**1.Reena is a high school teacher and she wants to investigate if there's a relationship between the number of hours students spend studying for a math test and their resulting scores. She decides to collect data from a random sample of 20 students. She record each student's study hours (independent variable) and their corresponding math test scores (dependent variable).**

**Predict a student's math test score based on the number of hours they spend studying?**

|  |  |
| --- | --- |
| hours\_studied | exam\_scores |
| 2 | 60 |
| 3 | 70 |
| 4 | 80 |
| 5 | 85 |
| 6 | 75 |
| 7 | 90 |
| 8 | 95 |
| 9 | 85 |
| 10 | 80 |

**Code:**

import pandas as pd

from sklearn.linear\_model import LinearRegression

# Read the CSV dataset

data = pd.read\_csv('student\_scores.csv')

# Extracting features (hours studied) and target (exam scores)

x\_train = data['hours\_studied'].values.reshape(-1,1)

y\_train = data['exam\_scores'].values

# Initialize and train the linear regression model

model = LinearRegression()

model.fit(x\_train, y\_train)

# Function to predict exam score based on hours studied

def predict\_score(hours):

    return model.predict([[hours]])

# Example usage

hours\_prompt = int(input("Enter the number of hours studied: "))

predicted\_score = predict\_score(hours\_prompt)

print(f"Predicted exam score based on {hours\_prompt} hours studied: {predicted\_score[0]}")

**2. Consider you're a manager at an ice cream shop located in a tourist destination. You're interested in predicting the daily ice cream sales based on the temperature because you believe that warmer weather leads to higher ice cream sales. Over the past month, you've been keeping track of the daily temperature (in degrees Celsius) and the corresponding ice cream sales (in dollars) at your shop.**

**You have collected the following data:**

|  |  |
| --- | --- |
| Temp  (degree Celsius) | Ice Cream Sale  (In litres) |
| 20 | 17 |
| 25 | 23 |
| 30 | 28 |
| 33 | 31 |
| 35 | 38 |
| 38 | 41 |
| 40 | 53 |
| 47 | 62 |
| 52 | 90 |

**Code:**

import pandas as pd

from sklearn.linear\_model import LinearRegression

data = pd.read\_csv('ice\_cream\_sales.csv')

# Extracting features (temperature) and target (ice cream sales)

X = data['Temp(degree Celsius)'].values.reshape(-1,1)*;*

y = data['Ice Cream Sale(In litres)'].values

# Initialize and train the linear regression model

model = LinearRegression()

model.fit(X, y)

# Function to predict ice cream sales based on temperature

def predict\_sales(temperature):

    return model.predict([[temperature]])

# Example usage

temperature\_prompt = int(input("Enter the temperature (in °C): "))

predicted\_sales = predict\_sales(temperature\_prompt)

print(f"Predicted ice cream sales for {temperature\_prompt}°C: {predicted\_sales[0]} litres")

**3. Consider you're an owner of a tea shop located in a tourist destination. You're interested in predicting the daily tea sales based on the temperature because you believe that cold weather leads to higher tea sales. Over the past month, you've been keeping track of the daily temperature (in degrees Celsius) and the corresponding tea sales (in dollars) at your shop.**

|  |  |
| --- | --- |
| Temp  (degree Celsius) | Tea Sale  (In litres) |
| 20 | 45 |
| 25 | 37 |
| 30 | 31 |
| 33 | 28 |
| 35 | 26 |
| 38 | 23 |
| 40 | 17 |
| 47 | 9 |
| 52 | 7 |

**Code:**

import pandas as pd

from sklearn.linear\_model import LinearRegression

data = pd.read\_csv('tea\_sales.csv')

# Extracting features (temperature) and target (tea sales)

X = data['Temperature (°C)'].values.reshape(-1, 1)

y = data['Tea Sale (litres)'].values

# Initialize and train the linear regression model

model = LinearRegression()

model.fit(X, y)

# Function to predict tea sales based on temperature

def predict\_sales(temperature):

    return model.predict([[temperature]])

# Example usage

temperature\_prompt = int(input("Enter the temperature (in °C): "))

predicted\_sales = predict\_sales(temperature\_prompt)

print(f"Predicted tea sales for {temperature\_prompt}°C: {predicted\_sales[0]} litres")

**4.Perform multiple linear regressions for below mentioned data set.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **CAR** | **MODEL** | **VOLUME** | **WEIGHT** | **CO2** |
| Toyato | Aygu | 1000 | 790 | 99 |
| Mitsubish | Space Star | 1200 | 1160 | 95 |
| Skoda | Citigo | 1000 | 929 | 95 |
| Flat | Gold | 900 | 865 | 90 |
| Mini | cooper | 1500 | 1140 | 105 |

**Code:**

import pandas as pd

from sklearn.linear\_model import LinearRegression

# Load the dataset from CSV

data = pd.read\_csv('car\_data.csv')

# Extracting features (volume and weight) and target (CO2 emissions)

X = data[['VOLUME', 'WEIGHT']].values.reshape(-1,2)

y = data['CO2'].values

# Initialize and train the linear regression model

model = LinearRegression()

model.fit(X, y)

# Function to predict CO2 emissions based on volume and weight

def predict\_co2(volume, weight):

    return model.predict([[volume, weight]])

# Example usage

volume\_prompt = float(input("Enter the volume of the car (in cc): "))

weight\_prompt = float(input("Enter the weight of the car (in kg): "))

predicted\_co2 = predict\_co2(volume\_prompt, weight\_prompt)

print(f"Predicted CO2 emissions: {predicted\_co2[0]} g/km")

**6. Consider the below mentioned sample dataset and apply linear binary classifier to decide whether the customer will purchase particular product or not.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **User ID** | **Gender** | **Age** | **Estimated Salary** | **Purchased** |
| **15624510** | **Male** | **19** | **19000** | **0** |
| **15810944** | **Male** | **35** | **20000** | **0** |
| **15728773** | **Male** | **27** | **58000** | **0** |
| **15598044** | **Female** | **27** | **84000** | **0** |
| **15694829** | **Female** | **32** | **150000** | **1** |
| **15600575** | **Male** | **25** | **33000** | **0** |
| **15704987** | **Male** | **32** | **18000** | **0** |
| **15628972** | **Male** | **18** | **82000** | **0** |
| **15697686** | **Male** | **29** | **80000** | **0** |
| **15733883** | **Male** | **47** | **25000** | **1** |
| **15617482** | **Male** | **45** | **26000** | **1** |
| **15736760** | **Female** | **47** | **49000** | **1** |
| **15714658** | **Male** | **48** | **41000** | **1** |

**Code:**

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import LogisticRegression

from sklearn.metrics import accuracy\_score, classification\_report

# Load the dataset from CSV

data = pd.read\_csv('customer\_data.csv')

# Drop 'User ID' column from X, and store 'Purchased' in y

X = data.drop(['User ID', 'Purchased'], axis=1)

y = data['Purchased'].values

# One-hot encode the 'Gender' column

X = pd.get\_dummies(X, drop\_first=True)

# 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 and train the logistic regression model

model = LogisticRegression()

model.fit(X\_train, y\_train)

# Predictions on the testing set

y\_pred = model.predict(X\_test)

# Evaluate the model

accuracy = accuracy\_score(y\_test, y\_pred)

print("Accuracy:", accuracy)

# Classification report

print("Classification Report:")

print(classification\_report(y\_test, y\_pred))

**7. Raj is tasked with building a machine learning model to determine whether an individual is eligible to receive a credit card. Using a Naive Bayes classifier and the provided dataset containing features such as age, income, employment status, credit score, how he would approach this task.**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Rid** | **Age** | **Income** | **Student** | **Credit** | **Buy** |
| **1** | **youth** | **high** | **no** | **fair** | **no** |
| **2** | **youth** | **high** | **no** | **excellent** | **no** |
| **3** | **middle\_aged** | **high** | **no** | **fair** | **yes** |
| **4** | **senior** | **medium** | **no** | **fair** | **yes** |
| **5** | **senior** | **low** | **yes** | **fair** | **yes** |
| **6** | **senior** | **low** | **yes** | **excellent** | **no** |
| **7** | **middle\_aged** | **low** | **yes** | **excellent** | **yes** |
| **8** | **youth** | **medium** | **no** | **fair** | **no** |
| **9** | **youth** | **low** | **yes** | **fair** | **yes** |
| **10** | **senior** | **medium** | **yes** | **fair** | **yes** |
| **11** | **youth** | **medium** | **yes** | **excellent** | **yes** |
| **12** | **middle\_aged** | **medium** | **no** | **excellent** | **yes** |
| **13** | **middle\_aged** | **high** | **yes** | **fair** | **yes** |
| **14** | **senior** | **medium** | **no** | **excellent** | **no** |

**Code**:

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.naive\_bayes import GaussianNB

from sklearn.preprocessing import LabelEncoder

from sklearn.metrics import accuracy\_score, classification\_report

# Load the dataset

data = pd.read\_csv("credit\_card\_data.csv")

# Encode categorical variables

label\_encoders = {}

for column in data.columns:

    if data[column].dtype == 'object':

        label\_encoders[column] = LabelEncoder()

        data[column] = label\_encoders[column].fit\_transform(data[column])

# Split features and target variable

X = data.drop(columns=['Buy'])

y = data['Buy']

# 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 and train the Naive Bayes classifier

nb\_classifier = GaussianNB()

nb\_classifier.fit(X\_train, y\_train)

# Make predictions

y\_pred = nb\_classifier.predict(X\_test)

# Calculate accuracy

accuracy = accuracy\_score(y\_test, y\_pred)

print("Accuracy:", accuracy)

# Generate and display classification report

classification\_rep = classification\_report(y\_test, y\_pred)

print("Classification Report:")

print(classification\_rep)

**8. Apply naïve bayes algorithm to decide whether to play tennis on a given day (if Outlook=Rainy, Temperature=Mild, Humidity=High, Windy=True) based on weather conditions. You have a dataset of past instances where you recorded whether tennis was played or not, along with corresponding weather conditions.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Outlook** | **Temperature** | **Humidity** | **Windy** | **Play Tennis** |
| **Sunny** | **Hot** | **High** | **False** | **No** |
| **Sunny** | **Hot** | **High** | **True** | **No** |
| **Overcast** | **Hot** | **High** | **False** | **Yes** |
| **Rainy** | **Mild** | **High** | **False** | **Yes** |
| **Rainy** | **Cool** | **Normal** | **False** | **Yes** |
| **Rainy** | **Cool** | **Normal** | **True** | **No** |
| **Overcast** | **Cool** | **Normal** | **True** | **Yes** |
| **Sunny** | **Mild** | **High** | **False** | **No** |
| **Sunny** | **Cool** | **Normal** | **False** | **Yes** |
| **Rainy** | **Mild** | **Normal** | **False** | **Yes** |
| **Sunny** | **Mild** | **Normal** | **True** | **Yes** |
| **Overcast** | **Mild** | **High** | **True** | **Yes** |
| **Overcast** | **Hot** | **Normal** | **False** | **Yes** |

**Code:**

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.naive\_bayes import GaussianNB

from sklearn.metrics import accuracy\_score, classification\_report

df = pd.read\_csv('tennis\_data.csv')

df\_encoded = pd.get\_dummies(df, columns=['Outlook', 'Temperature', 'Humidity', 'Windy'])

X = df\_encoded.drop('Play Tennis', axis=1)

y = df\_encoded['Play Tennis']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

nb\_classifier = GaussianNB()

nb\_classifier.fit(X\_train, y\_train)

y\_pred = nb\_classifier.predict(X\_test)

accuracy = accuracy\_score(y\_test, y\_pred)

print("Accuracy:", accuracy)

print("Classification Report:")

print(classification\_report(y\_test, y\_pred))

new\_instance = pd.DataFrame({'Outlook\_Overcast': [0], 'Outlook\_Rainy': [1], 'Outlook\_Sunny': [0],

                             'Temperature\_Cool': [0], 'Temperature\_Hot': [0], 'Temperature\_Mild': [1],

                             'Humidity\_High': [1], 'Humidity\_Normal': [0],

                             'Windy\_False': [0], 'Windy\_True': [1]})

prediction = nb\_classifier.predict(new\_instance)

print("Prediction:", prediction[0])

**9. Suppose you're tasked with building a machine learning model to predict whether a patient is likely to have diabetes. Using a decision tree classifier and the provided dataset containing features such as glucose level, BMI, blood pressure, age, and family history, how would you approach this task? Consider the below mentioned dataset.**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Glucose** | **Blood Pressure** | **Skin Thickness** | **Insulin** | **BMI** | **Age** | **Outcome** |
| **148** | **72** | **35** | **0** | **33.6** | **50** | **1** |
| **85** | **66** | **29** | **0** | **26.6** | **31** | **0** |
| **183** | **64** | **0** | **0** | **23.3** | **32** | **1** |
| **89** | **66** | **23** | **94** | **28.1** | **21** | **0** |
| **137** | **40** | **35** | **168** | **43.1** | **33** | **1** |
| **116** | **74** | **0** | **0** | **25.6** | **30** | **0** |
| **78** | **50** | **32** | **88** | **31** | **26** | **1** |
| **115** | **0** | **0** | **0** | **35.3** | **29** | **0** |
| **197** | **70** | **45** | **543** | **30.5** | **53** | **1** |
| **125** | **96** | **0** | **0** | **0** | **54** | **1** |
| **110** | **92** | **0** | **0** | **37.6** | **30** | **0** |
| **168** | **74** | **0** | **0** | **38** | **34** | **1** |

**Code:**

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.tree import DecisionTreeClassifier

from sklearn.metrics import accuracy\_score, classification\_report

data = pd.read\_csv('diabetes\_data.csv')

X = data.drop('Outcome', axis=1)

y = data['Outcome']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

model = DecisionTreeClassifier(random\_state=42)

model.fit(X\_train, y\_train)

y\_pred = model.predict(X\_test)

accuracy = accuracy\_score(y\_test, y\_pred)

print("Accuracy:", accuracy)

print("Classification Report:")

print(classification\_report(y\_test, y\_pred))

**10. Consider the below mentioned dataset and apply decision tree algorithm to decide whether the particular day is preferable for playing tennis.**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Day** | **Outlook** | **Temprature** | **Humidity** | **Wind** | **Play\_Tennis** |
| **D1** | **Sunny** | **Hot** | **High** | **Weak** | **No** |
| **D2** | **Sunny** | **Hot** | **High** | **Strong** | **No** |
| **D3** | **Overcast** | **Hot** | **High** | **Weak** | **Yes** |
| **D4** | **Rain** | **Mild** | **High** | **Weak** | **Yes** |
| **D5** | **Rain** | **Cool** | **Normal** | **Weak** | **Yes** |
| **D6** | **Rain** | **Cool** | **Normal** | **Strong** | **No** |
| **D7** | **Overcast** | **Cool** | **Normal** | **Strong** | **Yes** |
| **D8** | **Sunny** | **Mild** | **High** | **Weak** | **No** |
| **D9** | **Sunny** | **Cool** | **Normal** | **Weak** | **Yes** |
| **D10** | **Rain** | **Mild** | **Normal** | **Weak** | **Yes** |

import numpy as np

import pandas as pd

from sklearn import metrics #Import scikit-learn metrics module for accuracy calculation

from sklearn.metrics import classification\_report, confusion\_matrix

# Read csv file

df=pd.read\_csv("tennis\_dataset.csv")

value=['Outlook','Temprature','Humidity','Wind']

from sklearn import preprocessing

string\_to\_int= preprocessing.LabelEncoder()

df=df.apply(string\_to\_int.fit\_transform)

feature\_cols = ['Outlook','Temprature','Humidity','Wind']

X = df[feature\_cols ]

y = df['Play Tennis']

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.30)

# perform training

from sklearn.tree import DecisionTreeClassifier # import the classifier

classifier =DecisionTreeClassifier(criterion="entropy", random\_state=100) # create a classifier object

classifier.fit(X\_train, y\_train)

#Predict the response for test dataset

y\_pred= classifier.predict(X\_test)

# Accuracy

from sklearn.metrics import accuracy\_score

print("Accuracy:",metrics.accuracy\_score(y\_test, y\_pred))

# Confusion Matrix

print(confusion\_matrix(y\_test, y\_pred))

print(classification\_report(y\_test, y\_pred,zero\_division=1))

**11. Consider you're analyzing customer transaction data for a grocery store chain. Using the Apriori algorithm and the provided dataset of customer transactions, how would you identify frequent item sets? Sample dataset is given below.**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **T1** | **meatballs** | **milk** | **honey** | **french fries** | **protein bar** | |  |  |
| **T2** | **red wine** | **shrimp** | **pasta** | **pepper** | **eggs** | **chocolate** | **shampoo** |  |
| **T3** | **rice** | **sparkling water** | |  |  |  |  |  |
| **T4** | **spaghetti** | **mineral water** | **ham** | **body spray** | **pancakes** | **green tea** |  |  |
| **T5** | **burgers** | **grated cheese** | **shrimp** | **pasta** | **avocado** | **honey** | **white wine** | **toothpaste** |
| **T6** | **eggs** |  |  |  |  |  |  |  |
| **T7** | **parmesan cheese** | **spaghetti** | **soup** | **avocado** | **milk** | **fresh bread** | |  |

import csv

import pandas as pd

from mlxtend.preprocessing import TransactionEncoder

from mlxtend.frequent\_patterns import apriori

# Initialize an empty list to store the dataset

dataset = []

# Read the CSV file and convert it into the desired format

with open('customer\_transactions.csv', newline='') as csvfile:

    reader = csv.reader(csvfile)

    for row in reader:

        # Append each row (transaction) to the dataset as a list

        dataset.append(row)

# Convert the transactions into a transaction encoding format

te = TransactionEncoder()

te\_ary = te.fit(dataset).transform(dataset)

df = pd.DataFrame(te\_ary, columns=te.columns\_)

# Apply the Apriori algorithm to find frequent item sets

frequent\_itemsets = apriori(df, min\_support=0.003, use\_colnames=True)

# Sort the frequent item sets by support values

frequent\_itemsets\_sorted = frequent\_itemsets.sort\_values(by='support', ascending=False)

# Display the first 20 frequent item sets

print(frequent\_itemsets\_sorted.head(20))

**12. Consider the below mentioned data types and apply partition clustering algorithm to group the similar instances.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **CustomerID** | **Genre** | **Age** | **Annual Income (k$)** | **Spending Score**  **(1-100)** |
| **1** | **Male** | **19** | **15** | **39** |
| **2** | **Male** | **21** | **15** | **81** |
| **3** | **Female** | **20** | **16** | **6** |
| **4** | **Female** | **23** | **16** | **77** |
| **5** | **Female** | **31** | **17** | **40** |
| **6** | **Female** | **22** | **17** | **76** |
| **7** | **Female** | **35** | **18** | **6** |
| **8** | **Female** | **23** | **18** | **94** |
| **9** | **Male** | **64** | **19** | **3** |
| **10** | **Female** | **30** | **19** | **72** |
| **11** | **Male** | **67** | **19** | **14** |
| **12** | **Female** | **35** | **19** | **99** |
| **13** | **Female** | **58** | **20** | **15** |
| **14** | **Female** | **24** | **20** | **77** |
| **15** | **Male** | **37** | **20** | **13** |

Code:

import pandas as pd

from sklearn.cluster import KMeans

# Read the data from the CSV file

data = pd.read\_csv("customer\_data2.csv")

# Extracting relevant features for clustering

X = data.iloc[:, [2, 3, 4]].values  # Age, Annual Income, Spending Score

# Applying K-means to the dataset with 5 clusters

kmeans = KMeans(n\_clusters=5, init='k-means++', random\_state=42)

y\_kmeans = kmeans.fit\_predict(X)

# Adding the cluster labels to the dataset

data['Cluster'] = y\_kmeans

# Displaying the clustered data

print(data)

**13. Consider the below mentioned data types and apply K-Means clustering algorithm to group**

**the similar instances.**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **status\_id** | **status\_type** | **num\_**  **commnts** | **num\_**  **shares** | **num\_**  **likes** | **num\_**  **wows** | **num\_**  **hahas** | **num\_sads** | **num\_angrys** |
| **7474** | **video** | **512** | **262** | **432** | **3** | **1** | **1** | **0** |
| **7757** | **photo** | **0** | **0** | **150** | **0** | **0** | **0** | **0** |
| **7397** | **video** | **236** | **57** | **204** | **1** | **1** | **0** | **0** |
| **9452** | **photo** | **0** | **0** | **111** | **0** | **0** | **0** | **0** |
| **3739** | **photo** | **0** | **0** | **204** | **0** | **0** | **0** | **0** |
| **8773** | **photo** | **6** | **0** | **211** | **1** | **0** | **0** | **0** |

import pandas as pd

from sklearn.cluster import KMeans

# Read the data from the CSV file

data = pd.read\_csv("status\_data.csv")

# Extracting relevant features for clustering

X = data.iloc[:, 2:].values  # Exclude the status\_id and status\_type columns

# Applying K-means to the dataset with 2 clusters (for demonstration)

kmeans = KMeans(n\_clusters=5, init='k-means++', random\_state=42)

y\_kmeans = kmeans.fit\_predict(X)

# Adding the cluster labels to the dataset

data['Cluster'] = y\_kmeans

# Displaying the clustered data

print(data)