuracy :", accuracy_score(y_test, y_pred))
print("Precision:", precision_score(y_test, y_pred))
print("Recall  :", recall_score(y_test, y_pred))
print("F1 Score :", f1_score(y_test, y_pred))
print("ROC-AUC :", roc_auc_score(y_test, y_pred))

print("\nClassification Report:")
print(classification_report(y_test, y_pred))

cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(5,4))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.title("Confusion Matrix - Logistic Regression")
plt.xlabel("Predicted")

```

```
plt.ylabel("Actual")

plt.show()

coefficients = pd.Series(model.coef_[0], index=X.columns)

coefficients.sort_values().plot(kind='barh', color='teal')

plt.title("Feature Importance (Logistic Regression Coefficients)")

plt.xlabel("Coefficient Value")

plt.show()
```

Experiment 16

Use the tips dataset:

1. Explore the data by visualizing:
2. Distributions using `sns.histplot()` or `sns.kdeplot()`
3. Visualize the total bill and tip amounts using a scatterplot.
4. Use a boxplot to analyze how `total_bill` varies across different days.
5. Plot a heatmap to show the correlation matrix for numeric variables in the dataset.

```
6. import pandas as pd
7. import numpy as np
8. import matplotlib.pyplot as plt
9. import seaborn as sns
   from scipy.stats import skew, kurtosis

   tips = sns.load_dataset("tips")

10.
11. print("Dataset Preview:")
12. print(tips.head())
13.
14. plt.figure(figsize=(6,4))
15. sns.scatterplot(data=tips,x="total_bill",y="tip",hue="sex")
16. plt.title("scatterplot of total bill vs tip")
17. plt.xlabel("total bill")
18. plt.ylabel("tip")
19. plt.show()
20.
21. plt.figure(figsize=(6,4))
22. sns.boxplot(data=tips,x='day',y='total_bill')
23. plt.title("boxplot of the week")
24. plt.xlabel("day of week")
25. plt.ylabel("total_bill")
```

```

26. plt.show()
27.
28. corr_matrix=tips.corr(numeric_only=True)
29. plt.figure(figsize=(6,4))
30. sns.heatmap(corr_matrix,annot=True,color="Blue")
31. plt.title("correlation heatmap of numeric faetures")
32. plt.show()
33.
34.

```

35. Experiment 17

1. **Load the Wine dataset from sklearn.datasets and preprocess it by scaling the features.**
2. **Split the data into training and testing sets.**
3. **Train a Naive Bayes classifier (GaussianNB) on the training set.**
4. **Evaluate the model using classification metrics: accuracy, precision, recall, and F1-score.**
5. **Visualize the confusion matrix using a heatmap to assess the classifier's performance.**

```

wine=load_wine(as_frame=True)

df=wine.frame

df.head()

df.info()

x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2)

model=GaussianNB()

model.fit(x_train,y_train)

y_pred=model.predict(x_test)

acc=accuracy_score(y_test,y_pred)

prec=precision_score(y_test,y_pred,average="weighted")

rc=recall_score(y_test,y_pred,average="weighted")

f1=f1_score(y_test,y_pred,average="weighted")

print(acc,prec,rc,f1)

cm=confusion_matrix(y_test,y_pred)

```

```
plt.figure(figsize=(5,4))
sns.heatmap(cm,annot=True,cmap='coolwarm',fmt='d')
plt.title("confusion matrix-navie bayes")
plt.xlabel("predicted")
plt.ylabel("actual")
plt.show()
```

Experiment 18

Apply following queries in iris dataset

1. Understand data using head, tail, shape, column name etc
2. Describe function used to find the total count, mean, standard deviation, minimum value, maximum value, 25,50,75 percentage of the dataset value
3. how many nulls present in each attribute of dataset
4. Handle the missing values of Age and Cabin attribute as : Fill missing values of age attribute by mean of age. Fill missing values of Cabin attribute by value “Unknown”.
5. Detect outliers present in attributes(e.g. for Fare attribute). Use matplotlib to detect it. Replace outliers with some values within range.

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
df = sns.load_dataset('iris')

# Display first and last 5 rows
print("First 5 rows:")
print(df.head())
print(df.tail())
df.shape
print(df.columns)
print(df.info())
print(df.describe())
print(df.isnull().sum())
df['age']=[25,28,None,30,36]*30
df['cabin']=['c1',None,'b2',None,'a3']*30
```

```
df['age'].fillna(df['age'].mean(),inplace=True)
```

```
df['cabin'].fillna('unknown',inplace=True)
```

```
print(df.isnull().sum())
```

```
plt.figure(figsize=(6,4))
```

```
sns.boxplot(x=df['sepal_length'])
```

```
plt.title("outlier detection sepal length")
```

```
plt.show()
```

```
Q1=df['sepal_length'].quantile(0.25)
```

```
Q3=df['sepal_length'].quantile(0.75)
```

```
IQR=Q3-Q1
```

```
lower_limit= Q1-1.5*IQR
```

```
upper_limit=Q3+1.5*IQR
```

```
print(IQR)
```

```
clean_data=df[(df['sepal_length']>=lower_limit ) & (df['sepal_length'] <=upper_limit)]
```

```
print(clean_data)
```