Question 1:

Demonstrate the following data preprocessing tasks using python libraries.

a. Loading the dataset

b. Dealing with missing data

```
import pandas as pd
from sklearn.datasets import load_breast_cancer

breast_cancer = load_breast_cancer()

data = pd.DataFrame(breast_cancer, columns =
breast_cancer.feature_names)

data['target'] = breast_cancer.target

missing_values = data.isnull().sum()

print(missing_values)
```

Question 2:

Demonstrate the following data preprocessing tasks using python libraries.

a. Dealing with categorical data

b. Scaling the features

c. Splitting dataset into Training and Testing Sets

```
import pandas as pd
import numpy as np

from sklearn.datasets import load_breast_cancer

breast_cancer = load_breast_cancer()

data = pd.DataFrame(breast_cancer.data, columns=
breast_cancer.feature_names)

data['target'] = breast_cancer.target

data['radius_category']=pd.cut(data['mean
radius'],bins=[0,10,20,30,np.inf],labels=['0-10','10-20','20-
30','>30'])

from sklearn.preprocessing import StandardScaler
```

```
independent = ['mean radius', 'mean texture', 'mean perimeter', 'mean
smoothness', 'mean compactness', 'mean concavity', 'mean concave
points', 'mean symmetry', 'mean fractal dimension']

scaler = StandardScaler()
data[independent] = scaler.fit_transform(data[independent])

data.head()

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(data[independent],
data['target'], test_size=0.25, random_state=33)
```

Question 3:

Demonstrate the following Similarity and Dissimilarity Measures using python. .

Pearson's Correlation

a. Cosine Similarity

b. Jaccard Similarity

c. Euclidean Distance Manhattan Distance

```
from sklearn.datasets import load_iris
import numpy as np
import pandas as pd
iris=load_iris()

data=pd.DataFrame(iris.data,columns=iris.feature_names)
data.head()

x=data.drop('petal width (cm)',axis=1).values
target='target'
data['target']=iris.target

from sklearn.metrics.pairwise import
cosine_similarity,euclidean_distances,manhattan_distances

cos_sim=cosine_similarity(data.iloc[[0]],data.iloc[[1]])
euc_dis=euclidean_distances(data.iloc[[0]],data.iloc[[1]])
man_dis=manhattan_distances(data.iloc[[0],:-1],data.iloc[[1],:-1])
print('cosine_similarity',cos_sim)
print('euclidean_distance',euc_dis)
print('manhattan_distances',man_dis[0][0])
```

```
from sklearn.metrics import jaccard_score

# Assuming that column 4 contains the labels for the two samples
labels_1 = data.iloc[0]['target']
labels_2 = data.iloc[1]['target']

# Convert labels to arrays or lists
labels_1_array = [labels_1]
labels_2_array = [labels_2]

# Calculate Jaccard Score
jac_sc = jaccard_score(labels_1_array, labels_2_array)

print(jac_sc)

from scipy.stats import pearsonr
corr,_=pearsonr(data['petal length (cm)'],data['petal width (cm)'])
print(corr)
```

Question 4:

Build a classification model using Decision Tree algorithm on iris dataset?

```
from sklearn.datasets import load_iris
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
import pandas as pd
import numpy as np

iris =load_iris()

data=pd.DataFrame(iris.data,columns=iris.feature_names)

data['target']=iris.target

from sklearn.model_selection import train_test_split

x_train,x_test,y_train,y_test=train_test_split(data.drop('target', axis
= 1),data['target'],test_size=0.3,random_state=42)

dt_model.fit(x_train, y_train)

y_pred = dt_model.predict(x_test)
accuracy = accuracy_score(y_test, y_pred)
print(accuracy)

new_data = np.array([[5.0, 3.5, 1.3, 0.2]])

new_pred = dt_model.predict(new_data)
```

```
print(new_pred)
```

Question 5:

Build a model using linear regression algorithm on any dataset.

```
from sklearn.datasets import load_digits
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
import pandas as pd
import numpy as np

digits =load_digits()

data=pd.DataFrame(digits.data,columns=digits.feature_names)
data['target']=digits.target

from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(data.drop('target', axis
= 1),data['target'],test_size=0.3,random_state=42)

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

y_pred = dt_model.predict(x_test)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
print(mse, r2)
```

Question 6:

Apply Naïve Bayes Classification algorithm on any dataset?

```
from sklearn.datasets import load_breast_cancer
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score
import pandas as pd
import numpy as np

breast_cancer = load_breast_cancer()

data=pd.DataFrame(breast_cancer.data,columns=breast_cancer.feature_name
s)
data['target']=breast_cancer.target

from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(data.drop('target', axis
= 1),data['target'],test_size=0.3,random_state=42)
```

```
dt_model = GaussianNB()
dt_model.fit(x_train, y_train)

y_pred = dt_model.predict(x_test)
accuracy = accuracy_score(y_test, y_pred)
print(accuracy)
```

Question 7:

Apply K- Means clustering algorithm on any dataset.

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.datasets import load_iris
iris = load_iris()

X = pd.DataFrame(iris.data, columns= iris.feature_names)
X

kmeans = KMeans(n_clusters=3)
kmeans.fit(X)

labels = kmeans.predict(X)

#Write sepal length and sepal width names as it was in the dataset
plt.scatter(X['sepal length (cm)'], X['sepal width (cm)'], c=labels)
plt.xlabel('sepal_length')
plt.ylabel('sepal_width')
plt.show()
```

Question 8:

Apply DBSCAN clustering algorithm on any dataset.

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.cluster import DBSCAN
from sklearn.datasets import load_iris
iris = load_iris()

df = pd.DataFrame(iris.data, columns=iris.feature_names)

X = df[['sepal length (cm)', 'sepal width (cm)']]

dbscan = DBSCAN(eps = 0.5, min_samples=5)
```

```
dbscan.fit(X)

labels = dbscan.labels_

plt.scatter(X['sepal length (cm)'], X['sepal width (cm)'], c = labels)
plt.xlabel('sepal legth')
plt.ylabel('sepal width')
plt.show
```