1. Write a python program to Pre-process the given data set such as cleaning, removing duplicate, outlier etc.

Problem: Employee Dataset with problems

ID	Name	Age	Salary	Department	Join Date	Bonus
1	Alice	25	50000	HR	01-01-2020	5000
2	Bob	29	60000	IT	15-05-2020	6000
3	Charlie	28	55000	IT	20-03-2021	5500
4		NaN	70000	HR	25-11-2020	7000
5	David	32		IT	30-08-2021	NaN
6	Eve	35	80000	Sales		8000
7	Alice	25	50000	HR	01-01-2020	5000
8	Frank	100	100000	HR	18-06-2022	10000
9	Grace	26	65000	IT	11-07-2021	6500
10	Charlie	28	60000	IT	20-03-2021	5500
11	Charlie	28	60000	IT	20-03-2021	6000

Task

- Display Original DataFrame
- Display DataFrame after removing rows with missing values
- Dispaly DataFrame after removing duplicate rows
- Dispaly DataFrame after removing outliers
- Dispaly Final Cleaned DataFrame

ANSWER:

```
import pandas as pd
import numpy as np

# Original data
data = {
    'ID': [1, 2, 3, 4, 5, 6, 7, 8, 9],
    'Name': ['Alice', 'Bob', 'Charlie', None, 'David', 'Eve', 'Alice', 'Frank', 'Grace'],
    'Age': [25, 29, 28, None, 32, 35, 25, 100, 26],
    'Salary': [50000, 60000, 55000, 70000, None, 80000, 50000, 100000, 65000],
    'Department': ['HR', 'IT', 'IT', 'HR', 'IT', 'Sales', 'HR', 'HR', 'IT'],
    'Join Date': ['2020-01-01', '2020-05-15', '2021-03-20', '2020-11-25', '2021-08-30', None, '2020-01-01', '2022-06-18', '2021-07-11'],
    'Bonus': [5000, 6000, 5500, 7000, None, 8000, 5000, 10000, 6500]
}
```

```
df = pd.DataFrame(data)
print("Original DataFrame:")
print(df)

# Step 1: Drop rows with missing values in key columns
df_no_missing = df.dropna(subset=['Age', 'Salary', 'Bonus'])
print("\nDataFrame after removing rows with missing Age, Salary, or Bonus:")
print(df_no_missing)

# Step 2: Remove duplicate names (keep first occurrence)
df_no_duplicates = df_no_missing.drop_duplicates(subset=["Name"])
print("\nDataFrame after removing duplicate names:")
print(df_no_duplicates)

# Step 3: Calculate IQR for Age and Salary
Q1_age, Q3_age = df_no_duplicates["Age"].quantile([0.25, 0.75])
Q1_salary, Q3_salary = df_no_duplicates["Salary"].quantile([0.25, 0.75])

IQR_age = Q3_age - Q1_age
IQR_salary = Q3_salary - Q1_salary

# Step 4: Define bounds for outlier detection
lower_age = Q1_age - 1.5 * IQR_age
upper_age = Q3_age + 1.5 * IQR_age
lower_salary = Q1_salary - 1.5 * IQR_salary
upper_salary = Q3_salary + 1.5 * IQR_salary

# Step 5: Filter out outliers
df_no_outliers = df_no_duplicates[
    (df_no_duplicates["Age"] >= lower_age) &
```

```
    (df_no_duplicates["Age"] <= upper_age) &
    (df_no_duplicates["Salary"] >= lower_salary) &
    (df_no_duplicates["Salary"] <= upper_salary)
]

print("\nDataFrame after removing outliers:")
print(df_no_outliers)

print("\nFinal Cleaned DataFrame:")
print(df_no_outliers)
```

2. Write a Python program to illustrate Gaussian Naive Bayes algorithm with minimum15 training data set and prediction for the new data.

ANSWER:

```
import numpy as np
from sklearn.naive_bayes import GaussianNB

# Sample dataset (Study hours, Past grades)
X = np.array([
 [1, 3], [2, 4], [3, 5], [4, 6], [5, 7], # Fail group
 [6, 8], [7, 9], [8, 9], [9, 10] # Pass group explain the code step by step
])

# Labels (0 = Fail, 1 = Pass)
Y = np.array([0, 0, 0, 0, 0, 1, 1, 1, 1])

# Create and train the Naive Bayes classifier
model = GaussianNB()
model.fit(X, Y)

# Predict for a new student with 6 hours of study and past grade of 9
```

```
new_student = [[7,7]]
prediction = model.predict(new_student)

print(f"The predicted class for the new student is:{'Pass'if prediction[0] == 1 else 'Fail'}")
```

3. Write a Python program to classify features as Cat or Dog using Gaussian Naive Bayes algorithm with minimum 10 training data set. Features to be considered ['mean', 'stddev'] and prediction for the new data [219.74617433, 53.87654393].

ANSWER:

```
from sklearn.naive_bayes import GaussianNB
import numpy as np

# Training data (at least 10 samples)
# Format: [mean, stddev]
X_train = [
    [210.0, 50.0],
    [205.5, 49.5],
    [215.0, 51.0],
    [199.0, 48.0],
    [202.3, 52.1],
    [230.0, 60.0],
    [250.0, 65.0],
    [240.0, 62.0],
    [245.0, 63.5],
    [235.0, 61.0]
]

# Labels for each sample ('Cat' or 'Dog')
# First 5 are 'Cat', next 5 are 'Dog'
y_train = ['Cat', 'Cat', 'Cat', 'Cat', 'Cat', 'Dog', 'Dog', 'Dog', 'Dog', 'Dog']
```

```
# New data to predict
X_test = [[219.74617433,  53.87654393]]

# Initialize and train the Gaussian Naive Bayes classifier
model = GaussianNB()
model.fit(X_train, y_train)

# Predict the class for the new data
prediction = model.predict(X_test)

print("The predicted class for the data", X_test[0], "is:", prediction[0])
```

4. Write a Python program to illustrate Linear regression algorithm. Features to be considered [year, GDP] with minimum 15 data.

ANSWER:

```
from sklearn.linear_model import LinearRegression

X =
[[2001,5.2],[2002,5.1],[2003,5.1],[2004,4.9],[2005,5.0],[2006,5.1],[2007,5.4],[2008,5.6],[2009,5
.9],[2010,5.8],[ 2011,6.2],
   [2012,6.0],[2013,5.8],[2014,6.1],[2015,6.4]]
Y = [2.5,2.52,2.54,2.48,2.52,2.54,2.55,2.7,2.9,3.2,3.16,3.28,3.2,3.15,3.26]

len(X), len(Y)
LinR_model = LinearRegression()
LinR_model.fit(X, Y)
prediction = LinR_model.predict([[2021,6.1]])
print(prediction)
prediction_2022 = LinR_model.predict([[2022,6.4]])
```

```
print(prediction_2022)
```

5. Write a Python program to illustrate Linear regression algorithm on the data features [Year, Rainfall] with minimum15 data.

Predict for the new year 2026

Display the result Year vs Rainfall in scatter plot.

ANSWER:

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression

# Data: Year vs Rainfall (Rainfall data in mm)
X = [[2001], [2002], [2003], [2004], [2005], [2006], [2007], [2008], [2009], [2010],
    [2011], [2012], [2013], [2014], [2015], [2016], [2017], [2018], [2019], [2020]]
Y = [2.5, 2.52, 2.54, 2.48, 2.52, 2.54, 2.55, 2.7, 2.9, 3.2, 3.16, 3.28, 3.2, 3.15,
    3.26, 3.29, 3.17, 3.25, 3.29, 3.18]  # Rainfall in mm corresponding to each year

# Initialize the linear regression model
LinR_model = LinearRegression()

# Fit the model with the data (Year as X and Rainfall as Y)
LinR_model.fit(X, Y)

# Predict rainfall for year 2026
prediction_2026 = LinR_model.predict([[2026]])
print(f"Predicted rainfall for the year 2026: {prediction_2026[0]:.2f} mm")

# Scatter plot of data points
plt.scatter(X, Y, color='blue', label='Data points')  # Data points
plt.plot(X, LinR_model.predict(X), color='red', label='Regression line')  # Regression line
```

```
# Plotting the prediction for 2026
plt.scatter(2026, prediction_2026, color='green', label=f'Prediction (2026):
{prediction_2026[0]:.2f} mm')

# Labels and title
plt.xlabel('Year')
plt.ylabel('Rainfall (mm)')
plt.title('Year vs Rainfall (Linear Regression)')

# Show legend and plot
plt.legend()
plt.grid(True)
plt.show()
```

6. Write a Python program to illustrate Logistic regression algorithm.

ANSWER:

```
import numpy as np
from sklearn import linear_model
import matplotlib.pyplot as plt
X = [[165,19],[175,32],[136,35],[174,65],[141,28],[176,15],[131,32],
[166,6],[128,32],[179,10],[136,34],[186,2],[126,25],[176,28],[112,38],
[169,9],[171,36],[116,25],[196,25]]
Y = ['Man','Woman','Woman','Man','Woman','Man','Woman','Man','Woman',
'Man','Woman','Man','Woman','Woman','Woman','Man','Woman','Woman','Man']
data_feature_names = ['height','age']
LR_model = linear_model.LogisticRegression()
LR_model.fit(X, Y)
prediction = LR_model.predict([[169,19]])
print(prediction)
print('Accuracy on the training subset is:',format(LR_model.score(X,Y)))
```

7. Write a Python program to illustrate SVM algorithm to predict 'Man' or 'woman' with the data features ['height','age']. Minimum training data set=20

ANSWER:

```
from sklearn.svm import SVC
data_feature_names = ['height','age'] # This is nothing but the 2 Columns of my dataset
X = [[165,19],[175,32],[136,35],[174,65],[141,28],[176,15],[131,32],
[166,6],[128,32],[179,10],[136,34],[186,2],[126,25],[176,28],[112,38],
[169,9],[171,36],[116,25],[196,25],[197,8]]
Y = ['Man','Woman','Woman','Man','Woman','Man','Woman','Man','Woman',
'Man','Woman','Man','Woman','Woman','Woman','Man','Woman','Woman','Man','Women']
SVC_model = SVC(gamma='auto')
SVC_model.fit(X, Y)
print(SVC_model.predict([[156, 53]]))
print('Accuracy on the training subset:',format(SVC_model.score(X,Y)))
```

8. Write a Python program to illustrate KNN algorithm with minimum 20 training dataset, data features (Height in cm, Weight in kg), Labels (e.g., Class 0 or 1) and predict the class for the new data [[165, 60]]

ANSWER:

```
from sklearn.neighbors import KNeighborsClassifier
import matplotlib.pyplot as plt
import numpy as np

# Sample dataset: [height (cm), weight (kg)] and class labels (0 or 1)
X = [
   [150, 45], [152, 48], [155, 50], [157, 52], [160, 54],
   [162, 55], [163, 57], [164, 58], [166, 60], [168, 62],
   [170, 65], [172, 67], [174, 70], [176, 72], [178, 75],
   [180, 78], [182, 80], [184, 83], [186, 85], [188, 88]
]
```

```
# Labels (0: Group A, 1: Group B)
y = [
    0, 0, 0, 0, 0,
    0, 0, 0, 1, 1,
    1, 1, 1, 1, 1,
    1, 1, 1, 1, 1
]

# Initialize the KNN classifier with k=3
knn = KNeighborsClassifier(n_neighbors=3)

# Train the model
knn.fit(X, y)

# New data point for prediction
new_data = [[165, 60]]
predicted_class = knn.predict(new_data)
print(f"Predicted class for {new_data[0]} is: {predicted_class[0]}")

# Optional: Visualization
X_array = np.array(X)
y_array = np.array(y)

# Scatter plot with color-coded classes
for class_value in np.unique(y_array):
    plt.scatter(
        X_array[y_array == class_value][:, 0],  # heights
        X_array[y_array == class_value][:, 1],  # weights
        label=f'Class {class_value}'
    )
```

```
# Mark the new data point
plt.scatter(new_data[0][0], new_data[0][1], color='red', marker='X', s=100, label='New Data (165, 60)')


plt.xlabel('Height (cm)')
plt.ylabel('Weight (kg)')
plt.title('KNN Classification: Height vs Weight')
plt.legend()
plt.grid(True)
plt.show()
```

9. Write a Python program to illustrate K-means algorithm for the data_features = [“Hieght”, “Age”] with minimum 20 training data. Display the the output in scatter plot.

ANSWER:

```
from sklearn.cluster import KMeans
data_features = ["Height", "Age"]
X = [[165,19],[175,32],[136,35],[174,65],[174,66],[174,67],[141,28],[176,15],[131,32],
[166,6],[128,32],[179,10],[136,34],[186,20],[126,25],[176,28],[112,38],
[169,9],[171,36],[116,25],[196,25]]
model = KMeans(n_clusters=3)
model.fit(X)
cluster_labels = model.predict(X)
print(cluster_labels)
x1 = []
x2 = []
for item in X:
 x1.append(item[0])
 x2.append(item[1])
print(x1)
print(x2)
```

```
import matplotlib.pyplot as plt
plt.scatter(x1,x2, c=model.labels_)
plt.show()
```

10. Write a Python program to illustrate Market Basket Analysis using Apriori algorithm.

ANSWER:

```
from efficient_apriori import apriori
transactions=[
 ['butter','milk','bread'],
 ['butter','milk','apple'],
 ['bread','milk','banana'],
 ['milk','bread','butter']
]
itemsets,rules=apriori(transactions,min_support=0.3,min_confidence=0.8)
print("Frequent Itemsets:")
for k,v in itemsets.items():
 print(f"Level {k}:{v}")

print("\nAssociation Rules:")
for rule in rules:
 print(rule)
```

11 Write a Python program to illustrate the working of Random Forest by loading Iris data set.

ANSWER:

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.datasets import load_iris
from sklearn.metrics import accuracy_score
import matplotlib.pyplot as plt
```

```
# Load the Iris dataset
data = load_iris()
X = data.data[:, :2]  # Only use sepal length and sepal width
y = data.target

# Split the data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Initialize Random Forest classifier
rf = RandomForestClassifier(n_estimators=100, random_state=42)
rf.fit(X_train, y_train)

# Make predictions
y_pred = rf.predict(X_test)

# Calculate the accuracy
accuracy = accuracy_score(y_test,  y_pred)
print(f"Random Forest Model Accuracy: {accuracy * 100:.2f}%")

# Visualize the result
plt.figure(figsize=(8, 6))

# Scatter plot for the data points, color-coded by predicted labels
plt.bar(['Accuracy'], [accuracy], color='skyblue')
plt.ylabel('Accuracy')
plt.title('Random Forest Classifier Accuracy')
plt.ylim(0, 1)  # Ensure the y-axis goes from 0 to 1
plt.show()
```

12. Write a Python program to illustrate the working of Decision Tree by loading Iris data set.

ANSWER:

```
import numpy as np
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn import metrics
import matplotlib.pyplot as plt
from sklearn.tree import plot_tree

# Load the Iris dataset (a simple, well-known dataset)
data = load_iris()
X = data.data  # Features
y = data.target  # Labels

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Initialize the Decision Tree Classifier
clf = DecisionTreeClassifier(random_state=42)

# Train the classifier
clf.fit(X_train, y_train)

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

# Evaluate the performance
accuracy = metrics.accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy * 100:.2f}%")
```

```
# Optionally, visualize the decision tree
plt.figure(figsize=(12, 8))

# Convert class_names to a list
class_names = data.target_names.tolist()

plot_tree(clf, filled=True, feature_names=data.feature_names, class_names=class_names)

plt.title("Decision Tree Visualization")
plt.show()
```

13. Write a python program to extract extract two simple 2D features from an image.

ANSWER:

```
import cv2
import numpy as np

def extract_two_features(image):
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    mean = np.mean(gray)
    stddev = np.std(gray)
    return np.array([mean, stddev])

def load_and_extract_features(image_path, label):
    img = cv2.imread(image_path)
    if img is not None:
        features = extract_two_features(img)
        print(f"Features for {'Cat' if label == 1 else 'Dog'} image ({image_path}):", features)
        return features, label
    else:
```

```
        print(f"Error loading image: {image_path}")
        return None, None

# Provide your image paths
cat_image_path = r"C:\Users\HP\OneDrive\Desktop\pexels-kmerriman-20787.jpg"
dog_image_path = r"C:\Users\HP\OneDrive\Desktop\Cute_dog.jpg"

# Extract features and labels
cat_features, cat_label = load_and_extract_features(cat_image_path, label=1)
dog_features, dog_label = load_and_extract_features(dog_image_path, label=0)

# Combine into arrays
features = np.array([cat_features, dog_features])
labels = np.array([cat_label, dog_label])

print("\n ✅ 2D features for both Cat and Dog images have been extracted and displayed.")
```

**********END**********