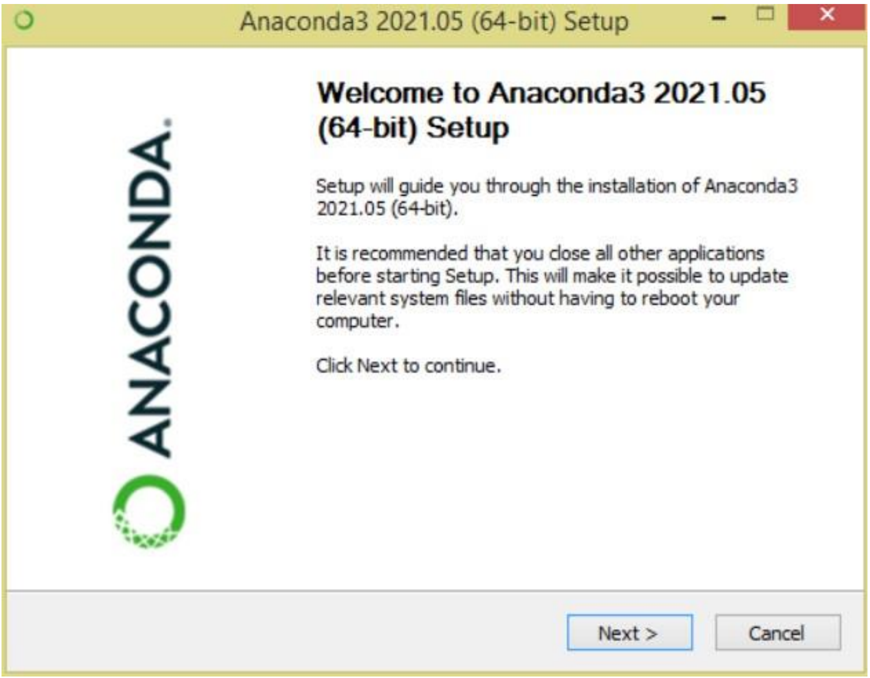
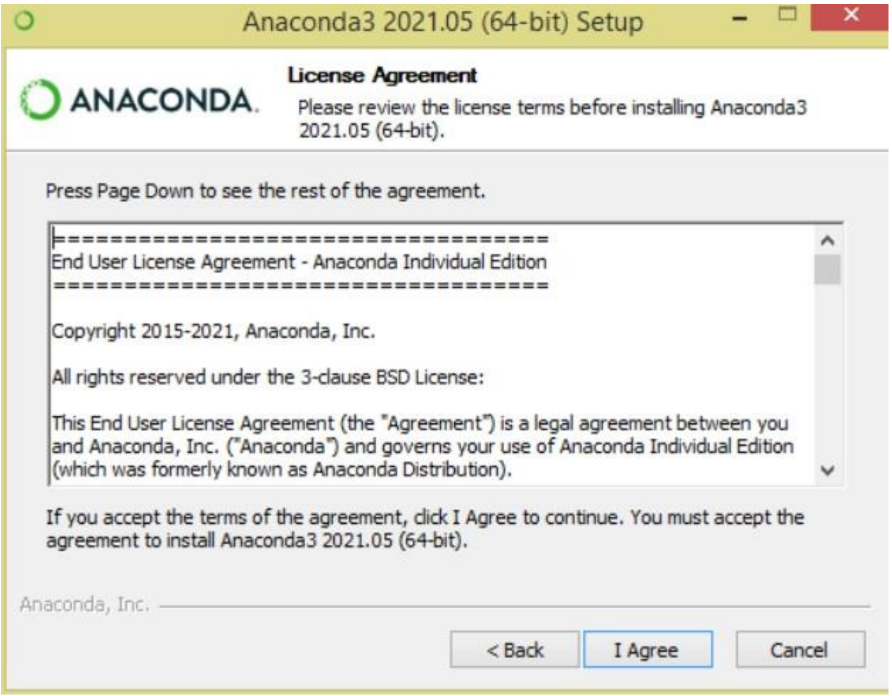
PRACTICAL 1A

AIM: Installation of Anaconda and Jupyter Notebook.

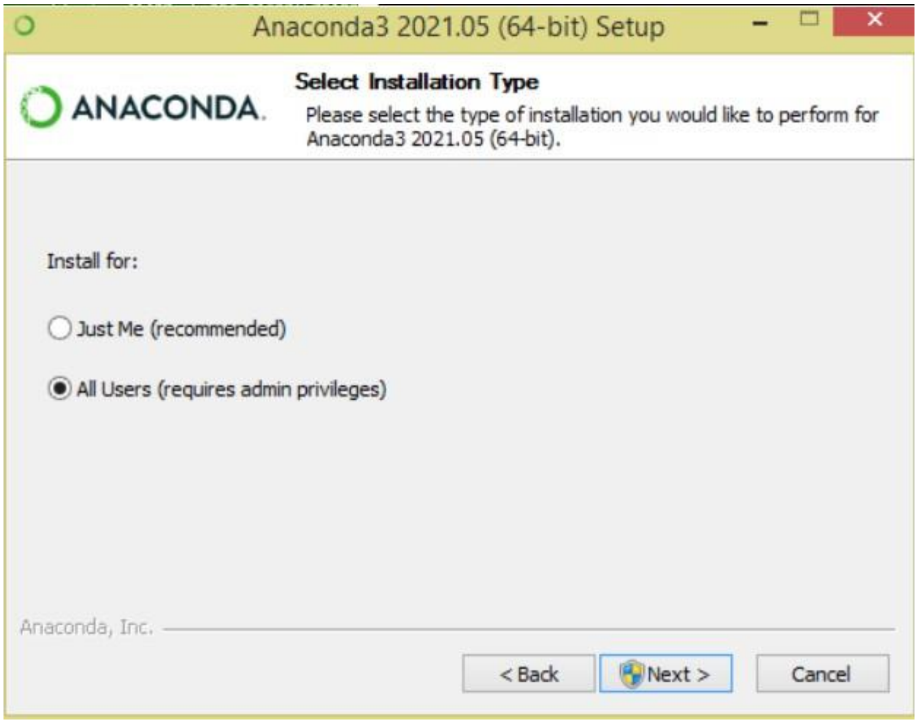
- Download the Anaconda installer.
- Go to your Downloads folder and double-click the installer to launch.



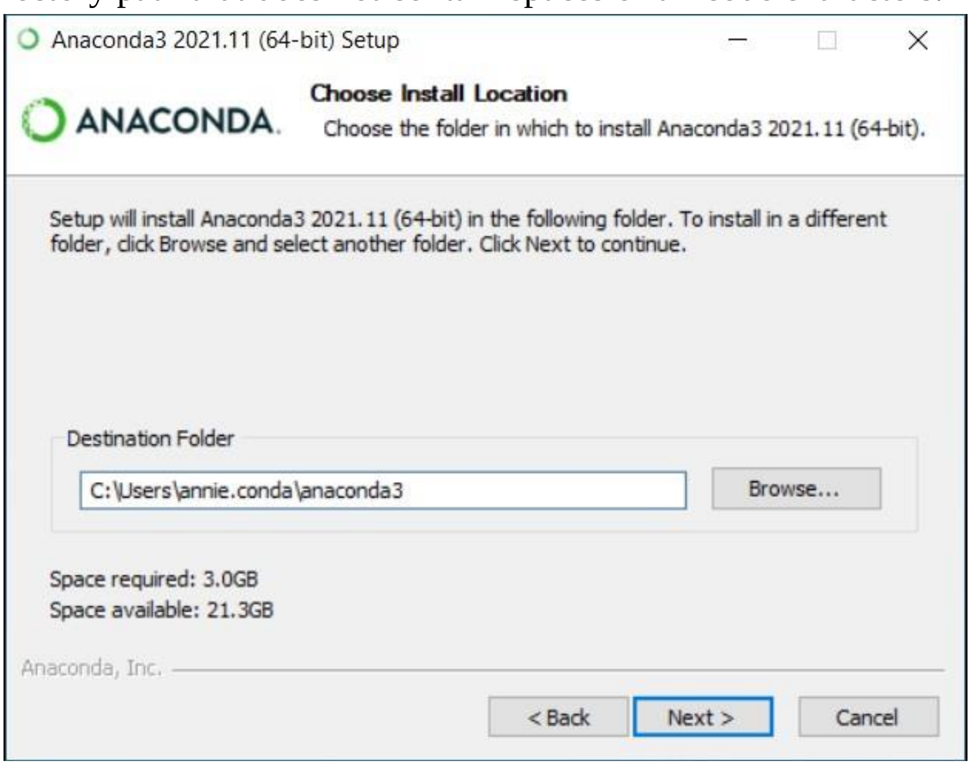
- Click **Next**.
- Read the licensing terms and click **I Agree**.



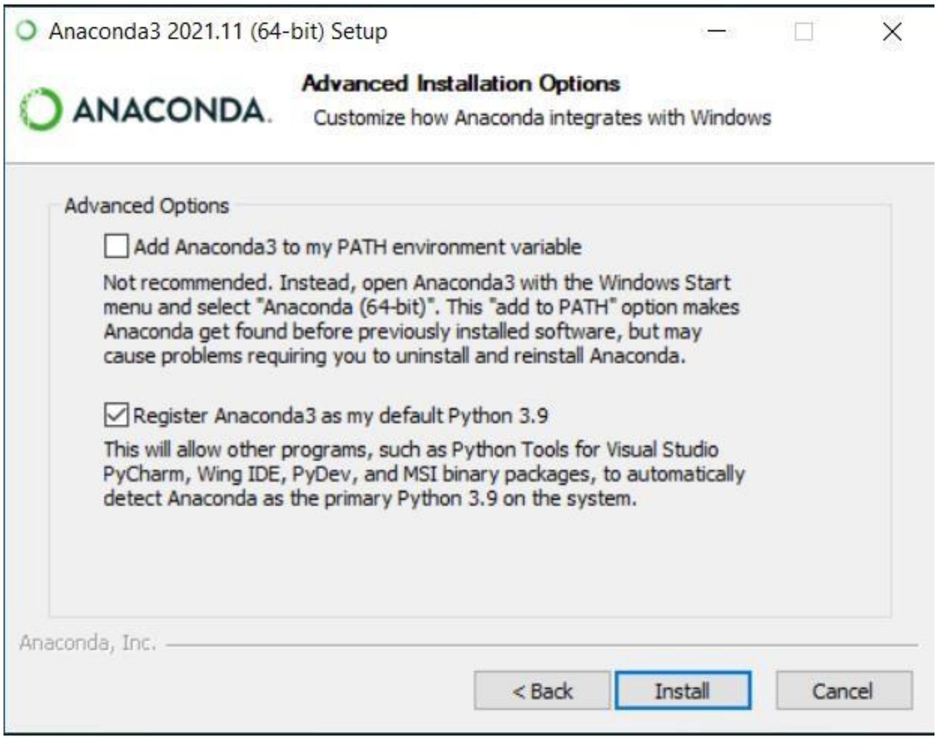
- It is recommended that you install for **Just Me**, which will install Anaconda Distribution to just the current user account.



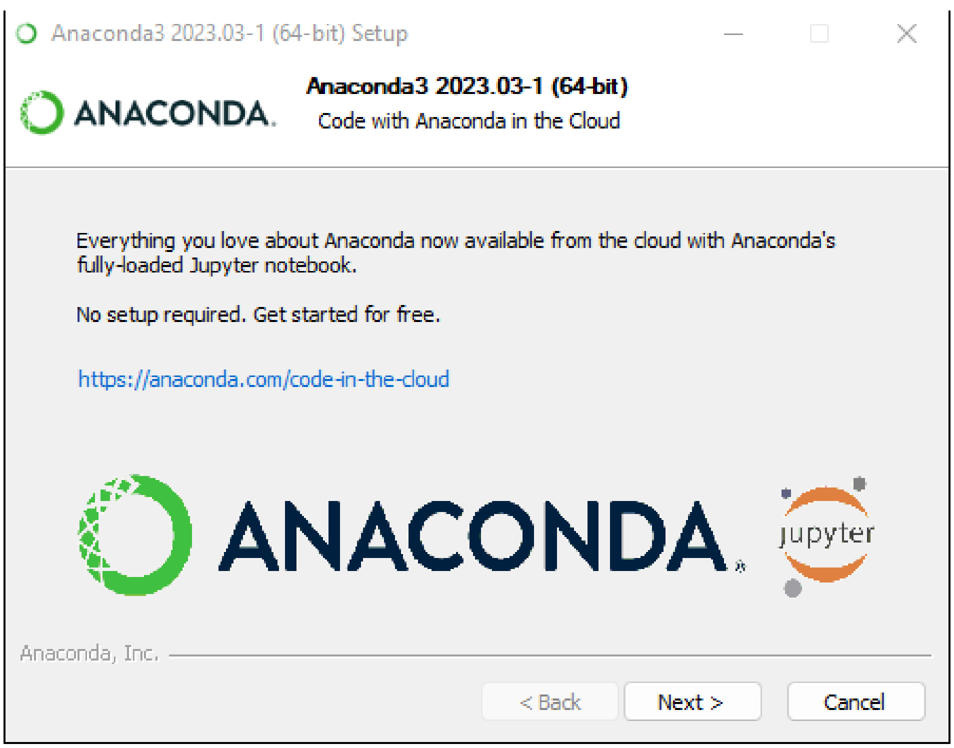
- Click Next.
- Select a destination folder to install Anaconda and click Next. Install Anaconda to a directory path that does not contain spaces or unicode characters.



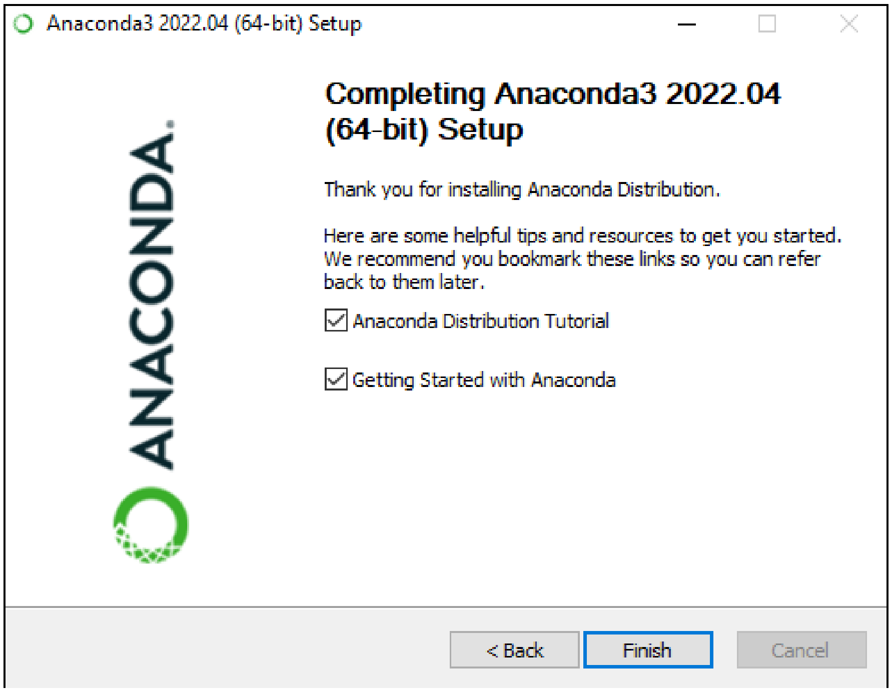
- Choose whether to add Anaconda to your PATH environment variable or register Anaconda as your default Python.
- Click **Install**. If you want to watch the packages Anaconda is installing, click Show Details.



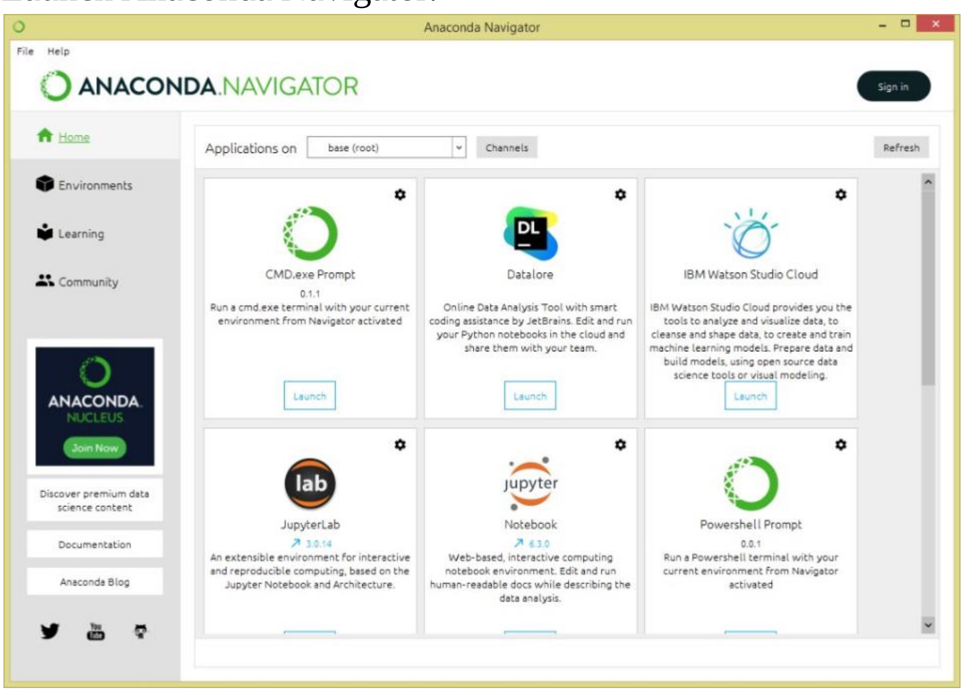
- Click on next.



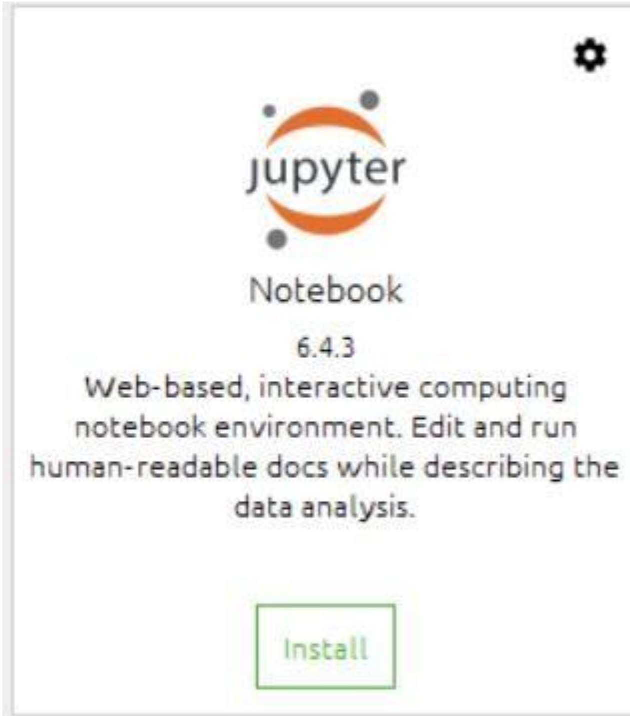
- After a successful installation you will see the "Thanks for installing Anaconda" dialog box:



- Install Jupyter Notebook with pip: pip install notebook
- To run the notebook: Jupyter notebook
- Launch Anaconda Navigator:



➢ Click on the Install Jupyter Notebook Button



➢ Beginning the Installation

➢ Loading Packages

➢ Finished Installation

➢ Launching Jupyter

PRACTICAL 1B

AIM: To Perform numerical analysis and data manipulation using numpy arrays.

CONCEPT:

- NumPy stands for Numerical Python. It is a Python library used for working with an array.
- In Python, we use the list for the array but it’s slow to process.
- NumPy array is a powerful N-dimensional array object and is used in linear algebra, Fourier transform, and random number capabilities.
- It provides an array object much faster than traditional Python lists.

LIBRARIES USED:

- NumPy Library

CODE:

```
#Creating a Numpy Array
import numpy as np
ar1=np.array([1,2,3])
print(ar1)
#ACCESS AND MANIPULATE ELEMENTS IN THE ARRAY.
ar1[1]=4
print(ar1)
#CREATE A 2-DIMENSIONAL ARRAY AND CHECK THE SHAPE OF THE ARRAY.
d=np.array([[1,2,3],[7,8,9]])
print(d)
print(d.shape)
#ACCESS ELEMENTS FROM THE 2D ARRAY USING INDEX POSITIONS.
print(d[0][0])
print(d[0][1])
print(d[0][2])
print(d[1][0])
#CREATE AN ARRAY OF TYPE STRING
import numpy as np
ar=np.array(['fi','hi','ji'])
print(ar)
```

#USING THE ARANGE() AND LINSPACE() FUNCTION TO EVENLY SPACE VALUES IN A SPECIFIED INTERVAL.
f=np.arange(0,20,2)
print(f)
t=np.linspace(0,10,30)
print(t)
#CREATE AN ARRAY OF RANDOM VALUES BETWEEN 0 AND 1 IN A GIVEN SHAPE.
arr=np.random.rand(10)
print(arr)
f=np.full((4,6),10)
print(f)
# CREATE AN IDENTITY MATRIX USING EYE()
c=np.eye(3)
print(c)
Arr=np.random.rand(5,5)
print(Arr)

OUTPUT:

```
[1 2 3]
```

```
[1 4 3]
```

```
[[1 2 3]
 [7 8 9]]
(2, 3)
```

```
1
2
3
7
```

```
['fi' 'hi' 'ji']
```

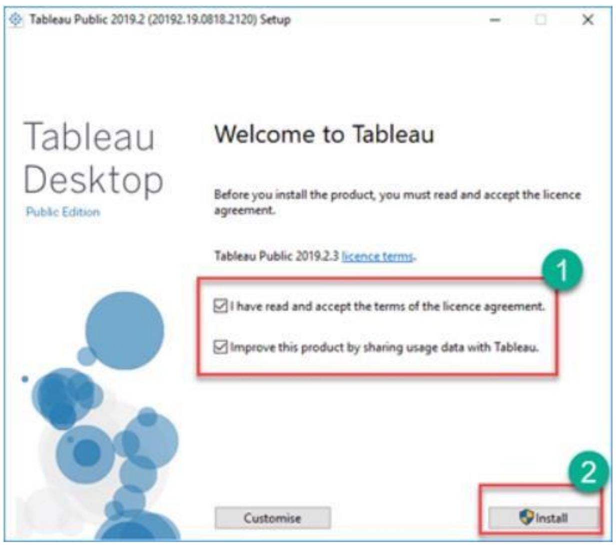
```
[ 0  2  4  6  8 10 12 14 16 18]
[ 0.          0.34482759  0.68965517  1.03448276  1.37931034  1.72413793
  2.06896552  2.4137931   2.75862069  3.10344828  3.44827586  3.79310345
  4.13793103  4.48275862  4.82758621  5.17241379  5.51724138  5.86206897
  6.20689655  6.55172414  6.89655172  7.24137931  7.5862069   7.93103448
  8.27586207  8.62068966  8.96551724  9.31034483  9.65517241 10.        ]
```

```
[0.34507896 0.25909741 0.64900139 0.05725924 0.71593438 0.30590996
 0.50309558 0.18233954 0.09586582 0.55605888]
[[10 10 10 10 10 10]
 [10 10 10 10 10 10]
 [10 10 10 10 10 10]
 [10 10 10 10 10 10]]
```

```
[[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]]
[[0.63058748 0.76764125 0.18235475 0.00125264 0.14834043]
 [0.20941241 0.96302275 0.26620106 0.69449178 0.81974778]
 [0.56747254 0.44076294 0.73796766 0.84213501 0.07015194]
 [0.23257825 0.90833474 0.6418751  0.03224962 0.26177425]
 [0.76679194 0.03869945 0.04286086 0.04005446 0.13961614]]
```

PRACTICAL 1C

Aim: Installation of Tableau.

- Go to https://www.tableau.com/products/desktop on your web browser.
- Click on the “TRY NOW” button shown in the top right corner of the website.
- It will redirect to the page where you need to enter your email id and click on “DOWNLOAD FREE TRIAL” button.
- This will start downloading tableau latest version. An .exe file for Windows is downloaded, and you can see the downloading process in the bottom left corner of the website.
- Open the downloaded file. Check in to accept the terms and conditions and click on “Install” button.
  - 

- A optional pop-up message will be shown to get the approval of Administrator to install the software. Click on “Yes” to approve it. Installation of the Tableau Desktop on Windows system starts
- Once the tableau desktop download and installation is completed, open the Tableau Desktop software.

PRACTICAL 2

AIM: To implement association mining algorithm of big data analysis (using apriori algorithm)

CONCEPT:

- **Identifying Frequent Itemsets:** The algorithm begins by scanning the dataset to identify individual items (1-item) and their frequencies. It then establishes a minimum support threshold, which determines whether an itemset is considered frequent.
- **Creating Possible item group:** Once frequent 1-itemgroup(single items) are identified, the algorithm generates candidate 2-itemgroup by combining frequent items. This process continues iteratively, forming larger itemsets (k-itemgroup) until no more frequent itemgroup can be found.
- **Removing Infrequent Item groups:** The algorithm employs a pruning technique based on the Apriori Property, which states that if an itemset is infrequent, all its supersets must also be infrequent. This significantly reduces the number of combinations that need to be evaluated.
- **Generating Association Rules:** After identifying frequent itemsets, the algorithm generates association rules that illustrate how items relate to one another, using metrics like support, confidence, and lift to evaluate the strength of these relationships.

LIBRARIES USED:

- Numpy
- Pandas
- Seaborn
- Matplotlib
- Apyori

CODE:

```
pip install apyori
from apyori import apriori
import pandas as pd
df=pd.read_csv("Groceries_dataset - Groceries_dataset.csv")
df.head()
```

```
print(df)
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
df=pd.read_csv("Groceries_dataset - Groceries_dataset.csv")
print(df)
df.info()
df['Date'] = pd.to_datetime(df['Date'])
df.info()
df.isnull().sum()
df['itemDescription'].value_counts().head()
member_shopping_frequency=
df.groupby('Member_number')['Date'].count().sort_values(ascending=False)
print(member_shopping_frequency)
sns.distplot(member_shopping_frequency, bins=8, kde=False, color='skyblue')
plt.xlabel('Number of purchasing')
plt.ylabel('Number of Member')
plt.title('member_shopping_frequency')
plt.show()
df['Month'] = df['Date'].dt.month
monthly_purchase_counts = df['Month'].value_counts().sort_index()
plt.figure(figsize=(10, 6))
sns.barplot(x=monthly_purchase_counts.index,
y=monthly_purchase_counts.values, color='skyblue')
plt.xlabel('month')
plt.ylabel('purchase_count')
plt.title('Purchase_number_count')
plt.show()
items=df.groupby('itemDescription').size().reset_index(name='frequancy_of_item')
.sort_values(by='frequancy_of_item',ascending=False)
items
top_items = items.head(10)
plt.figure(figsize=(10, 6))
sns.barplot(x='itemDescription',y='frequancy_of_item',data=top_items,order=top_i
tems['itemDescription'],color='skyblue')
plt.xticks(rotation=45, ha='right')
plt.title('Top 10 Items Frequency Box Plot')
plt.show()
df.head()
```

```
df = df.drop(['Month'],axis = 1)
df
group_same_customer = df.sort_values(by = 'Member_number',ascending = True)
group_same_customer['itemDescription'].str.strip()
transactions_list = []
for _, group in group_same_customer.groupby('Member_number'):
   customer_purchases = group['itemDescription'].tolist()
   transactions_list.append(customer_purchases)
get_ipython().system('pip install apyori')
group
association_rules=apriori(transactions_list, min_support=0.001,
min_confidence=0.05,  min_lift=4, min_length=2,max_length = 2)
rules_list = list(association_rules)
rules_list
def apriori_df(results):
    extracted_data = []
    for result in results:
        items_base = tuple(result.ordered_statistics[0].items_base)[0]
        items_add = tuple(result.ordered_statistics[0].items_add)[0]
        support = result.support
        confidence = result.ordered_statistics[0].confidence
        lift = result.ordered_statistics[0].lift
        extracted_data.append((items_base, items_add, support, confidence, lift))
    return extracted_data
resultsinDataFrame = pd.DataFrame(apriori_df(rules_list), columns = ['antecedent',
'consequent', 'Support', 'Confidence', 'Lift'])
resultsinDataFrame
```

OUTPUT:

	Member_number	Date	itemDescription
0	1808	21-07-2015	tropical fruit
1	2552	05-01-2015	whole milk
2	2300	19-09-2015	pip fruit
3	1187	12-12-2015	other vegetables
4	3037	01-02-2015	whole milk

```
       Member_number        Date      itemDescription
0               1808  21-07-2015       tropical fruit
1               2552  05-01-2015           whole milk
2               2300  19-09-2015            pip fruit
3               1187  12-12-2015     other vegetables
4               3037  01-02-2015           whole milk
...              ...         ...                  ...
38760           4471  08-10-2014        sliced cheese
38761           2022  23-02-2014                candy
38762           1097  16-04-2014             cake bar
38763           1510  03-12-2014  fruit/vegetable juice
38764           1521  26-12-2014             cat food

[38765 rows x 3 columns]
```

```
       Member_number        Date      itemDescription
0               1808  21-07-2015       tropical fruit
1               2552  05-01-2015           whole milk
2               2300  19-09-2015            pip fruit
3               1187  12-12-2015     other vegetables
4               3037  01-02-2015           whole milk
...              ...         ...                  ...
38760           4471  08-10-2014        sliced cheese
38761           2022  23-02-2014                candy
38762           1097  16-04-2014             cake bar
38763           1510  03-12-2014  fruit/vegetable juice
38764           1521  26-12-2014             cat food

[38765 rows x 3 columns]
```

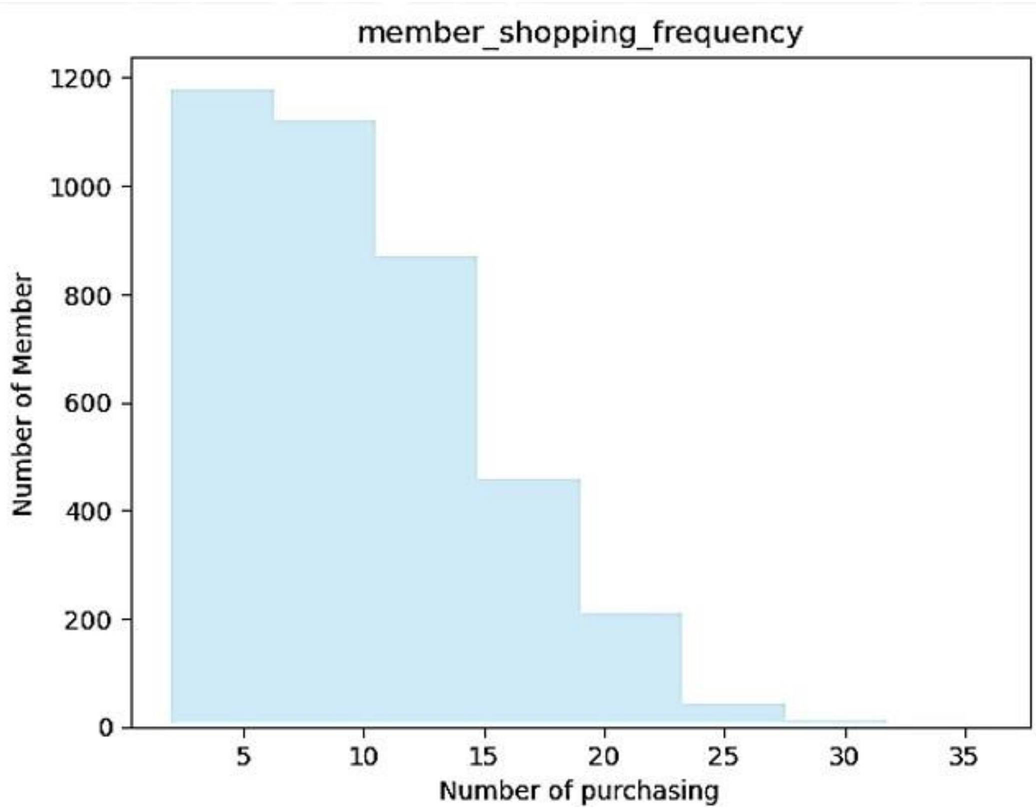
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 38765 entries, 0 to 38764
Data columns (total 3 columns):
 #   Column           Non-Null Count  Dtype 
---  ------           --------------  ----- 
 0   Member_number    38765 non-null  int64 
 1   Date             38765 non-null  object
 2   itemDescription  38765 non-null  object
dtypes: int64(1), object(2)
memory usage: 908.7+ KB
```

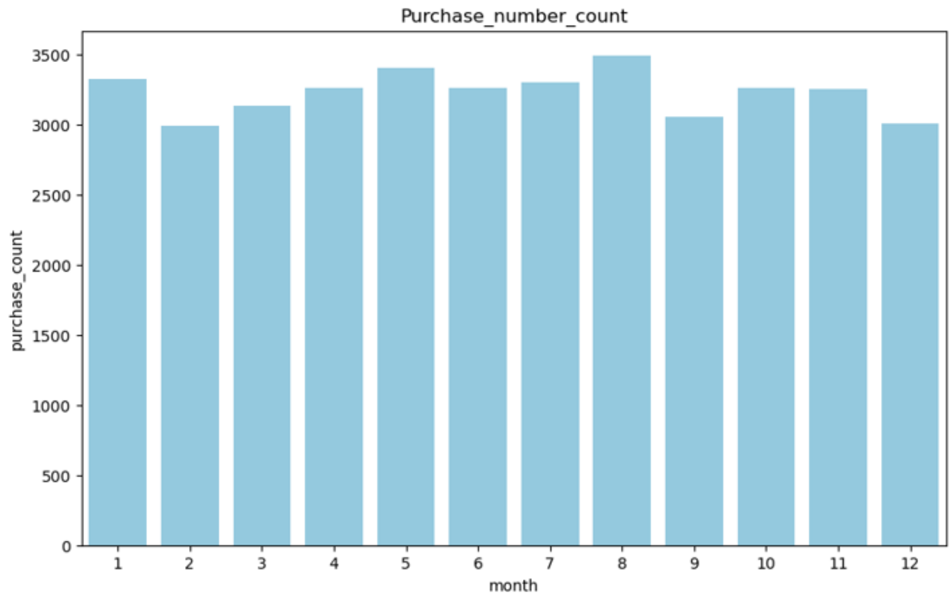
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 38765 entries, 0 to 38764
Data columns (total 3 columns):
 #   Column           Non-Null Count  Dtype         
---  ------           --------------  -----         
 0   Member_number    38765 non-null  int64         
 1   Date             38765 non-null  datetime64[ns]
 2   itemDescription  38765 non-null  object        
dtypes: datetime64[ns](1), int64(1), object(1)
memory usage: 908.7+ KB
```

```
Member_number      0
Date               0
itemDescription    0
dtype: int64
```

```
itemDescription
whole milk          2502
other vegetables    1898
rolls/buns          1716
soda                1514
yogurt              1334
Name: count, dtype: int64
```

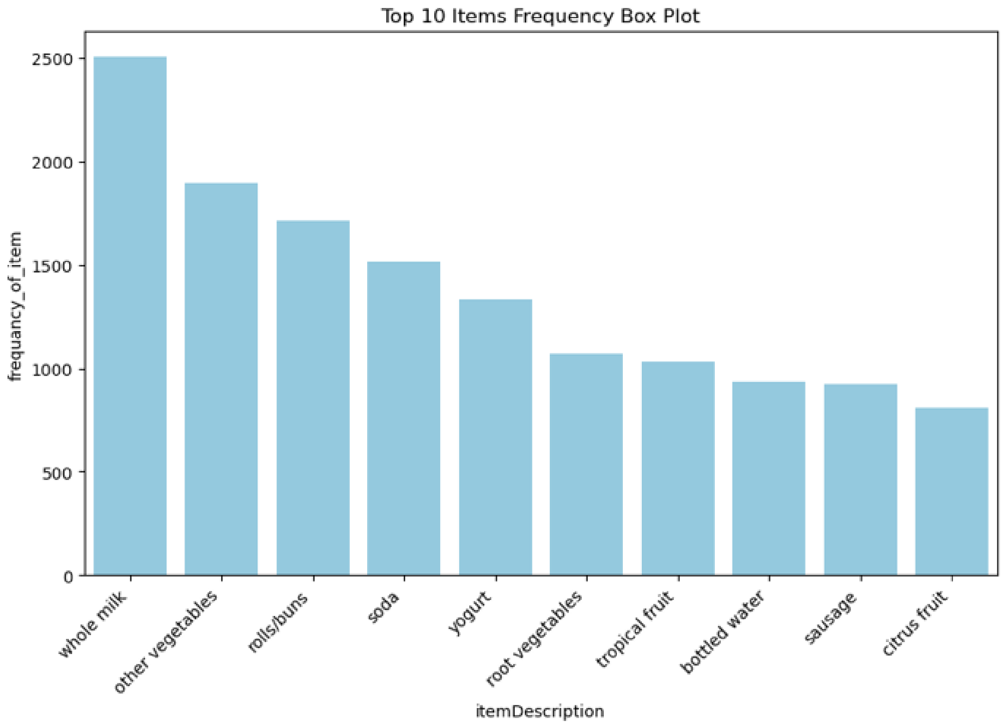
```
Member_number
3180    36
3737    33
3050    33
2051    33
3915    31
        ..
3533     2
2302     2
4824     2
1084     2
3377     2
Name: Date, Length: 3898, dtype: int64
```





	itemDescription	frequancy_of_item
164	whole milk	2502
102	other vegetables	1898
122	rolls/buns	1716
138	soda	1514
165	yogurt	1334
...	...	...
124	rubbing alcohol	5
5	bags	4
4	baby cosmetics	3
114	preservation products	1
79	kitchen utensil	1

167 rows × 2 columns



	Member_number	Date	itemDescription
0	1808	2015-07-21	tropical fruit
1	2552	2015-01-05	whole milk
2	2300	2015-09-19	pip fruit
3	1187	2015-12-12	other vegetables
4	3037	2015-02-01	whole milk
...	...	...	...
38760	4471	2014-10-08	sliced cheese
38761	2022	2014-02-23	candy
38762	1097	2014-04-16	cake bar
38763	1510	2014-12-03	fruit/vegetable juice
38764	1521	2014-12-26	cat food

38765 rows × 3 columns

```
1629                    soda
13331             whole milk
8395              whole milk
4843                 sausage
17778      pickled vegetables
                ...
34885     semi-finished bread
25489        other vegetables
9340             bottled beer
27877                  onions
3578                     soda
Name: itemDescription, Length: 38765, dtype: object
```

```
Defaulting to user installation because normal site-packages is not writeable
```

	Member_number	Date	itemDescription
19727	5000	2015-02-10	root vegetables
11728	5000	2014-03-09	fruit/vegetable juice
34885	5000	2015-02-10	semi-finished bread
25489	5000	2014-11-16	other vegetables
9340	5000	2014-11-16	bottled beer
27877	5000	2014-03-09	onions
3578	5000	2015-02-10	soda

	antecedent	consequent	Support	Confidence	Lift
0	Instant food products	soups	0.001026	0.066667	5.413889
1	canned fruit	soft cheese	0.001026	0.190476	5.050858
2	rum	canned vegetables	0.001026	0.125000	6.090625
3	tea	cat food	0.001796	0.259259	5.909898
4	ready soups	chewing gum	0.001026	0.266667	5.973946
5	cookware	ice cream	0.001026	0.235294	4.168984
6	curd cheese	mustard	0.001283	0.108696	4.655996
7	decalcifier	dessert	0.001026	0.444444	5.140785
8	dental care	seasonal products	0.001026	0.121212	4.678068
9	dog food	frozen potato products	0.001283	0.074627	4.040216
10	female sanitary products	pot plants	0.001283	0.125000	4.200431
11	hair spray	frozen vegetables	0.001026	0.444444	4.331111
12	light bulbs	hard cheese	0.001539	0.214286	4.015797
13	tea	herbs	0.001283	0.185185	4.717986
14	jam	rice	0.001026	0.117647	9.358944
15	liver loaf	photo/film	0.001026	0.083333	4.218615
16	ready soups	oil	0.001283	0.333333	5.987711
17	soups	seasonal products	0.001283	0.104167	4.020215

PRACTICAL 3

AIM: To implement linear regression algorithm of big data analysis.

CONCEPT:

- Linear regression is also a type of supervised machine-learning algorithm that learns from the labelled datasets and maps the data points with most optimized linear functions which can be used for prediction on new datasets.
- It computes the linear relationship between the dependent variable and one or more independent features by fitting a linear equation with observed data.
- It predicts the continuous output variables based on the independent input variable.
- For example if we want to predict house price we consider various factor such as house age, distance from the main road, location, area and number of room, linear regression uses all these parameter to predict house price as it consider a linear relation between all these features and price of house.

LIBRARIES USED:

- NumPy
- Panda
- Sklear
- Matplot

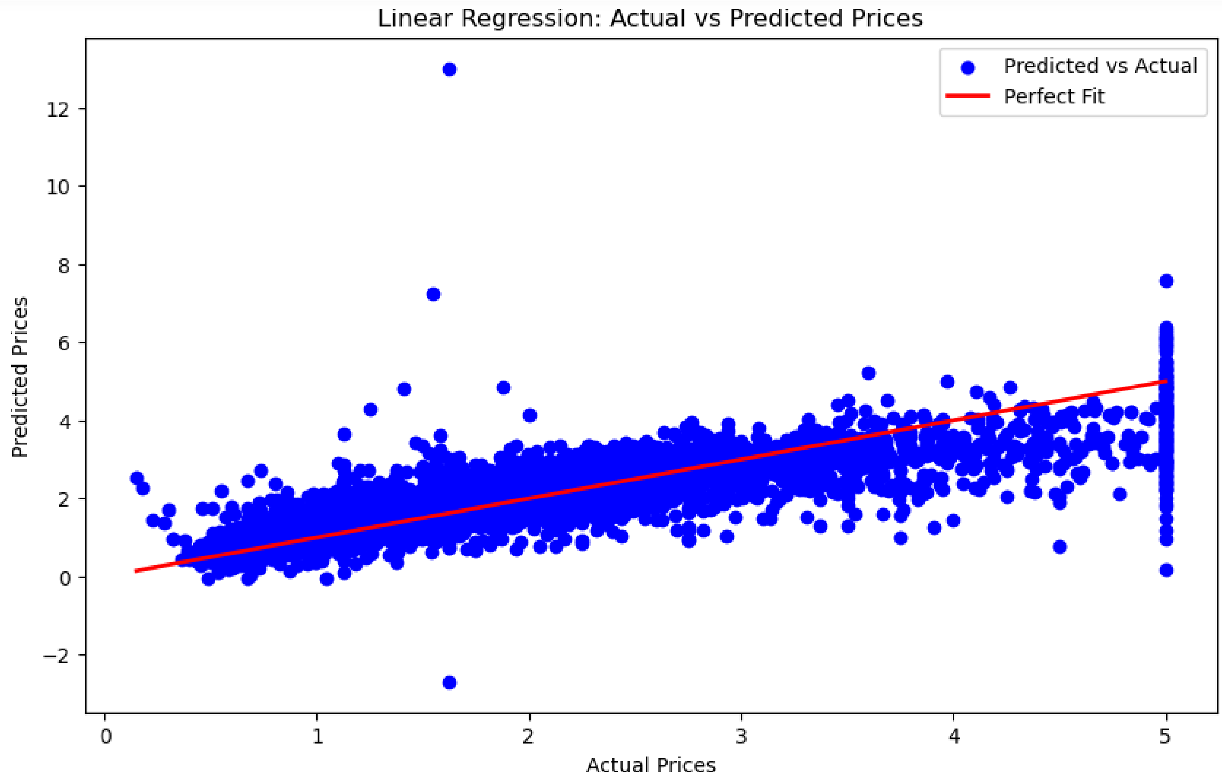
CODE:

```
import numpy as np
import pandas as pd
from sklearn.datasets import fetch_california_housing
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import StandardScaler, PolynomialFeatures
from sklearn.metrics import mean_squared_error, r2_score
import matplotlib.pyplot as plt
california_housing = fetch_california_housing()
X=pd.DataFrame(california_housing.data,columns=california_housing.feature_name
s)
y = california_housing.target
```

```
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
poly = PolynomialFeatures(degree=2, include_bias=False)
X_poly = poly.fit_transform(X_scaled)
X_train, X_test, y_train, y_test = train_test_split(X_poly, y, test_size=0.2,
random_state=42)
linear_model = LinearRegression()
linear_model.fit(X_train, y_train)
y_pred = linear_model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test,y_pred)
cv_scores = cross_val_score(linear_model, X_poly, y, cv=5,
scoring='neg_mean_squared_error')
cv_mse = -cv_scores.mean()
print(f"Linear Regression Mean Squared Error (MSE) : {mse:.4f}")
print(f"Linear Regression R-Squared: {r2:.4f}")
print(f"Cross-validation MSE: {cv_mse:.4f}")
plt.figure(figsize=(10,6))
plt.scatter(y_test, y_pred, color='blue', label='Predicted vs Actual')
plt.plot([min(y_test), max(y_test)], [min(y_test), max(y_test)], color='red', lw=2,
label='Perfect Fit')
plt.xlabel('Actual Prices')
plt.ylabel('Predicted Prices')
plt.title('Linear Regression: Actual vs Predicted Prices')
plt.legend()
plt.show()
```

OUTPUT:

```
Linear Regression Mean Squared Error (MSE) : 0.4643
Linear Regression R-Squared: 0.6457
Cross-validation MSE: 37.9482
```


Linear Regression: Actual vs Predicted Prices
Predicted vs Actual
Perfect Fit
Predicted Prices
Actual Prices
12
10
8
6
4
2
0
−2
0
1
2
3
4
5

PRACTICAL 4

AIM: To implement a logistic regression in big data analysis.

CONCEPT:

- Logistic regression is used for binary classification where we use sigmoid function, that takes input as independent variables and produces a probability value between 0 and 1.
- Logistic regression predicts the output of a categorical dependent variable. Therefore, the outcome must be a categorical or discrete value.
- It can be either Yes or No, 0 or 1, true or False, etc. but instead of giving the exact value as 0 and 1, it gives the probabilistic values which lie between 0 and 1.
- In Logistic regression, instead of fitting a regression line, we fit an “S” shaped logistic function, which predicts two maximum values (0 or 1).

LIBRARIES USED:

- NumPy
- Panda
- Sklearn

CODE:

```
import pandas as pd
import numpy as np
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
iris = load_iris()
data = pd.DataFrame(data=iris.data, columns=iris.feature_names)
data['species'] = iris.target
print(data.head())
print(data.shape)
print(data.info())
print(data.describe())
print(data['species'].value_counts())
```

```
X = iris.data
y = iris.target
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42,
stratify=y)
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
model = LogisticRegression(multi_class='multinomial', solver='lbfgs', max_iter=200)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.2f}")
conf_matrix = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:\n", conf_matrix)
print("Classification Report:\n", classification_report(y_test, y_pred))
```

OUTPUT:

```
   sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)  \
0                5.1               3.5                1.4               0.2
1                4.9               3.0                1.4               0.2
2                4.7               3.2                1.3               0.2
3                4.6               3.1                1.5               0.2
4                5.0               3.6                1.4               0.2

   species
0        0
1        0
2        0
3        0
4        0
(150, 5)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   sepal length (cm)  150 non-null    float64
 1   sepal width (cm)   150 non-null    float64
 2   petal length (cm)  150 non-null    float64
 3   petal width (cm)   150 non-null    float64
 4   species            150 non-null    int64
dtypes: float64(4), int64(1)
memory usage: 6.0 KB
None
```

```
       sepal length (cm)  sepal width (cm)  petal length (cm)  \
count         150.000000        150.000000         150.000000
mean            5.843333          3.057333           3.758000
std             0.828066          0.435866           1.765298
min             4.300000          2.000000           1.000000
25%             5.100000          2.800000           1.600000
50%             5.800000          3.000000           4.350000
75%             6.400000          3.300000           5.100000
max             7.900000          4.400000           6.900000

       petal width (cm)     species
count        150.000000  150.000000
mean           1.199333    1.000000
std            0.762238    0.819232
min            0.100000    0.000000
25%            0.300000    0.000000
50%            1.300000    1.000000
75%            1.800000    2.000000
max            2.500000    2.000000
species
0    50
1    50
2    50
Name: count, dtype: int64
Accuracy: 0.91
Confusion Matrix:
 [[15  0  0]
 [ 0 14  1]
 [ 0  3 12]]
Classification Report:
               precision    recall  f1-score   support

           0       1.00      1.00      1.00        15
           1       0.82      0.93      0.88        15
           2       0.92      0.80      0.86        15

    accuracy                           0.91        45
   macro avg       0.92      0.91      0.91        45
weighted avg       0.92      0.91      0.91        45
```

PRACTICAL 5

AIM: To create a decision tree classifier and visualize it graphically.

CONCEPT:

- A decision tree is a graphical representation of different options for solving a problem and show how different factors are related.
- It has a hierarchical tree structure starts with one main question at the top called a node which further branches out into different possible outcomes where:
- Root Node is the starting point that represents the entire dataset.
- Branches: These are the lines that connect nodes. It shows the flow from one decision to another.
- Internal Nodes are Points where decisions are made based on the input features.
- Leaf Nodes: These are the terminal nodes at the end of branches that represent final outcomes or predictions

LIBRARIES USED:

Following libraries used in this code for implementation:

- Pandas
- Numpy
- Pyplot
- Seaborn
- Sklearn

CODE:

```
import os
import warnings
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.metrics import classification_report, confusion_matrix
warnings.filterwarnings('ignore')
df = sns.load_dataset('iris')
```

```
le = LabelEncoder()
target = le.fit_transform(df['species'])
X = df.drop('species',axis=1)
y = target
X_train,X_test,y_train,y_test =train_test_split(X,y,test_size=0.4,random_state=42)
def evaluate_model(model, X_train, y_train, X_test, y_test):
    train_accuracy = model.score(X_train, y_train)
    test_accuracy = model.score(X_test, y_test)
    print(f"Training Accuracy: {train_accuracy:.4f}")
    print(f"Testing Accuracy: {test_accuracy:.4f}")
    if train_accuracy > test_accuracy:
        print("Warning: Potential Overfitting Detected!")
dtree = DecisionTreeClassifier(
    random_state = 42,
    max_depth = 3,
    min_samples_split = 4,
    min_samples_leaf = 2
)
dtree.fit(X_train,y_train)
print("\n ------Overfitting Check------ ")
evaluate_model(dtree,X_train,y_train,X_test,y_test)
cv_scores = cross_val_score(dtree, X, y, cv=5)
print("\nCross-validation scores:", cv_scores)
print(f"Mean cross-validation score: {cv_scores.mean():.4f}")
cv_scores = cross_val_score(dtree, X, y, cv=5)
print("\nCross-validation scores:", cv_scores)
print(f"Mean cross-validation score: {cv_scores.mean():.4f}")
y_pred = dtree.predict(X_test)
print("\n Classification Report:\n",classification_report(y_test,y_pred))
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(5, 5))
sns.heatmap(data=cm, annot=True, square=True, cmap='Blues', fmt='d')
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
plt.title(f'Confusion Matrix\nAccuracy Score: {dtree.score(X_test, y_test):.4f}')
plt.show()
plt.figure(figsize=(20, 20))
plot_tree(
    decision_tree=dtree,
    feature_names=["sepal_length", "sepal_width", "petal_length", "petal_width"],
```

```
    class_names=["setosa", "versicolor", "virginica"],
    filled=True,
    precision=4,
    rounded=True
)
plt.show()
```

OUTPUT:

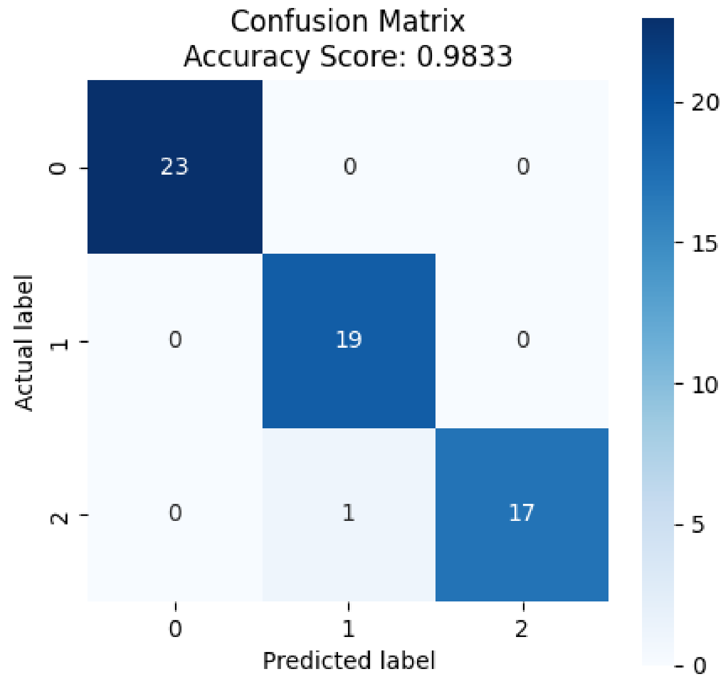
```
 ------Overfitting Check------
Training Accuracy: 0.9667
Testing Accuracy: 0.9833
```

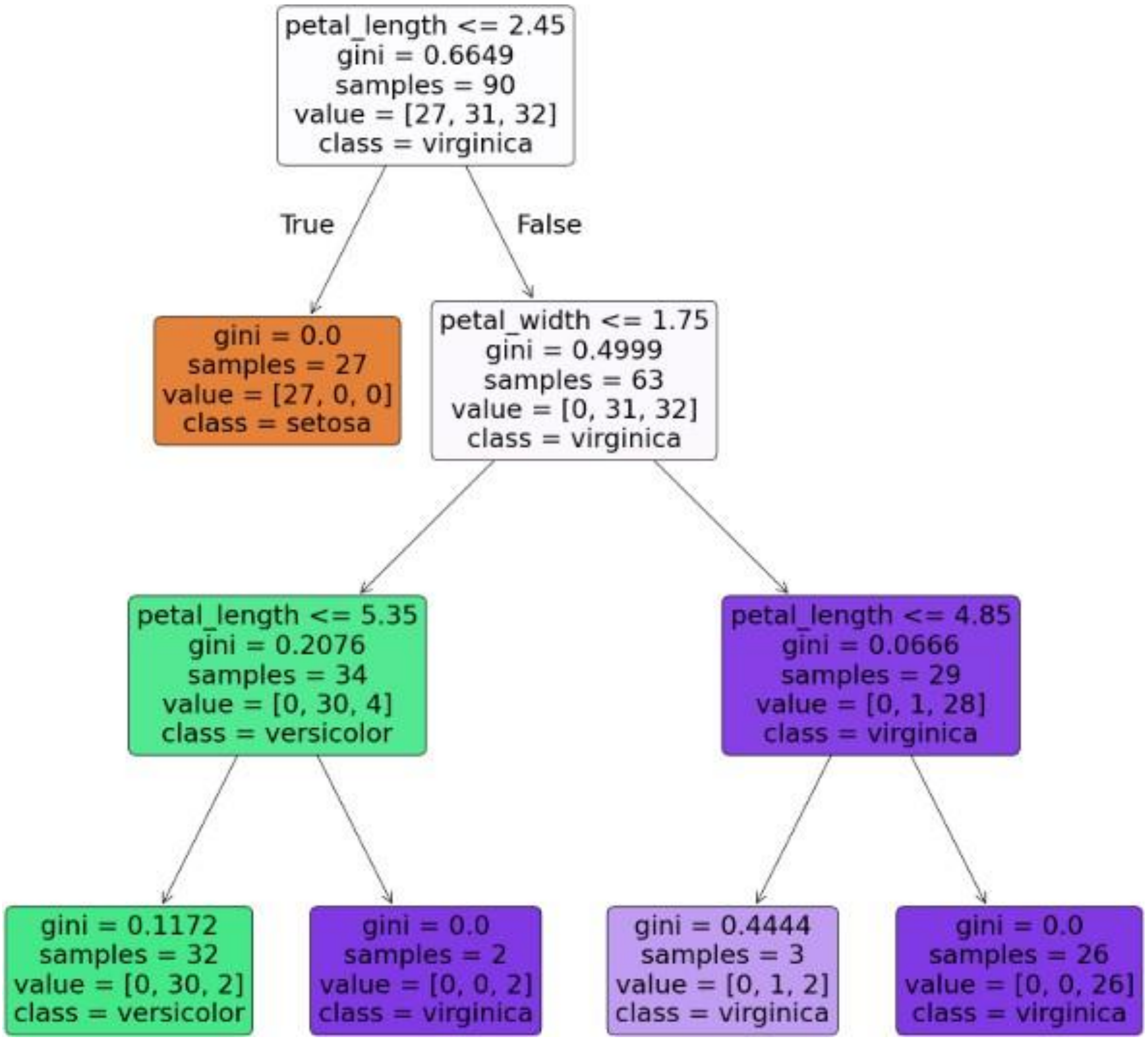
```
Cross-validation scores: [0.96666667 0.96666667 0.93333333 0.93333333 1.
Mean cross-validation score: 0.9600

  Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        23
           1       0.95      1.00      0.97        19
           2       1.00      0.94      0.97        18

    accuracy                           0.98        60
   macro avg       0.98      0.98      0.98        60
weighted avg       0.98      0.98      0.98        60
```




petal_length <= 2.45
gini = 0.6649
samples = 90
value = [27, 31, 32]
class = virginica
True
False
gini = 0.0
samples = 27
value = [27, 0, 0]
class = setosa
petal_width <= 1.75
gini = 0.4999
samples = 63
value = [0, 31, 32]
class = virginica
petal_length <= 5.35
gini = 0.2076
samples = 34
value = [0, 30, 4]
class = versicolor
petal_length <= 4.85
gini = 0.0666
samples = 29
value = [0, 1, 28]
class = virginica
gini = 0.1172
samples = 32
value = [0, 30, 2]
class = versicolor
gini = 0.0
samples = 2
value = [0, 0, 2]
class = virginica
gini = 0.4444
samples = 3
value = [0, 1, 2]
class = virginica
gini = 0.0
samples = 26
value = [0, 0, 26]
class = virginica