Q 1. Pandas Library

a) Write a python program to implement Pandas Series with labels.

b) Create a Pandas Series from a dictionary.

c) Creating a Pandas Data Frame.

d) Write a program

which makes use of the following Pandas methods

i) describe ()

ii) head ()

iii) tail ()

iv) info ()

Sub Write) .UNIT-I (Evolution of Machine Learning) from TextBook

Data Set or Data Input:

data = {

"Name" : ["A SADWIK" , "E SOWMYA" , "G SUBBAREDDY", "M SUNAYANA" , " S ASIF " , " S SHAMEERA ", "U TULASI"],

“Roll_Number” : [" 24HN1A3901" , "24HN1A3911" , " 24HN1A3921", "24HN1A3931" , "24HN1A3941" , " 24HN1A3951" , "24HN1A3961"],

“ML_subject_code” :[“ 23A31401T”, “ 23A31401T”, “ 23A31401T”, “ 23A31401T”, “ 23A31401T”, “ 23A31401T”, “ 23A31401T”, “ 23A31401T”],

“AI_ML_Lab_code” : [“ 23A31403”, “ 23A31403”, “ 23A31403”, “ 23A31403”, “ 23A31403”, “ 23A31403”, “ 23A31403”]}

A)

```
import pandas as pd


data = {
 "Name" : ["A SADWIK" , "E SOWMYA" , "G SUBBAREDDY", "M SUNAYANA" , " S ASIF " , " S SHAMEERA ", "U TULASI"],
"Roll_Number" : [ ' 24HN1A3901' , "24HN1A3911" , " 24HN1A3921", "24HN1A3931" , "24HN1A3941" , " 24HN1A3951" , "24HN1A3961"],
```

```
"ML_subject_code" :[" 23A31401T", " 23A31401T", "23A31401T", " 23A31401T",
"23A31401T", " 23A31401T", " 23A31401T"],
"AI_ML_Lab_code" : ["23A31403", "23A31403", "23A31403", "23A31403", "23A31403",
"23A31403", " 23A31403"]}

df = pd.DataFrame(data)
print(df)
# i) describe()
print("\nDescribe:")
print(df.describe())

# ii) head()
print("\nHead:")
print(df.head(3))

# iii) tail()
print("\nTail:")
print(df.tail(3))

# iv) info()
print("\nInfo:")
print(df.info())
```

Output:

```
Describe:
            Name  Roll_Number ML_subject_code AI_ML_Lab_code
count          7            7               7              7
unique         7            7               2              2
top     A SADWIK   24HN1A3901       23A31401T       23A31403
freq           1            1               5              6

Head:
           Name  Roll_Number ML_subject_code AI_ML_Lab_code
0      A SADWIK   24HN1A3901       23A31401T       23A31403
1      E SOWMYA   24HN1A3911       23A31401T       23A31403
2  G SUBBAREDDY   24HN1A3921       23A31401T       23A31403

Tail:
           Name  Roll_Number ML_subject_code AI_ML_Lab_code
4       S ASIF    24HN1A3941       23A31401T       23A31403
5   S SHAMEERA    24HN1A3951       23A31401T       23A31403
6      U TULASI   24HN1A3961       23A31401T       23A31403

Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7 entries, 0 to 6
Data columns (total 4 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   Name             7 non-null      object
 1   Roll_Number      7 non-null      object
 2   ML_subject_code  7 non-null      object
```

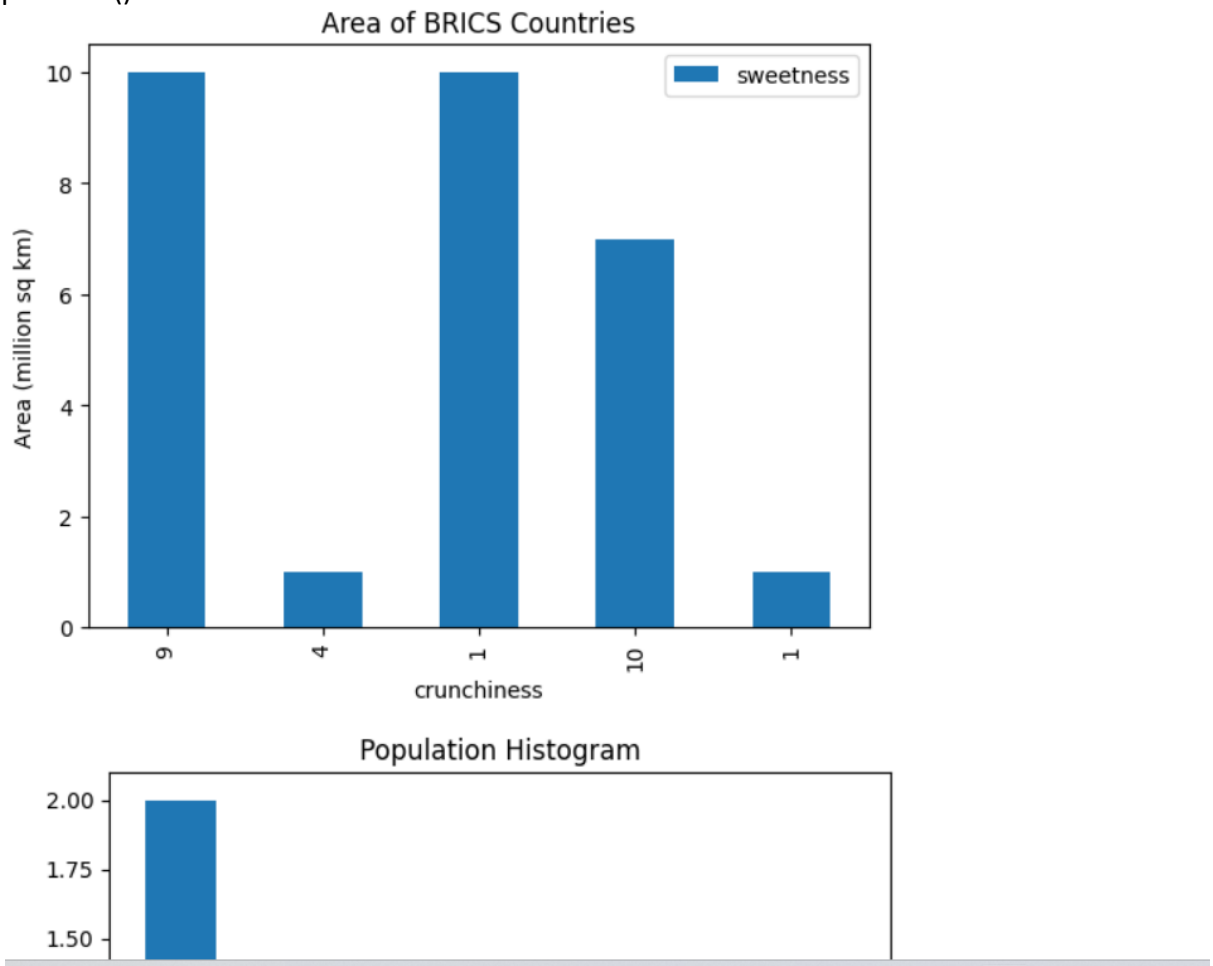
```
 3   AI_ML_Lab_code   7 non-null      object
dtypes: object(4)
memory usage: 356.0+ bytes
None
```

Q 2. Pandas Library: Visualization a) Write a program which use pandas inbuilt visualization to
plot following graphs:
i. Bar plots
ii. Histograms
iii. Line plots
iv. Scatter plots
Sub Write). UNIT-I (Paradigms for ML) from TextBook
Data Set or Data Input:
food_data = {
" ingredients " : ["apple","bacon","banana","carrot","cheese"],
"sweetness": [10, 1, 10, 7, 1],
"crunchiness":[9, 4, 1,10, 1],
"cat_type":["fruit","protein","fruit","vegetable","protein"]}

A)

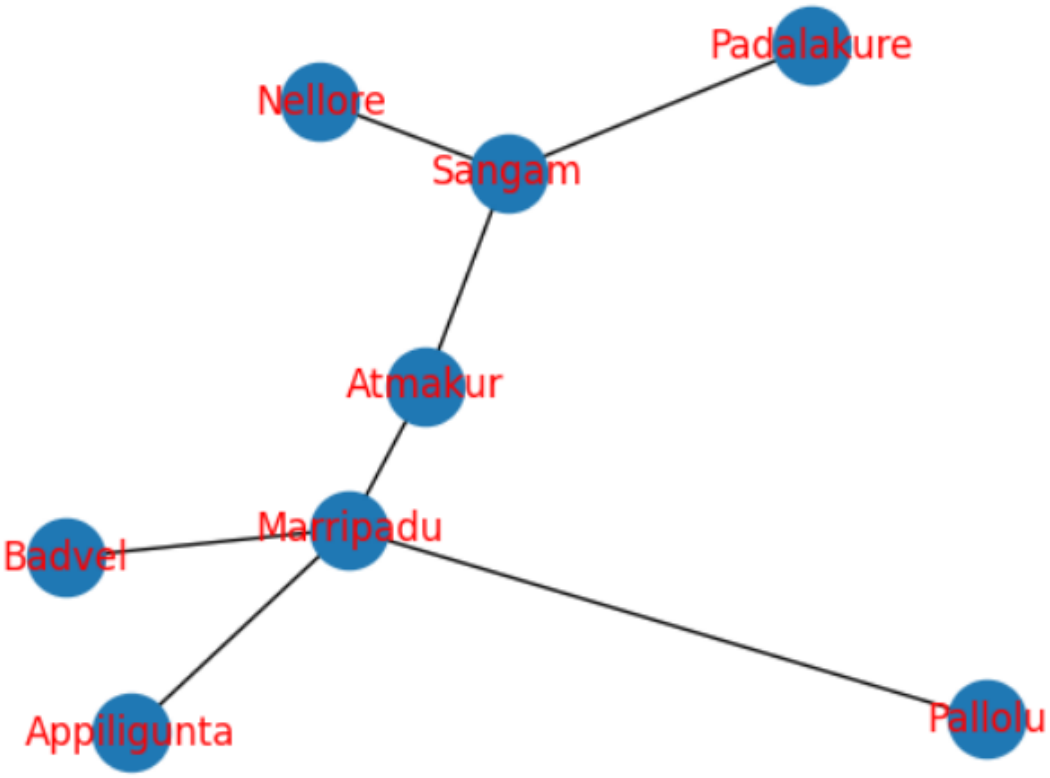
```
import pandas as pd
import matplotlib.pyplot as plt
food_data = {
" ingredients " : ["apple","bacon","banana","carrot","cheese"],
"sweetness": [10, 1, 10, 7, 1],
"crunchiness":[9, 4, 1,10, 1],
"cat_type":["fruit","protein","fruit","vegetable","protein"]}

brics = pd.DataFrame(food_data)

print(brics)

# ---------------------------
# 1. BAR PLOT (Area by Country)
# ---------------------------
brics.plot(kind='bar', x='crunchiness', y='sweetness', title='Area of BRICS Countries')
plt.ylabel("Area (million sq km)")
plt.show()

# ----------------------------------
# 2. HISTOGRAM (Population distribution)
# ----------------------------------
brics['crunchiness'].plot(kind='hist', title='Population Histogram')
plt.xlabel("Population (millions)")
plt.show()

# ---------------------------
```

```
# 3. LINE PLOT (Area trend)
# --------------------------
brics.plot(kind='line', x='crunchiness', y='sweetness', title='Area Line Plot')
plt.ylabel("Area (million sq km)")
plt.show()

# ----------------------------------------
# 4. SCATTER PLOT (Area vs Population)
# ----------------------------------------
brics.plot(kind='scatter', x='crunchiness', y='sweetness',
        title='Scatter Plot: Area vs Population')
plt.xlabel("Area (million sq km)")
plt.ylabel("Population (millions)")
plt.show()
```



Q)3. Write a Program to Implement Breadth First Search using Python.
Sub Write). UNIT-I (Learning by Deduction and Learning by Abduction) from TextBook
Data Set or Data Input:
[('Atmakur', 'Marripadu', 43), ('Atmakur', 'Sangam', 17), ('Marripadu', 'Appiligunta', 10), ('Marripadu', 'Badvel', 51), ('Marripadu', 'Pallolu', 5), ('Sangam', 'Padalakure', 20), ('Sangam', 'Nellore', 60)]
A) import networkx as nx

```
import matplotlib.pyplot as plt
G = nx.Graph()

G.add_weighted_edges_from([

 ('Atmakur', 'Marripadu', 43),
 ('Atmakur', 'Sangam', 17),
 ('Marripadu', 'Appiligunta', 10),
 ('Marripadu', 'Badvel', 51),
 ('Marripadu', 'Pallolu', 5),
 ('Sangam', 'Padalakure', 20),
 ('Sangam', 'Nellore', 60)

], color="red")

G
plt.axis('off')
nx.draw_networkx(G,node_size=600,font_color='red')
```




```
bfs_nodes = list(nx.bfs_tree(G, source='Atmakur'))
print("BFS Traversal:", bfs_nodes)
```

Output:

```
BFS Traversal: ['Atmakur', 'Marripadu', 'Sangam', 'Appiligunta', 'Badvel',
'Pallolu', 'Padalakure', 'Nellore']
```

Q)5. Write a Program to Implement Depth First Search using Python.
Sub Write). UNIT-I (Learning by Deduction and Learning by Abduction) from TextBook
Data Set or Data Input:
[('Atmakur', 'Marripadu', 43), ('Atmakur', 'Sangam', 17), ('Marripadu', 'Appiligunta', 10), ('Marripadu', 'Badvel', 51), ('Marripadu', 'Pallolu', 5), ('Sangam', 'Padalakure', 20), ('Sangam', 'Nellore', 60)]

A) dfs_nodes = list(nx.dfs_tree(G, source='Atmakur'))
print("DFS Traversal:", dfs_nodes)

```
DFS Traversal: ['Atmakur', 'Marripadu', 'Appiligunta', 'Badvel', 'Pallolu', 'Sangam', 'Padalakure', 'Nellore']
```

Q)
8. Apply the following Pre-processing techniques for a given dataset.
a. Attribute selection
b. Handling Missing Values
c. Discretization
d. Elimination of Outliers
Sub Write). UNIT-I (Feature Engineering) from TextBook
Data Set or Data Input:

```
student_data = {
 'roll_number':['901','911','921','931','941','951','961'],
 "Name" : ["A SADWIK" , "E SOWMYA" , "G SUBBAREDDY",
 "M SUNAYANA" , "S ASIF" , "S SHAMEERA", "U TULASI"],
 'Passed':[6,6,6,6,None,6,None],
 'Failed':[None,0,0,0,0,None,None],
 'Total':[6,6,6,6,6,6,6],
 'Branch':['AIML','AIML','AIML','AIML','AIML','AIML','AIML'],
 'Age':[22,20,21,20,23,19,None]
}
```

A)

```
import pandas as pd
student_data = {
 'roll_number':['901','911','921','931','941','951','961'],
 "Name" : ["A SADWIK" , "E SOWMYA" , "G SUBBAREDDY",
 "M SUNAYANA" , "S ASIF" , "S SHAMEERA", "U TULASI"],
 'Passed':[6,6,6,6,None,6,None],
 'Failed':[None,0,0,0,0,None,None],
 'Total':[6,6,6,6,6,6,6],
 'Branch':['AIML','AIML','AIML','AIML','AIML','AIML','AIML'],
 'Age':[22,20,21,20,23,19,None]
}

df = pd.DataFrame(student_data)
# df = pd.read_csv('df.csv')
print(df)
```

```
# a. Attribute selection
df1=df[['Age','Passed','Failed','Total','Age']]
df1
# b. Handling Missing Values
df['Age'].fillna(df['Age'].mean(),inplace=True)
df['Passed'].fillna(df['Passed'].mean(),inplace=True)
df['Failed'].fillna(df['Failed'].mean(),inplace=True)
df['Age'].fillna(df['Age'].mode(), inplace=True)
df
# c. Discretization
bins = [-0.1, 0, 2, 4, 6]
labels = ['Excellent', 'Good', 'Average', 'Poor Performance']
df['Failed_Group'] = pd.cut(df['Failed'], bins=bins, labels=labels,include_lowest=True)
print(df[['Failed', 'Failed_Group']])

bins_age = [0, 18, 25, 35, 50, 100]
labels_age = ['Minor', 'Young Adult', 'Adult', 'Senior', 'Very Senior']
df['Age_Group']=pd.cut(df['Age'], bins=bins_age, labels=labels_age)
df[['Age', 'Age_Group']]

# d. Elimination of Outliers
```

Output

```
roll_number          Name  Passed  Failed  Total Branch   Age
0         901      A SADWIK     6.0     NaN      6   AIML  22.0
1         911      E SOWMYA     6.0     0.0      6   AIML  20.0
2         921  G SUBBAREDDY     6.0     0.0      6   AIML  21.0
3         931    M SUNAYANA     6.0     0.0      6   AIML  20.0
4         941        S ASIF     NaN     0.0      6   AIML  23.0
5         951    S SHAMEERA     6.0     NaN      6   AIML  19.0
6         961      U TULASI     NaN     NaN      6   AIML   NaN
   Failed Failed_Group
0     0.0    Excellent
1     0.0    Excellent
2     0.0    Excellent
3     0.0    Excellent
4     0.0    Excellent
5     0.0    Excellent
6     0.0    Excellent
/tmp/ipykernel_126554/4265118070.py:20: FutureWarning: A value is trying to be set
on a copy of a DataFrame or Series through chained assignment using an inplace
method.
The behavior will change in pandas 3.0. This inplace method will never work
because the intermediate object on which we are setting values always behaves as a
copy.

For example, when doing 'df[col].method(value, inplace=True)', try using
'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value)
instead, to perform the operation inplace on the original object.


  df['Age'].fillna(df['Age'].mean(),inplace=True)
/tmp/ipykernel_126554/4265118070.py:21: FutureWarning: A value is trying to be set
on a copy of a DataFrame or Series through chained assignment using an inplace
method.
```

```
The behavior will change in pandas 3.0. This inplace method will never work
because the intermediate object on which we are setting values always behaves as a
copy.

For example, when doing 'df[col].method(value, inplace=True)', try using
'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value)
instead, to perform the operation inplace on the original object.


  df['Passed'].fillna(df['Passed'].mean(),inplace=True)
/tmp/ipykernel_126554/4265118070.py:22: FutureWarning: A value is trying to be set
on a copy of a DataFrame or Series through chained assignment using an inplace
method.
The behavior will change in pandas 3.0. This inplace method will never work
because the intermediate object on which we are setting values always behaves as a
copy.

For example, when doing 'df[col].method(value, inplace=True)', try using
'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value)
instead, to perform the operation inplace on the original object.


  df['Failed'].fillna(df['Failed'].mean(),inplace=True)
/tmp/ipykernel_126554/4265118070.py:23: FutureWarning: A value is trying to be set
on a copy of a DataFrame or Series through chained assignment using an inplace
method.
The behavior will change in pandas 3.0. This inplace method will never work
because the intermediate object on which we are setting values always behaves as a
copy.

For example, when doing 'df[col].method(value, inplace=True)', try using
'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value)
instead, to perform the operation inplace on the original object.


  df['Age'].fillna(df['Age'].mode(), inplace=True)
```

	Age	Age_Group
0	22.000000	Young Adult
1	20.000000	Young Adult
2	21.000000	Young Adult
3	20.000000	Young Adult
4	23.000000	Young Adult
5	19.000000	Young Adult

	Age	Age_Group
6	20.833333	Young Adult

Q
9. Apply KNN algorithm for classification and regression
Sub Write). UNIT-II (K-Nearest Neighbor Classifier, Radius Distance Nearest Neighbor Algorithm, KNN Regression) from PPT slide information
Data Set or Data Input:
logic

```
food_data = {
"ingredients": ["apple","bacon","banana","carrot","cheese"],
"sweetness": [10, 1, 10, 7, 1],
"crunchiness":[9, 4, 1, 10, 1],
"cat_type":["fruit","protein","fruit","vegetable","protein"]
}
# Features (sweetness, crunchiness)
X = np.array(list(zip(food_data["sweetness"], food_data["crunchiness"])))
# Labels
y = np.array(food_data["cat_type"])
```

Ans ?

```
{ "ingredients": "tomato", "sweetness": 6, "crunchiness": 4, "type": "???" }
```

logic

```
tomato = np.array([[6, 4]]) # sweetness=6, crunchiness=4
prediction = classifier.predict(tomato)
print("Tomato is:", prediction[0])
```

A

```
import numpy as np
from matplotlib import pyplot as plt
from sklearn.neighbors import KNeighborsClassifier

food_data = {
"ingredients": ["apple","bacon","banana","carrot","cheese"],
"sweetness": [10, 1, 10, 7, 1],
"crunchiness":[9, 4, 1, 10, 1],
"cat_type":["fruit","protein","fruit","vegetable","protein"]
}
# Features (sweetness, crunchiness)
X = np.array(list(zip(food_data["sweetness"], food_data["crunchiness"])))
# Labels
y = np.array(food_data["cat_type"])

classifier = KNeighborsClassifier(n_neighbors=3) # this is the k value
classifier.fit(X,y)
```

```
pred = classifier.predict(np.array([[6, 4]]))
print(pred)
```

Output:

```
['protein']
```

Q 10. Demonstrate decision tree algorithm for a classification problem and perform parameter
tuning for better results
Sub Write). Unit-III(Decision Trees for Classification & Regression) from PPT slide information
Data Set or Data Input:

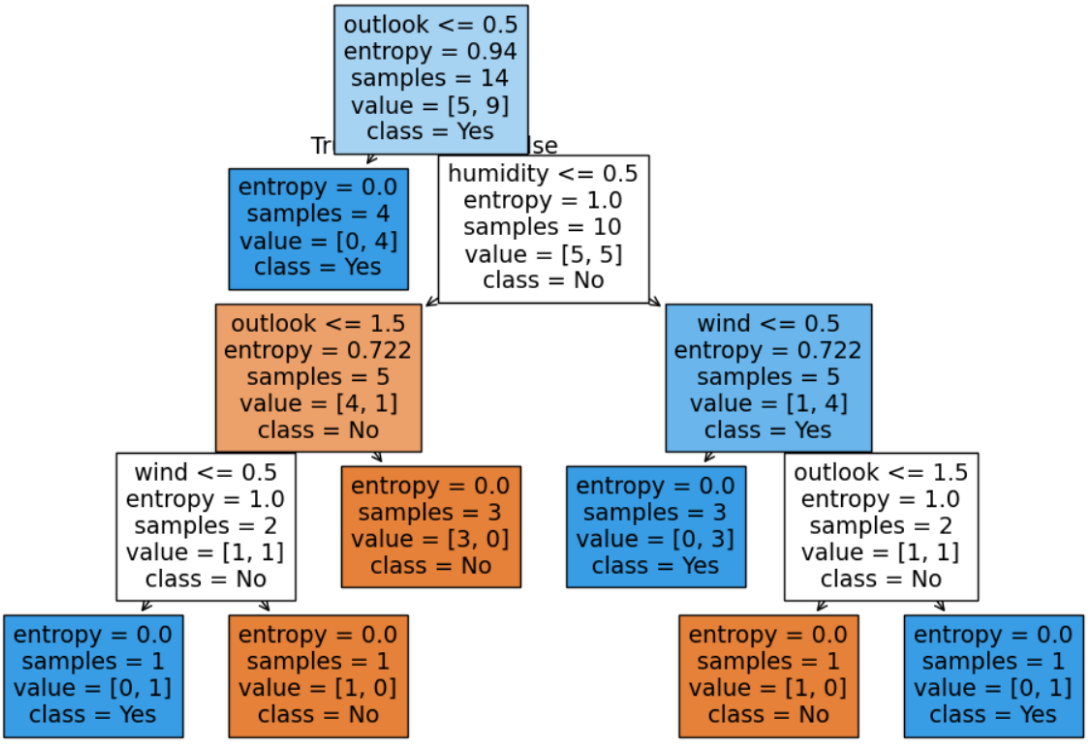
```
data = {
 'outlook': ['sunny','sunny','overcast','rainy','rainy','rainy',
 'overcast','sunny','sunny','rainy','sunny','overcast',
 'overcast','rainy'],
 'temperature': ['hot','hot','hot','mild','cold','cold',
 'cold','mild','cold','mild','mild','mild',
 'hot','mild'],
 'humidity': ['high','high','high','high','normal','normal',
 'normal','high','normal','normal','normal','high',
 'normal','high'],
 'wind': ['false','true','false','false','false','true',
 'true','false','false','false','true','true',
 'false','true'],

 'play': [0,0,1,1,1,0,1,0,1,1,1,1,1,0] # 0 = No, 1 = Yes
}
df = pd.DataFrame(data)
# ----------------------------
# Step 2: Encode Categorical Data
# ----------------------------
le = LabelEncoder()
for column in df.columns:
 df[column] = le.fit_transform(df[column])
X = df.drop('play', axis=1)
y = df['play']
```

Ans ?
Outlook:
overcast = 0,rainy = 1,sunny = 2
Temperature:
cold = 0,hot = 1,mild = 2
Humidity:
high = 0,normal = 1
Wind:
false = 0,true = 1

```
# Example: sunny, cool, high, true
sample = [[2, 1, 0, 1]] # Encoded values (depends on encoding)
prediction = classifier.predict(sample)
```
A)
```
import pandas as pd
from sklearn.preprocessing import LabelEncoder
from sklearn.tree import DecisionTreeClassifier, plot_tree
import matplotlib.pyplot as plt

# ----------------------------
# Step 1: Create Dataset
# ----------------------------

data = {
   'outlook': ['sunny','sunny','overcast','rainy','rainy','rainy',
            'overcast','sunny','sunny','rainy','sunny','overcast',
            'overcast','rainy'],

   'temperature': ['hot','hot','hot','mild','cold','cold',
                'cold','mild','cold','mild','mild','mild',
                'hot','mild'],

   'humidity': ['high','high','high','high','normal','normal',
                'normal','high','normal','normal','normal','high',
                'normal','high'],

   'wind': ['false','true','false','false','false','true',
           'true','false','false','false','true','true',
           'false','true'],

   'play': [0,0,1,1,1,0,1,0,1,1,1,1,1,0]   # 0 = No, 1 = Yes
}

df = pd.DataFrame(data)

# ----------------------------
# Step 2: Encode Categorical Data
# ----------------------------

le = LabelEncoder()

for column in df.columns:
   df[column] = le.fit_transform(df[column])

X = df.drop('play', axis=1)
y = df['play']
```

```
# ----------------------------
# Step 3: Train Decision Tree
# ----------------------------

classifier = DecisionTreeClassifier(criterion='entropy')
classifier.fit(X, y)

# ----------------------------
# Step 4: Prediction
# ----------------------------

# Example: sunny, cool, high, true
sample = [[2, 1, 0, 1]]   # Encoded values (depends on encoding)
prediction = classifier.predict(sample)

print("Prediction (0=No, 1=Yes):", prediction)
```
output:
```
Prediction (0=No, 1=Yes): [0]
```

```
# ----------------------------
# Step 5: Visualize Decision Tree
# ----------------------------

plt.figure(figsize=(12,8))
plot_tree(classifier,
        feature_names=X.columns,
        class_names=['No','Yes'],
        filled=True)

plt.show()
```



Q 11. Apply Random Forest algorithm for classification and regression
Sub Write). Unit-III(Bias–Variance Trade-off, Random Forests for Classification and Regression) from PPT slide information
Data Set or Data Input:

```
# ----------------------------
# Step 1: Create Play Tennis Dataset
# ----------------------------
data = {
'outlook': ['sunny','sunny','overcast','rainy','rainy','rainy',
'overcast','sunny','sunny','rainy','sunny','overcast',
'overcast','rainy'],
'temperature': ['hot','hot','hot','mild','cold','cold',
'cold','mild','cold','mild','mild','mild',
'hot','mild'],
'humidity': ['high','high','high','high','normal','normal',
'normal','high','normal','normal','normal','high',
'normal','high'],
'wind': ['false','true','false','false','false','true',
'true','false','false','false','true','true',
'false','true'],
'play': [0,0,1,1,1,0,1,0,1,1,1,1,1,0] # 0 = No, 1 = Yes
}
df = pd.DataFrame(data)
# ----------------------------
# Step 2: Encode Categorical Data
# ----------------------------
le = LabelEncoder()
```

```
for column in df.columns:
df[column] = le.fit_transform(df[column])
X = df.drop('play', axis=1)
y = df['play']
```
Ans ?
Outlook:
overcast = 0,rainy = 1,sunny = 2
Temperature:
cold = 0,hot = 1,mild = 2
Humidity:
high = 0,normal = 1
Wind:
false = 0,true = 1
```
# ----------------------------
# Step 6: Predict New Sample
# Example: sunny, mild, high, true
# ----------------------------
sample = [[2, 2, 0, 1]] # Encoded values
pred = model.predict(sample)
```
A :
```
from sklearn.ensemble import RandomForestClassifier
#from sklearn.cross_validation import train_test_split
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
data = {
'outlook': ['sunny','sunny','overcast','rainy','rainy','rainy',
'overcast','sunny','sunny','rainy','sunny','overcast',
'overcast','rainy'],
'temperature': ['hot','hot','hot','mild','cold','cold',
'cold','mild','cold','mild','mild','mild',
'hot','mild'],
'humidity': ['high','high','high','high','normal','normal',
'normal','high','normal','normal','normal','high',
'normal','high'],
'wind': ['false','true','false','false','false','true',
'true','false','false','false','true','true',
'false','true'],
'play': [0,0,1,1,1,0,1,0,1,1,1,1,1,0] # 0 = No, 1 = Yes
}
df = pd.DataFrame(data)
# ----------------------------
# Step 2: Encode Categorical Data
# ----------------------------
le = LabelEncoder()
for column in df.columns:
   df[column] = le.fit_transform(df[column])
```

```
X = df.drop('play', axis=1)
y = df['play']
model = RandomForestClassifier(n_estimators=1000, max_features='sqrt')
fitted_model = model.fit(X, y)
sample = [[2, 2, 0, 1]]
predictions = fitted_model.predict(sample)

pred = model.predict([[5.0, 3.5, 1.4, 0.2]])
print(pred)
```
Output

```
[1]
```

Q 12. Demonstrate Naïve Bayes Classification algorithm.
Sub Write). Unit-III(Naive Bayes Classifier) from PPT slide information
Data Set or Data Input:

```
# --------------------------------
# [Study Hours, CGPA]
X = np.array([[1, 5.5],[2, 6.0],[2.5, 6.5],[1.5, 5.8],[2, 6.2],[1, 5.0], # Not Placed (0)
[4, 8.0],[5, 8.5],[3.5, 7.5],[4.5, 8.2],[5, 9.0],[3.8, 7.8] # Placed (1)
])
# 0 = Not Placed, 1 = Placed
y = np.array([0]*6 + [1]*6)
```
Ans?

```
test_point = np.array([[3, 7]])
pred = model.predict(test_point)[0]
if pred == 0:
print("Prediction: Not Placed")
else:
print("Prediction: Placed")
```
A

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.naive_bayes import GaussianNB
X = np.array([[1, 5.5],[2, 6.0],[2.5, 6.5],[1.5, 5.8],[2, 6.2],[1, 5.0], # Not Placed (0)
[4, 8.0],[5, 8.5],[3.5, 7.5],[4.5, 8.2],[5, 9.0],[3.8, 7.8] # Placed (1)
])
# 0 = Not Placed, 1 = Placed
y = np.array([0]*6 + [1]*6)

model = GaussianNB()
model.fit(X, y)
test_point = np.array([[3, 7]])
pred = model.predict(test_point)[0]
if pred == 0:
   print("Prediction: Not Placed")
else:
```

```
   print("Prediction: Placed")
Prediction: Placed
```

Q 13. Apply Support Vector algorithm for classification
Sub Write). Unit-IV(Support Vector Machines, Linearly Non-Separable Case, Non-linear SVM, Kernel Trick) from PPT slide information
Data Set or Data Input:

```
from sklearn import svm, datasets
# Load digits dataset
digits = datasets.load_digits()
X = digits.data
y = digits.target
classifier = svm.SVC(gamma=0.001, C=10)
classifier.fit(X, y)
```

Ans ?

```
# Option 1: Take the first sample of digit 0 from the dataset
manual_0 = X[y == 0][0] # 64 features of the first 0
manual_0 = manual_0.reshape(1, -1)
# Predict
pred_label = classifier.predict(manual_0)[0]
import numpy as np
from sklearn import datasets, svm

# Load digits dataset
digits = datasets.load_digits()
X = digits.data
y = digits.target

# Train SVM on full dataset
classifier = svm.SVC(gamma=0.001, C=10)
classifier.fit(X, y)

# --------------------------------
# Manually create input for digit 0
# --------------------------------

# Option 1: Take the first sample of digit 0 from the dataset
manual_0 = X[y == 0][0]   # 64 features of the first 0
manual_0 = manual_0.reshape(1, -1)

# Predict
pred_label = classifier.predict(manual_0)[0]

print("Predicted digit:", pred_label)
```

```
 Output Predicted digit: 0
```