**LABPROGRAM 1: Develop a program to create histogram for all numerical features and analyze the distribution of each feature. Generate box plots for all numerical features and identify any outliers. Use California housing dataset.**

**PROGRAM:**

import seaborn as sns

import matplotlib.pyplot as plt

from sklearn.datasets import fetch\_california\_housing

import numpy as np

**# Step 1: Load the California Housing dataset**

data = fetch\_california\_housing(as\_frame=True)

housing\_df = data.frame

**# Step 2: Create histograms for numerical features**

numerical\_features = housing\_df.select\_dtypes(include=[np.number]).columns

# Plot histograms in a single figure

plt.figure(figsize=(15, 10))

plt.suptitle("Histograms of Numerical Features", fontsize=16)

for i, feature in enumerate(numerical\_features, 1):

plt.subplot(3, 3, i)

sns.histplot(housing\_df[feature], kde=True, bins=30, color='blue')

plt.title(f'Distribution of {feature}')

plt.tight\_layout()

plt.show() # Display all histograms at once

**# Step 3: Generate box plots for numerical features**

plt.figure(figsize=(15, 10))

plt.suptitle("Box Plots of Numerical Features", fontsize=16)

for i, feature in enumerate(numerical\_features, 1):

plt.subplot(3, 3, i)

sns.boxplot(x=housing\_df[feature], color='orange')

plt.title(f'Box Plot of {feature}')

plt.tight\_layout()

plt.show() # Display all box plots at once

**# Step 4: Identify outliers using the IQR method**

print("Outliers Detection:")

outliers\_summary = {}

for feature in numerical\_features:

Q1 = housing\_df[feature].quantile(0.25)

Q3 = housing\_df[feature].quantile(0.75)

IQR = Q3 - Q1

lower\_bound = Q1 - 1.5 \* IQR

upper\_bound = Q3 + 1.5 \* IQR

outliers = housing\_df[(housing\_df[feature] < lower\_bound) | (housing\_df[feature] > upper\_bound)]

outliers\_summary[feature] = len(outliers)

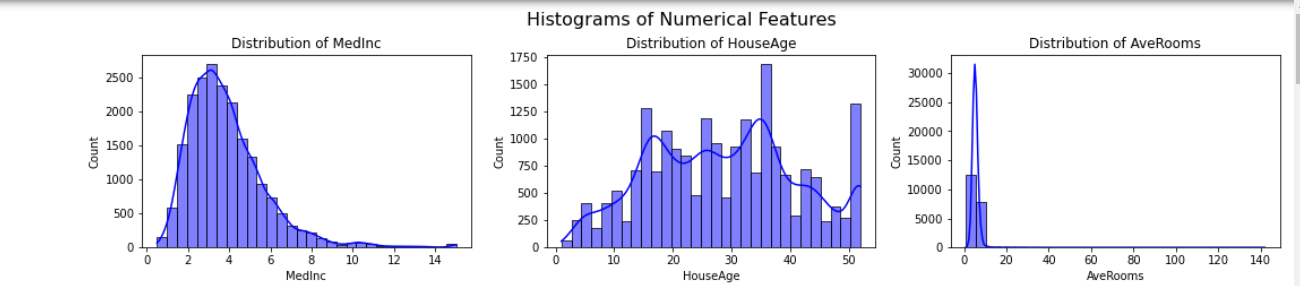
print(f"{feature}: {len(outliers)} outliers")

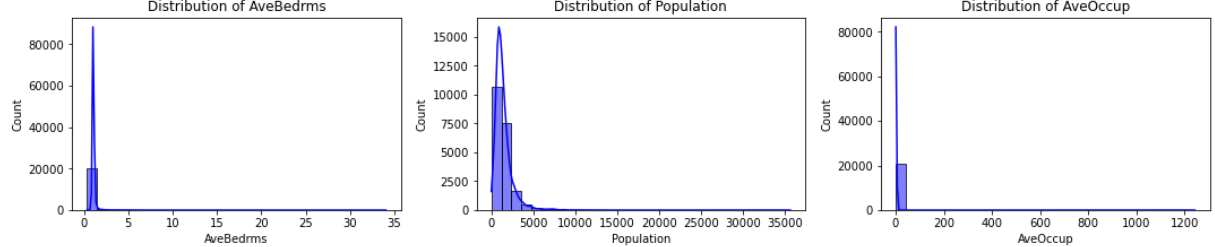
**# Step 5: Print a summary of the dataset**

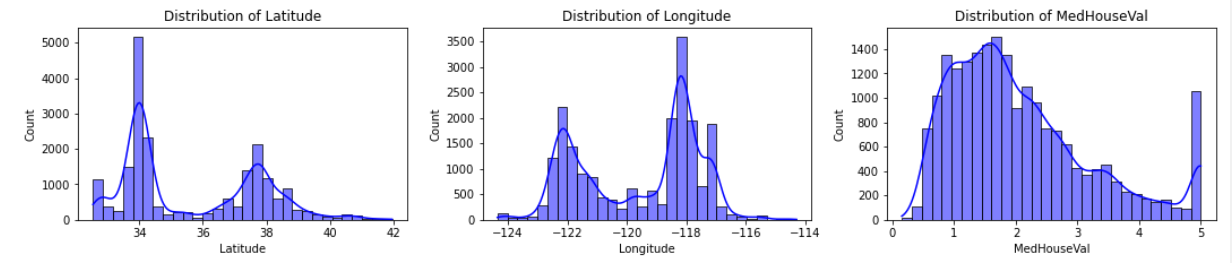
print("\nDataset Summary:")

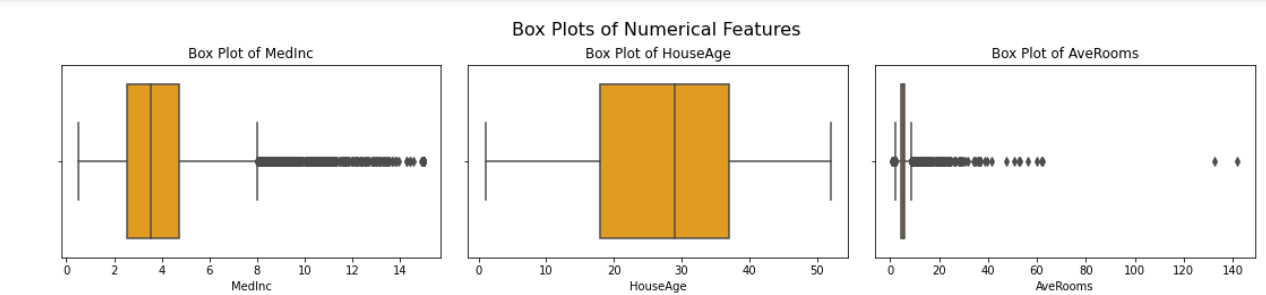
print(housing\_df.describe())

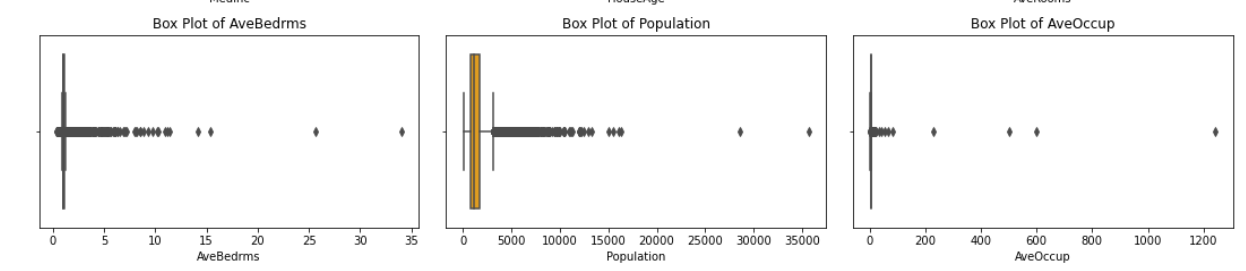
**OUTPUT:**

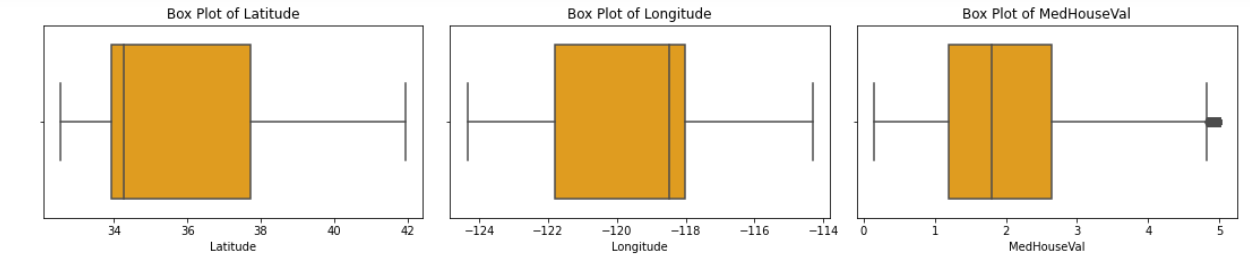


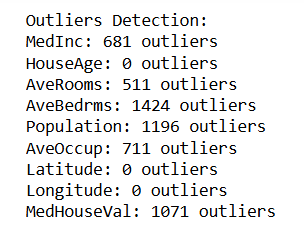
****

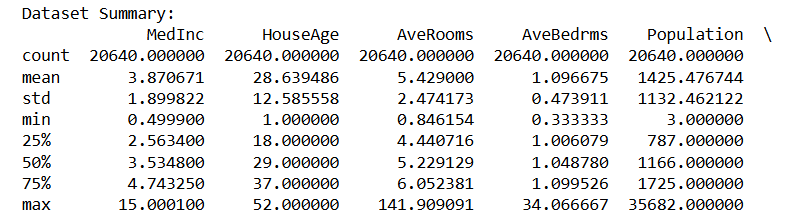
****

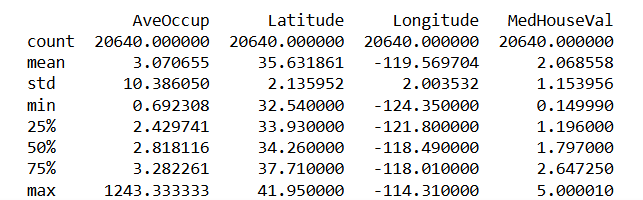
****

****

****

****

****

****

**LABPROGRAM 2: Develop a program to Compute the correlation matrix to understand the relationships between pairs of features. Visualize the correlation matrix using a heatmap to know which variables have strong positive/negative correlations. Create a pair plot to visualize pairwise relationships between features. Use California Housing dataset.**

**import pandas as pd**

**import seaborn as sns**

**import matplotlib.pyplot as plt**

**from sklearn.datasets import fetch\_california\_housing**

**# Step 1: Load the California Housing Dataset**

**california\_data = fetch\_california\_housing(as\_frame=True)**

**data = california\_data.frame**

**# Step 2: Compute the correlation matrix**

**correlation\_matrix = data.corr()**

**# Step 3: Visualize the correlation matrix using a heatmap**

**plt.figure(figsize=(10, 8))**

**sns.heatmap(correlation\_matrix, annot=True, cmap='coolwarm', fmt='.2f', linewidths=0.5)**

**plt.title('Correlation Matrix of California Housing Features')**

**plt.show()**

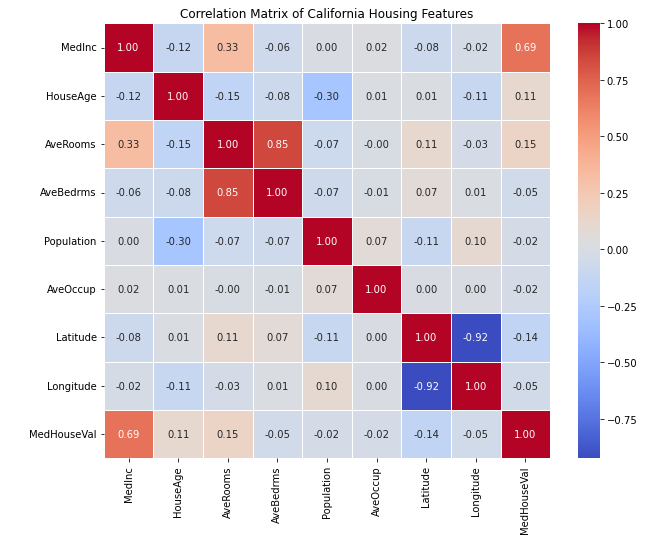
**# Step 4: Create a pair plot to visualize pairwise relationships**

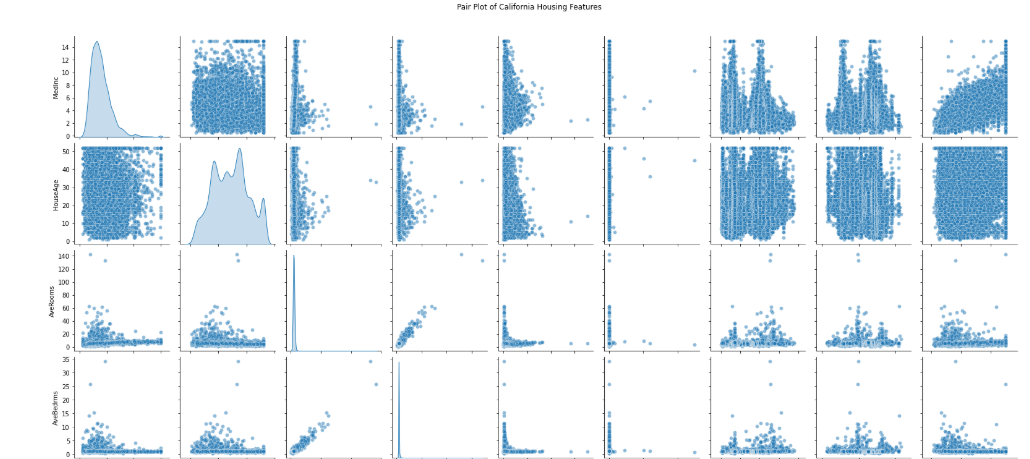
**sns.pairplot(data, diag\_kind='kde', plot\_kws={'alpha': 0.5})**

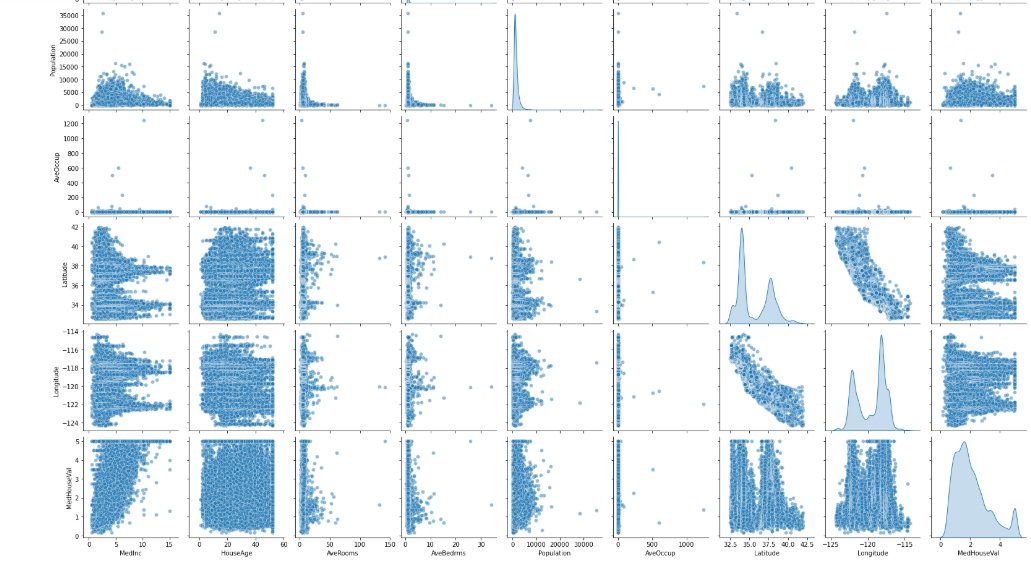
**plt.suptitle('Pair Plot of California Housing Features', y=1.02)**

**plt.show()**

**OUTPUT:**

****

****

****

**LAB PROGRAM 3: Develop a program to implement Principal Component Analysis (PCA) for reducing the dimensionality of the Iris dataset from 4 features to 2.**

**import numpy as np**

**import pandas as pd**

**from sklearn.datasets import load\_iris**

**from sklearn.decomposition import PCA**

**import matplotlib.pyplot as plt**

**# Load the Iris dataset**

**iris = load\_iris()**

**data = iris.data**

**labels = iris.target**

**label\_names = iris.target\_names**

**# Convert to a DataFrame for better visualization**

**iris\_df = pd.DataFrame(data, columns=iris.feature\_names)**

**# Perform PCA to reduce dimensionality to 2**

**pca = PCA(n\_components=2)**

**data\_reduced = pca.fit\_transform(data)**

**# Create a DataFrame for the reduced data**

**reduced\_df = pd.DataFrame(data\_reduced, columns=['Principal Component 1', 'Principal Component 2'])**

**reduced\_df['Label'] = labels**

**# Plot the reduced data**

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

**colors = ['r', 'g', 'b']**

**for i, label in enumerate(np.unique(labels)):**

**plt.scatter(**

**reduced\_df[reduced\_df['Label'] == label]['Principal Component 1'],**

**reduced\_df[reduced\_df['Label'] == label]['Principal Component 2'],**

**label=label\_names[label],**

**color=colors[i]**

**)**

**plt.title('Scatter Plot - PCA on Iris Dataset')**

**plt.xlabel('Principal Component 1')**

**plt.ylabel('Principal Component 2')**

**plt.legend()**

**plt.grid()**

**plt.show()**

**# Explained Variance by each Principal Component**

**explained\_variance = pca.explained\_variance\_ratio\_**

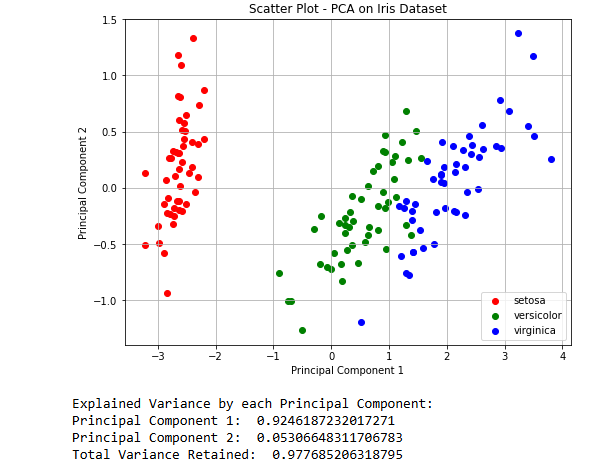
**print("Explained Variance by each Principal Component:")**

**print("Principal Component 1: ",explained\_variance[0])**

**print("Principal Component 2: ",explained\_variance[1])**

**print("Total Variance Retained: ",sum(explained\_variance))**

**OUTPUT:**

****