2. Demonstrate the following data preprocessing tasks using python library

a) Dealing with categorical data b) Scaling the features

c) Splitting dataset into Training and Testing Sets

```
import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

data = pd.read_csv('Titanic.csv')

print(data)

x = data.drop('Survived', axis = 1)

y = data['Survived']

print(x)

print(y)

x.drop(['Name', 'Ticket', 'Cabin'],axis = 1, inplace = True)

print(x)


x['Age'] = x['Age'].fillna(x['Age'].mean())

print(x)

x['Embarked'] = x['Embarked'].fillna(x['Embarked'].mode()[0])

print(x)

x = pd.get_dummies(x, columns = ['Sex', 'Embarked'],prefix = ['Sex', 'Embarked'],drop_first = True)

print(x)

from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.2, random_state = 0)

from sklearn.preprocessing import StandardScaler

std_x = StandardScaler()

x_train = std_x.fit_transform(x_train)
```

```
x_test = std_x.transform(x_test)

print(x_train)

print(x_test)
```

3. Write Python code to select features in machine learning using Python

```
from pandas import read_csv

from numpy import set_printoptions

from sklearn.model_selection import train_test_split

from sklearn.feature_selection import SelectKBest

from sklearn.feature_selection import f_classif

from matplotlib import pyplot

path=r'diabetes.csv'

names=['preg','plas','pres','skin','test','mass','peds','age','class']

dataframe=read_csv(path,names=names)

dataframe.head()

array=dataframe.values

x=array[:,0:8]

y=array[:,8]

print(x)

print(y)

x_train,x_test,y_train,y_test,=train_test_split(x,y,test_size=0.33,random_state=1)

print(x_train)

print(x_test)

fs= SelectKBest(score_func=f_classif,k='all')

fs.fit(x_train,y_train)

x_train_fs=fs.transform(x_train)

x_test_fs=fs.transform(x_test)

print(x_train)

for i in range(len(fs.scores_)):

 print('feature %d:%f'%(i,fs.scores_[i]))
 pyplot.bar([i for i in range(len(fs.scores_))],fs.scores_)

pyplot.show()
```

5. Build a classification model using Decision Tree algorithm on iris dataset.

```
import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

import seaborn as sns

from sklearn import tree

from sklearn import metrics

from sklearn.metrics import accuracy_score, classification_report

from sklearn.datasets import load_iris

from sklearn.tree import DecisionTreeClassifier

from sklearn.model_selection import train_test_split

iris = load_iris()

iris = sns.load_dataset('iris')

iris.head()

x=iris.iloc[:,:-1]

y=iris.iloc[:,-1]

x_train, x_test, y_train, y_test = train_test_split(x,y, test_size=0.33, random_state=42)

treemodel = DecisionTreeClassifier()

treemodel.fit(x_train, y_train)

y_pred = treemodel.predict(x_test)

plt.figure(figsize=(20,30))

tree.plot_tree(treemodel, filled=True)

print(classification_report(y_test, y_pred))

from sklearn.metrics import confusion_matrix

cm=confusion_matrix(y_test, y_pred)

print("Confusion Matrix:")
print(cm)

from sklearn.metrics import accuracy_score

accuracy_score(y_test, y_pred)
```

6. Apply Naïve Bayes Classification algorithm on any dataset.

```
import numpy as nm

import matplotlib.pyplot as mtp

import pandas as pd

dataset = pd.read_csv('User_data.csv')

x= dataset.iloc[:,[2,3]].values

print(x)

y = dataset.iloc[:,4].values

print(y)

from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(x,y, test_size=0.25, random_state=0)

from sklearn.preprocessing import StandardScaler

sc= StandardScaler()

x_train = sc.fit_transform(x_train)

x_test = sc.fit_transform(x_test)

from sklearn.naive_bayes import GaussianNB

classifier = GaussianNB()

classifier.fit(x_train , y_train)

y_pred= classifier.predict(x_test)

from sklearn.metrics import confusion_matrix

cm = confusion_matrix(y_test, y_pred)

print("Confusion Matrix:")

print(cm)
```

7. Apply KNN Classification algorithm on any dataset.

```
# Import necessary libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
# Load the dataset
dataset = pd.read_csv('Social_Network_Ads.csv')
# Encoding categorical data if necessary (e.g., Gender column)
# Assume the dataset has columns [UserID, Gender, Age, EstimatedSalary, Purchased]
# If 'Gender' exists, encode it
if 'Gender' in dataset.columns:
 from sklearn.preprocessing import LabelEncoder
 le = LabelEncoder()
 dataset['Gender'] = le.fit_transform(dataset['Gender']) # Encode Gender: Male=1,
Female=0
X = dataset.iloc[:, [1, 2, 3]].values
y = dataset.iloc[:, -1].values
print("First few rows of the dataset:")
print(dataset.head())
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, random_state=42)
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
print("\nScaled Training Features:")
print(X_train)
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(n_neighbors=5, metric='minkowski', p=2) # p=2
corresponds to Euclidean distance
classifier.fit(X_train, y_train)
custom_prediction = classifier.predict(sc.transform([[1, 46, 28000]]))
print(f"\nPrediction for [Gender=Male (1), Age=46, EstimatedSalary=28000]:
{custom_prediction[0]}")
y_pred = classifier.predict(X_test)
results = np.concatenate((y_pred.reshape(len(y_pred), 1), y_test.reshape(len(y_test), 1)),
axis=1)
print("\nPredicted vs Actual values:")
print(results)
from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_test, y_pred)
accuracy = accuracy_score(y_test, y_pred)
print("\nConfusion Matrix:")
print(cm)
print(f"\nAccuracy Score: {accuracy:.2f}")
```

8. Build a model using linear regression algorithm on any dataset.

```
import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

dataset = pd.read_csv('Salary_Data.csv')

x = dataset.iloc[:, :-1].values

y = dataset.iloc[:,-1].values

dataset.head()

from sklearn.model_selection import train_test_split

x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=1/3, random_state=0)

print(x_train)

print(x_test)

print(y_train)

print(y_test)

from sklearn.linear_model import LinearRegression

regressor=LinearRegression()

regressor.fit(x_train,y_train)

y_pred=regressor.predict(x_test)

print(y_test)

print(y_pred)

print(np.concatenate((y_test.reshape(len(y_test),1),y_pred.reshape(len(y_pred),1)),1))

from sklearn.metrics import mean_squared_error

mean=mean_squared_error(y_test,y_pred)

mean

plt.scatter(x_train,y_train,color='red')

plt.plot(x_train,regressor.predict(x_train),color='blue')
plt.title('salary vs Experience(Training set)')

plt.xlabel('years of Experience')

plt.ylabel('salary')

plt.show()

plt.scatter(x_test, y_test, color = 'red')

plt.plot(x_train, regressor.predict(x_train), color = 'blue')

plt.title('Salary vs Experience (Test set)')

plt.xlabel('Years of Experience')

plt.ylabel('Salary')

plt.show()
```

9. Build a model using multi linear regression algorithm on any dataset.

```
import numpy as np

import pandas as pd

from sklearn.model_selection import train_test_split

from sklearn.linear_model import LinearRegression

from sklearn.metrics import mean_squared_error, r2_score

from sklearn.preprocessing import OneHotEncoder

from sklearn.compose import ColumnTransformer


# Load the dataset

data = pd.read_csv('50_Startups.csv')

df = pd.DataFrame(data)


# Display the dataset

print("Dataset:")

print(df.head())


# Features (X) and target (y)

X = df.drop(columns=['Profit'])

y = df['Profit']


# Handle categorical variables (OneHotEncoding for 'State')

column_transformer = ColumnTransformer(

   transformers=[('encoder', OneHotEncoder(), ['State'])], remainder='passthrough')

X = column_transformer.fit_transform(X)
```

```
# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Initialize the Linear Regression model
model = LinearRegression()

# Train the model on the training data
model.fit(X_train, y_train)

# Make predictions on the test data
y_pred = model.predict(X_test)

# Evaluate the model
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f"Mean Squared Error: {mse}")
print(f"R-squared Score: {r2}")

# Display the coefficients and intercept
print("Coefficients:", model.coef_)
print("Intercept:", model.intercept_)
```

10. Apply Hierarchical Clustering algorithm on any dataset.

```
import numpy as nm

import matplotlib.pyplot as mtp

import pandas as pd


dataset = pd.read_csv('Mall_Customers.csv')

x = dataset.iloc[:, [3, 4]].values

dataset.head()


#Finding the optimal number of clusters using the dendrogram

import scipy.cluster.hierarchy as shc

dendro = shc.dendrogram(shc.linkage(x, method="ward"))

mtp.title("Dendrogrma Plot")

mtp.ylabel("Euclidean Distances")

mtp.xlabel("Customers")

mtp.show()


from sklearn.cluster import AgglomerativeClustering

hc = AgglomerativeClustering(n_clusters=5, metric='euclidean', linkage='ward')

y_pred = hc.fit_predict(x)


#visulaizing the clusters

mtp.scatter(x[y_pred == 0, 0], x[y_pred == 0, 1], s = 100, c = 'blue', label = 'Cluster 1')

mtp.scatter(x[y_pred == 1, 0], x[y_pred == 1, 1], s = 100, c = 'green', label = 'Cluster 2')

mtp.scatter(x[y_pred== 2, 0], x[y_pred == 2, 1], s = 100, c = 'red', label = 'Cluster 3')
mtp.scatter(x[y_pred == 3, 0], x[y_pred == 3, 1], s = 100, c = 'cyan', label = 'Cluster 4')

mtp.scatter(x[y_pred == 4, 0], x[y_pred == 4, 1], s = 100, c = 'magenta', label = 'Cluster 5')

mtp.title('Clusters of customers')

mtp.xlabel('Annual Income (k$)')

mtp.ylabel('Spending Score (1-100)')

mtp.legend()

mtp.show()
```

11. Apply DBSCAN clustering algorithm on any dataset

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from sklearn.cluster import DBSCAN

from sklearn.preprocessing import StandardScaler


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

df.head()


x_train=df[['Age','Annual Income (k$)','Spending Score (1-100)']]

x_train


scaler = StandardScaler()

X_scaled = scaler.fit_transform(X)


# Apply DBSCAN

dbscan = DBSCAN(eps=0.5, min_samples=5)  # Adjust 'eps' and 'min_samples' as needed

y_dbscan = dbscan.fit_predict(X_scaled)


# Add cluster labels to the original dataset

df['Cluster'] = y_dbscan


# Display clustering results

print(df['Cluster'].value_counts())
```

```
# Visualize the clusters

plt.figure(figsize=(8, 6))

plt.scatter(X['Annual Income (k$)'], X['Spending Score (1-100)'], c=y_dbscan, cmap='rainbow', s=50,
alpha=0.7)

plt.title("DBSCAN Clustering on Mall Customers Dataset")

plt.xlabel("Annual Income (k$)")

plt.ylabel("Spending Score (1-100)")

plt.colorbar(label="Cluster Label")

plt.show()
```