*1.WRITE A PROGRAM ON DATA CLEANING TECHNICS*

*#creating dataframe*import pandas as pd  
import numpy as np  
data=pd.DataFrame(np.random.randn(4,3),index=[1,3,5,7],columns=('A','B','C'))  
print(data)  
  
*#Re-indexing dataframe*data=data.reindex([1,2,3,4,5,6,7])  
print(data)  
  
*#checking for missing values*print(data['A'].isnull())  
*#filling missing values  
 #filling with specific value*print(data.fillna(612))  
*#remove duplicate data*print(data.drop\_duplicates())  
*#filling with Forward method*print(data.fillna(method='pad'))  
 *#filling with backward method*print(data.fillna(method='bfill'))  
*#Drop missing values*print(data.dropna())  
*#Replacing Generic value with specific value*df=pd.DataFrame({'ONE':[1,2,3,4],'TWO':[5,6,7,8]})  
print(df.replace({4:10,8:12}))

#using CSV file

import pandas as pd  
data=pd.read\_csv('dataset.csv')  
print(data)

1. Write a program on preprocessin Techniques

import pandas as pd  
from sklearn import preprocessing as p  
data=pd.read\_csv('dataset.csv')  
print(data)  
le=p.LabelEncoder()  
*#The fit() method helps in fitting the data into a model  
#transform() method helps in transforming the data into a form that is more suitable for the model.*data['dept']=le.fit\_transform(data['dept'])  
print(data)  
  
from sklearn.preprocessing import OneHotEncoder  
*#converting types of columns to category  
#astype returns a new DataFrame where the data types has been changed to the specified type.*data['dept']=data['dept'].astype('category')  
*#Assigning numerical values & storing in another columns  
#cat.codes assigns a numerical value to the (ordinal) categorical variables*data['deptnew']=data['dept'].cat.codes  
*#creating instance of onehotencoder*a=OneHotEncoder()  
*#passing encoded columns*a\_data=pd.DataFrame(a.fit\_transform(data[['dept']]).toarray())  
*#merging with main dataset*new\_df=data.join(a\_data)  
print(new\_df)  
  
*#CountVectorizer*empdata={'eid':[200,500,100,750],'eloc':['Delhi','Pune','Goa','Bangalore'],  
 'sal':[30000,35000,55000,43000]}  
df=pd.DataFrame(data=empdata)  
print(empdata)  
print(df)  
from sklearn.feature\_extraction.text import CountVectorizer  
v=CountVectorizer()  
*#The fit() method helps in fitting the data into a model  
#transform() method helps in transforming the data into a form that is more suitable for the model.*v.fit(df['eloc'].values)  
x=v.transform(df['eloc'].values)  
print(x)  
print('Extracting Emp Locations:',v.get\_feature\_names\_out())  
print(x.toarray())  
  
#d.Min-Max Scaling  
from sklearn.preprocessing import MinMaxScaler  
d=[[2,10],[3,2],[3.2,2],[0,10]]  
s=MinMaxScaler()  
s.fit(d)  
print(s.fit\_transform(d))  
  
#e.Standardization  
from sklearn.preprocessing import StandardScaler  
d=[[2,10],[3,2],[3.2,2],[0,10]]  
s=StandardScaler()  
s.fit(d)  
print(s.fit\_transform(d))  
  
#f.Normalizing  
from sklearn.preprocessing import Normalizer  
d=[[2,10],[3,2],[3.2,2],[0,10]]  
s=Normalizer()  
s.fit(d)  
print(s.fit\_transform(d))  
  
  
#g.bag of Words  
from sklearn.feature\_extraction.text import CountVectorizer  
d=pd.read\_csv('bag.csv')  
f=[]  
for i in d['Review2'].values:  
 sentence=' '.join(s for s in i.split())  
 f.append(sentence.upper().strip())  
print(f)  
v=CountVectorizer(ngram\_range=(1,3))  
BoW=v.fit\_transform(f)  
print('Extracting features:',v.get\_feature\_names\_out())  
print('Shape of Matrix:',BoW.toarray())

3 #.MAKE YOUR DATA READY TO TRAIN AND TEST A MODEL

import pandas as pd  
from sklearn.model\_selection import train\_test\_split  
from sklearn.preprocessing import StandardScaler  
from sklearn import preprocessing as p  
*# Load your dataset (replace with your data source)*data = pd.read\_csv("dataset.csv")  
  
  
*#Converting string data to Numeric*le=p.LabelEncoder()  
*#The fit() method helps in fitting the data into a model  
#transform() method helps in transforming the data into a form that is more suitable for the model.*data['dept']=le.fit\_transform(data['dept'])  
data['ename']=le.fit\_transform(data['ename'])  
  
*# Separate features (X) and target variable (y)*X = data.drop("dept", axis=1)  
y = data["sal"]  
  
*# Split data 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)  
  
*# Handle missing values (if necessary)  
# Example: Fill missing values with the mean*X\_train = X\_train.fillna(X\_train.mean())  
X\_test = X\_test.fillna(X\_test.mean())  
  
*# Feature scaling (optional but often recommended)*scaler = StandardScaler()  
X\_train = scaler.fit\_transform(X\_train)  
X\_test = scaler.transform(X\_test)

*# Now you can train your machine learning model using X\_train, y\_train  
# and evaluate its performance on the X\_test, y\_test set*from sklearn.linear\_model import LinearRegression  
a=LinearRegression()  
a.fit(X\_train,y\_train)  
print(a.predict(X\_test))  
print(a.score(X\_train,y\_train))  
print(a.intercept\_)  
print(a.coef\_)  
print(a.coef\_[0]\*10+a.coef\_[1]\*3+a.intercept\_)

3b#SIMPLE LINEAR

df=pd.read\_csv('price.csv')  
import matplotlib.pyplot as plt  
plt.scatter(df['sqft'],df['price'])  
plt.xlabel('squarefeet')  
plt.ylabel('price')  
plt.show()  
from sklearn.linear\_model import LinearRegression  
a=LinearRegression()  
x=df[['sqft']]  
y=df[['price']]  
a.fit(x,y)  
print(a.predict([[60]]))  
print(a.coef\_)  
print(a.intercept\_)

4.Train,Validate,Test KNN Model

import pandas as pd  
import numpy as np  
from sklearn.model\_selection import train\_test\_split, cross\_val\_score  
from sklearn.neighbors import KNeighborsClassifier  
from sklearn.metrics import accuracy\_score, precision\_score, recall\_score  
from sklearn.preprocessing import StandardScaler  
  
import matplotlib.pyplot as plt  
data = pd.read\_csv("transaction.csv")  
plt.scatter(data['dist\_from\_home'],data['purchase\_price\_ratio'], data['fraud'])  
plt.xlabel('dist\_from\_home')  
plt.ylabel('purchase\_price\_ratio')  
plt.show()  
  
*# Split the data into features (X) and target (y)*

*First, the data is split into features (X) and target (y) using the drop method to remove the 'fraud' column from the original dataframe df. The axis=1 parameter specifies that the column should be dropped along the columns axis (i.e., horizontally). The resulting X variable contains all the columns except for 'fraud', while y contains only the 'fraud' column.*

*Next, the data is split into training and test sets using the train\_test\_split function from the sklearn.model\_selection module. The test\_size parameter specifies the proportion of the data that should be allocated to the test set (in this case, 20%).*

X = data.drop('fraud', axis=1)  
y = data['fraud']  
  
*# Split the data into training and test sets*X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2)  
  
*# Scale the features using StandardScaler*scaler = StandardScaler()  
X\_train = scaler.fit\_transform(X\_train)  
X\_test = scaler.transform(X\_test)

#n\_neighbors set to 3. This means that the classifier #will consider the 3 nearest neighbors when making #predictions.  
knn = KNeighborsClassifier(n\_neighbors=3)  
knn.fit(X\_train, y\_train)

#used to evaluate the performance of the KNN #classifier on a test data set by comparing the predicted #labels to the true labels.  
y\_pred = knn.predict(X\_test)

#by comparing the predicted values (stored in y\_pred) #with the actual values (stored in y\_test). The #accuracy\_score function from the scikit-learn library #is used to perform this calculation. The resulting #accuracy value is then printed to the console using the #print function.  
  
accuracy = accuracy\_score(y\_test, y\_pred)  
print("Accuracy:", accuracy)  
  
k\_values = [i for i in range (1,5)]  
scores = []  
  
scaler = StandardScaler()  
X = scaler.fit\_transform(X)

#Overall, this code is performing a grid search over different values of k for the k-nearest neighbors classifier and using cross-validation to evaluate the performance of each model. The final output is a list of mean accuracy scores for each value of k.  
  
for k in k\_values:  
 knn = KNeighborsClassifier(n\_neighbors=k)  
 score = cross\_val\_score(knn, X, y, cv=3)  
 scores.append(np.mean(score))  
print(scores)  
  
#Overall, this code is selecting the best value of k for the K-Nearest Neighbors algorithm based on the highest score, and then using this value to train a KNN model on the training data.  
best\_index = np.argmax(scores)  
best\_k = k\_values[best\_index]  
print(best\_k)  
  
knn = KNeighborsClassifier(n\_neighbors=best\_k)  
knn.fit(X\_train, y\_train)  
  
y\_pred = knn.predict(X\_test)  
  
accuracy = accuracy\_score(y\_test, y\_pred)  
precision = precision\_score(y\_test, y\_pred)  
recall = recall\_score(y\_test, y\_pred)

print("Accuracy:", accuracy)  
print("Precision:", precision)  
print("Recall:", recall)

Transaction.csv

dist\_from\_home,purchase\_price\_ratio,fraud  
2.1,6.4,1  
3.8,2.2,1  
15.7,4.4,1  
26.7,4.6,1  
10.7,4.9,1