DS programs

1. **Fish.csv dataset is a record of 7 common different fish species in fish market sales.**
    a. **Train a suitable model with the data, to predict the weight of the fish.**
    b. **Visualize some insights.**

Code:

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.model_selection import train_test_split

from sklearn.linear_model import LinearRegression

from sklearn.metrics import mean_squared_error, r2_score


# Read the Fish dataset

fish = pd.read_csv('Fish.csv')


# Explore and visualize the dataset

print(fish.head())

sns.pairplot(fish)

plt.show()


# Split the data into features and target variable

X = fish.drop('Weight', axis=1)

y = fish['Weight']


# Perform one-hot encoding on the 'Species' column

X_encoded = pd.get_dummies(X, columns=['Species'])
```

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

# Train a linear regression model
model = LinearRegression()
model.fit(X_train, y_train)

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

# Evaluate the model's performance
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
print('Mean Squared Error:', mse)
print('R-squared Score:', r2)
```

2. **Diabetics.csv - dataset is originally from the National Institute of Diabetes and Digestive and Kidney Diseases. The objective of the project is to diagnostically predict whether or not a patient has diabetes, based on certain diagnostic measurements included in the dataset. Visualize some insights.**

   Code:

```
import pandas as pd

from sklearn.model_selection import train_test_split

from sklearn.neighbors import KNeighborsClassifier

from sklearn.metrics import accuracy_score, classification_report, confusion_matrix



# Load the dataset

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



# Split the dataset into features (X) and target variable (y)
```

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

# Split the 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)

# Create and fit the KNN classifier
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X_train, y_train)

# Make predictions on the test set
y_pred = knn.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)

classification_report = classification_report(y_test, y_pred)
print("Classification Report:")
print(classification_report)

confusion_matrix = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:")
print(confusion_matrix)
```

3. **This is a Glass Identification Data set from UCI. The glass contains materials like sodium(Na), Silicon(Si) etc, based on which the type of the glass is found. Prepare a suitable model for glass classification. Visualize some insights.**

Code:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score

# Read the Glass Identification dataset
glass = pd.read_csv('glass.csv')

# Explore and visualize the dataset
print(glass.head())
sns.countplot(x='Type', data=glass)
plt.title('Distribution of Glass Types')
plt.show()

# Split the data into features and target variable
X = glass.drop('Type', axis=1)
y = glass['Type']

# Split the 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)


# Train a random forest classifier

model = RandomForestClassifier()

model.fit(X_train, y_train)


# Make predictions on the test set

y_pred = model.predict(X_test)


# Evaluate the model's performance

print('Classification Report:')

print(classification_report(y_test, y_pred))

print('Confusion Matrix:')

print(confusion_matrix(y_test, y_pred))

print('Accuracy:', accuracy_score(y_test, y_pred))
```

4. **Given is a 50_startups dataset from UCI. The dataset has 4 features on which profit of the start_up is depending. Prepare a prediction model for profit of 50_startups data. Visualize some insights.**

Code:

```
import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

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
```

```
# Load the dataset
df = pd.read_csv('50_Startups.csv')

# Perform one-hot encoding for the 'State' column
encoder = OneHotEncoder(sparse=False)
encoded_states = encoder.fit_transform(df[['State']])
state_labels = encoder.categories_[0]
encoded_states_df = pd.DataFrame(encoded_states, columns=state_labels)

# Concatenate the encoded states with the original dataframe
df_encoded = pd.concat([df.drop('State', axis=1), encoded_states_df], axis=1)

# Explore the encoded dataset
print(df_encoded.head())

# Split the dataset into features (X) and the target variable (y)
X = df_encoded.drop('Profit', axis=1)
y = df_encoded['Profit']

# Split the 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)

# Create and fit the linear regression model
regressor = LinearRegression()
```

```
regressor.fit(X_train, y_train)


# Make predictions on the test set

y_pred = regressor.predict(X_test)


# Evaluate the model's performance

mse = mean_squared_error(y_test, y_pred)

r2 = r2_score(y_test, y_pred)

print("Mean Squared Error:", mse)

print("R-squared Score:", r2)
```

5. **Given is a computer dataset showing on how the price of the computers vary according to speed, ram etc. Use a suitable model to predict the price of the computer. Visualize some insights.**

```
    import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from sklearn.linear_model import LinearRegression

from sklearn.model_selection import train_test_split

from sklearn.preprocessing import OneHotEncoder

from sklearn.compose import ColumnTransformer

from sklearn.metrics import mean_squared_error


# Load the dataset

dataset = pd.read_csv('Computer_Data.csv')
```

```
# Split the dataset into features (X) and target variable (y)

X = dataset[['speed', 'hd', 'ram', 'screen', 'cd']]

y = dataset['price']


# Perform one-hot encoding on the 'cd' feature

ct = ColumnTransformer(transformers=[('encoder', OneHotEncoder(), [4])], remainder='passthrough')

X = ct.fit_transform(X)


# Split the 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)


# Create a linear regression model

model = LinearRegression()


# Fit the model to the training data

model.fit(X_train, y_train)


# Predict the prices for the test set

y_pred = model.predict(X_test)


# Calculate the root mean squared error (RMSE)

rmse = np.sqrt(mean_squared_error(y_test, y_pred))

print('Root Mean Squared Error:', rmse)


# Visualize the predicted prices vs. the actual prices
```

```
plt.scatter(y_test, y_pred)
plt.xlabel('Actual Price')
plt.ylabel('Predicted Price')
plt.title('Actual vs. Predicted Prices')
plt.show()
```

6. **A Tour & Travels Company Is Offering Travel Insurance Package to their Customers. Given TravelInsurancePrediction.csv with customer details fit a suitable model to predict the insurance will be granted or not. Visualize some insights**

Code:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.metrics import confusion_matrix, accuracy_score, roc_curve, roc_auc_score


# Load the dataset
dataset = pd.read_csv('TravelInsurancePrediction.csv')


# Split the dataset into features (X) and target variable (y)
X = dataset[['RID', 'Age', 'AnnualIncome', 'FamilyMembers', 'ChronicDiseases', 'FrequentFlyer', 'EverTravelledAbroad']]
```

```
y = dataset['TravelInsurance']

# Perform one-hot encoding on the categorical features
categorical_features = ['ChronicDiseases', 'FrequentFlyer', 'EverTravelledAbroad']
ct = ColumnTransformer(transformers=[('encoder', OneHotEncoder(), categorical_features)], remainder='passthrough')
X = ct.fit_transform(X)

# Split the 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)

# Create a logistic regression model
model = LogisticRegression()

# Fit the model to the training data
model.fit(X_train, y_train)

# Predict the travel insurance for the test set
y_pred = model.predict(X_test)

# Calculate the accuracy of the model
accuracy = accuracy_score(y_test, y_pred)
print('Accuracy:', accuracy)

# Calculate the ROC AUC score of the model
roc_auc = roc_auc_score(y_test, y_pred)
```

```
print('ROC AUC Score:', roc_auc)


# Visualize the confusion matrix

cm = confusion_matrix(y_test, y_pred)

plt.imshow(cm, cmap='Blues', interpolation='nearest')

plt.colorbar()

plt.xlabel('Predicted Labels')

plt.ylabel('True Labels')

plt.title('Confusion Matrix')

plt.xticks([0, 1])

plt.yticks([0, 1])

plt.show()
```

7. **Water_quality.csv captures the characteristics of water like pH, hardness etc based on which the quality of water has to be predicted. Fit a suitable model to predict the water quality. Visualize some insights.**

Code:

```
  import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
de1 = pd.read_csv('water_quality.csv')
print(de1.head())
print(de1.isnull().sum())
de1.ph.fillna(de1.ph.mode()[0], inplace=True)
de1.Sulfate.fillna(de1.Sulfate.mode()[0], inplace=True)
de1.Trihalomethanes.fillna(de1.Trihalomethanes.mode()[0], inplace=True)
print(de1.isnull().sum())
# # Split the data into training and testing sets
y = de1['Quality']
x = de1[['ph', 'Hardness', 'Solids', 'Chloramines',
'Sulfate','Conductivity','Organic_carbon','Trihalomethanes','Turbidity'
]]
# print(x.shape)
# print(y.shape)
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=4)
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(x_train,y_train)
feature_names = x.columns
# feature_names = X.columns
model_coefficients = regressor.coef_

coefficients_df = pd.DataFrame(data = model_coefficients,
                              index = feature_names,
                              columns = ['Coefficient value'])
print(coefficients_df)
y_pred = regressor.predict(x_test)
results = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred})
print(results)
```

8. **Placement.csv consists of data of students in a XYZ campus who are placed. It includes secondary and higher secondary school percentage and specialization. It also includes degree specialization, type and Work experience. Given new data fit a model to classify the student will be placed or**

Code:

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split

from sklearn.preprocessing import LabelEncoder

from sklearn.tree import DecisionTreeClassifier

from sklearn.metrics import confusion_matrix, accuracy_score



# Load the dataset

dataset = pd.read_csv('Placement_Data_Full_Class.csv')
```

```
# Split the dataset into features (X) and target variable (y)
X = dataset[['gender', 'ssc_p', 'hsc_p', 'degree_p', 'etest_p', 'mba_p']]
y = dataset['status']

# Convert categorical variables to numerical using LabelEncoder
le = LabelEncoder()
X['gender'] = le.fit_transform(X['gender'])

# Split the 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)

# Create a decision tree classifier
model = DecisionTreeClassifier()

# Fit the model to the training data
model.fit(X_train, y_train)

# Predict the placement status for the test set
y_pred = model.predict(X_test)

# Calculate the accuracy of the model
accuracy = accuracy_score(y_test, y_pred)
print('Accuracy:', accuracy)
```

```
# Visualize the confusion matrix

cm = confusion_matrix(y_test, y_pred)

plt.imshow(cm, cmap='Blues', interpolation='nearest')

plt.colorbar()

plt.xlabel('Predicted Labels')

plt.ylabel('True Labels')

plt.title('Confusion Matrix')

plt.xticks([0, 1])

plt.yticks([0, 1])

plt.show()
```

9. **Dream Housing Finance company deals in all home loans. They have presence across all urban, semi urban and rural areas. Customer first apply for home loan after that company validates the customer eligibility for loan. Company wants to automate the loan eligibility process (real time) based on customer detail provided while filling online application form. To automate this process, fit a suitable model which can predict whether the customer loan can be approved or not.**

Code:

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split

de1 = pd.read_csv('loan_data_set.csv')
print(de1.head())
print(de1.isnull().sum())

# Drop Loan_ID column
de1.drop('Loan_ID', inplace=True, axis=1)

# Map categorical variables to numeric values
de1.Gender = de1.Gender.map({'Male': 1, 'Female': 0})
de1.Married = de1.Married.map({'Yes': 1, 'No': 0})
de1.Education = de1.Education.map({'Graduate': 1, 'NotGraduate': 0})
de1.Self_Employed = de1.Self_Employed.map({'Yes': 1, 'No': 0})
de1.Dependents = de1.Dependents.map({'0': 0, '1': 1, '2': 2, '3+': 3})
```

```
# Drop Property_Area column
de1.drop('Property_Area', inplace=True, axis=1)

# Fill missing values with mode
de1.Gender.fillna(de1.Gender.mode()[0], inplace=True)
de1.Married.fillna(de1.Married.mode()[0], inplace=True)
de1.Dependents.fillna(de1.Dependents.mode()[0], inplace=True)
de1.Education.fillna(de1.Education.mode()[0], inplace=True)
de1.Self_Employed.fillna(de1.Self_Employed.mode()[0], inplace=True)

# Convert numeric columns to appropriate types
# de1[['Gender', 'Married', 'Dependents', 'Education',
'Self_Employed']] = de1[
#     ['Gender', 'Married', 'Dependents', 'Education', 'Self_Employed']
# ].astype('int64')

# Fill missing values in LoanAmount, Loan_Amount_Term, and
Credit_History with their respective medians
de1.LoanAmount.fillna(de1.LoanAmount.median(), inplace=True)
de1.Loan_Amount_Term.fillna(de1.Loan_Amount_Term.median(),
inplace=True)
de1.Credit_History.fillna(de1.Credit_History.median(), inplace=True)

print("\n")
print(de1.isnull().sum())

# Split the data into training and testing sets
y = de1['Loan_Status']
x = de1[['Gender', 'Married', 'Dependents', 'Education',
'Self_Employed']]
print(x.shape)
print(y.shape)
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=4)
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(n_neighbors=5)
classifier.fit(x_train, y_train)
y_pred = classifier.predict(x_test)
from sklearn.metrics import classification_report, confusion_matrix
print("Confusion matrix:::","\n",confusion_matrix(y_test, y_pred))
print("\n")
print("classification report:::","\n",classification_report(y_test,
y_pred))
```

10. **Given the sepal and petal lengths and width of the flowers, design a model that makes useful classifications to predict the class of new iris flowers.**

Code:

```
import pandas as pd
import numpy as np

# Read the Iris dataset
iris = pd.read_csv('iris.csv')

# Map the 'Species' column values to numeric labels
iris['Species'] = iris['Species'].map({'Iris-setosa': 0, 'Iris-versicolor': 1, 'Iris-virginica': 2})

# Split the data into training and testing sets
from sklearn.model_selection import train_test_split
X = iris.drop('Species', axis=1)  # Use all features except 'Species' as input
y = iris['Species']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Create a KNN classifier, fit it on the training data
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X_train, y_train)

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

```
print(y_pred)


# Evaluate the performance of the classifier
from sklearn.metrics import accuracy_score
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```