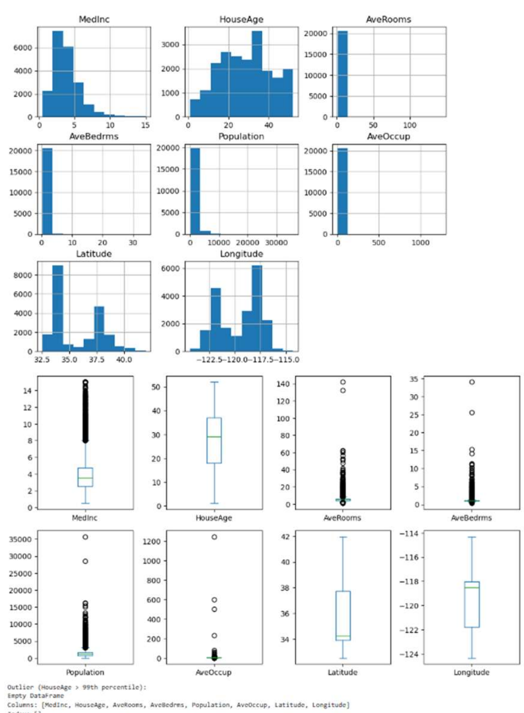
ML LAB PROGRAM’S

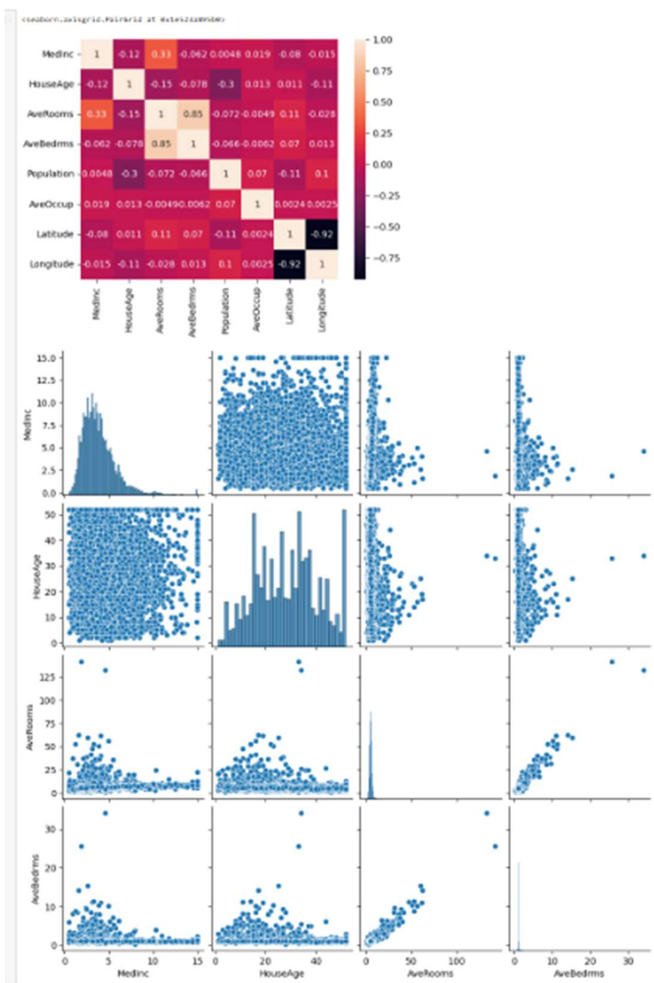
1. ***Develop a program to create histograms for all numerical features and analyze the distribution of each feature. Generate box plots for all numerical features and identify any outliers. Use California Housing dataset.***

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.datasets import fetch_california_housing

data = fetch_california_housing()
df = pd.DataFrame(data.data, columns=data.feature_names)

df.hist(figsize=(10,8))
df.plot(kind='box', subplots=True, layout=(2,4), figsize=(10,6))

plt.tight_layout()
plt.show()

print("Outlier (HouseAge > 99th percentile):")
print(df[df['HouseAge'] > df['HouseAge'].quantile(0.99)])
```



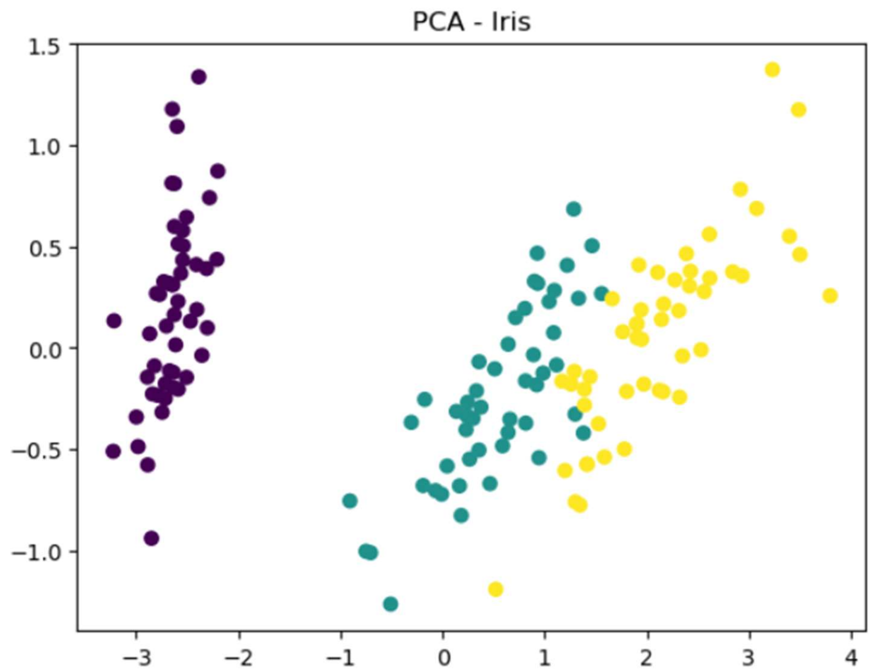
2. ***Develop a program to Compute the correlation matrix to understand the relationships between pairs of features. Visualize the correlation matrix using a heatmap to know which variables have strong positive/negative correlations. Create a pair plot to visualize pairwise relationships between features.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

```
import seaborn as sns
import pandas as pd
from sklearn.datasets import fetch_california_housing
data = fetch_california_housing()
df = pd.DataFrame(data.data, columns=data.feature_names)
sns.heatmap(df.corr(), annot=True)
sns.pairplot(df.iloc[:, :4])
```



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

```
from sklearn.decomposition import PCA
from sklearn.datasets import load_iris
import matplotlib.pyplot as plt
iris = load_iris()
result = PCA(n_components=2).fit_transform(iris.data)
plt.scatter(result[:,0], result[:,1], c=iris.target)
plt.title("PCA - Iris")
plt.show()
```



4. ***For a given set of training data examples stored in a .CSV file, implement and demonstrate the Find-S algorithm to output a description of the set of all hypotheses consistent with the training examples.***

```
import pandas as pd
data = pd.read_csv("data.csv")
print("Training Data:\n", data, "\n")
h = ['0']*len(data.columns[:-1])
for _, row in data.iterrows():
    if row[-1] == 'Yes':
        for j, col in enumerate(data.columns[:-1]):
            if h[j] == '0': h[j] = row[col]
            elif h[j] != row[col]: h[j] = '?'
print("Final hypothesis:", h)
```

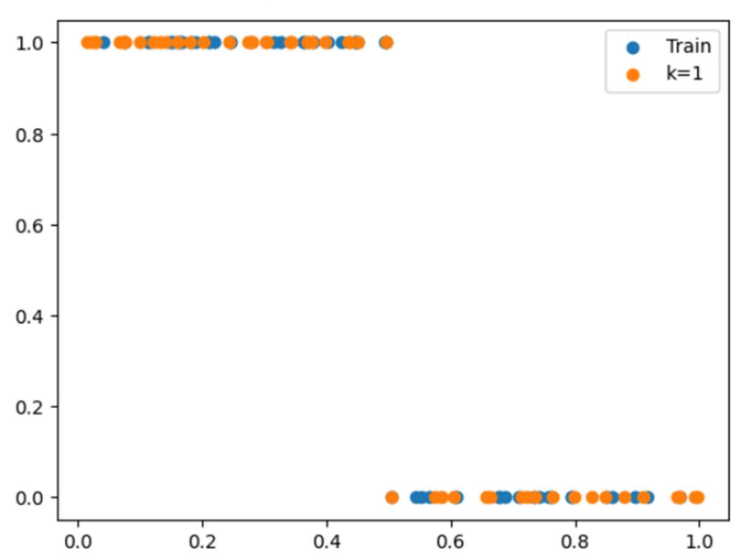
```
Training Data:
     Outlook Temperature Humidity  Windy PlayTennis
0     Sunny         Hot     High  False         No
1     Sunny         Hot     High   True         No
2  Overcast         Hot     High  False        Yes
3      Rain        Cold     High  False        Yes
4      Rain        Cold     High   True         No
5  Overcast         Hot     High   True        Yes
6     Sunny         Hot     High  False         No

Final hypothesis: ['?', '?', 'High', '?']
```

5. ***Develop a program to implement k-Nearest Neighbour algorithm to classify the randomly generated 100 values of x in the range of [0,1]. Perform the following based on dataset generated.***

   a. ***Label the first 50 points {x1,……,x50} as follows: if (xi ≤ 0.5), then xi ∊ Class1, else xi ∊ Class1***
   b. ***Classify the remaining points, x51,……,x100 using KNN. Perform this for k=1,2,3,4,5,20,30***

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

x = np.random.rand(100,1)
y = np.array([1 if xi <= 0.5 else 2 for xi in x[:50].flatten()])  # Class1=1, Class2=2

for k in [1,2,3,4,5,20,30]:
    knn = KNeighborsClassifier(n_neighbors=k).fit(x[:50], y)
    p = knn.predict(x[50:])
    print(f'k={k} Predicted:', p)
    plt.scatter(x[:50], y, label='Train')
    plt.scatter(x[50:], p, label=f'k={k}')
    plt.legend()
    plt.title(f'K={k}')
    plt.show()
```

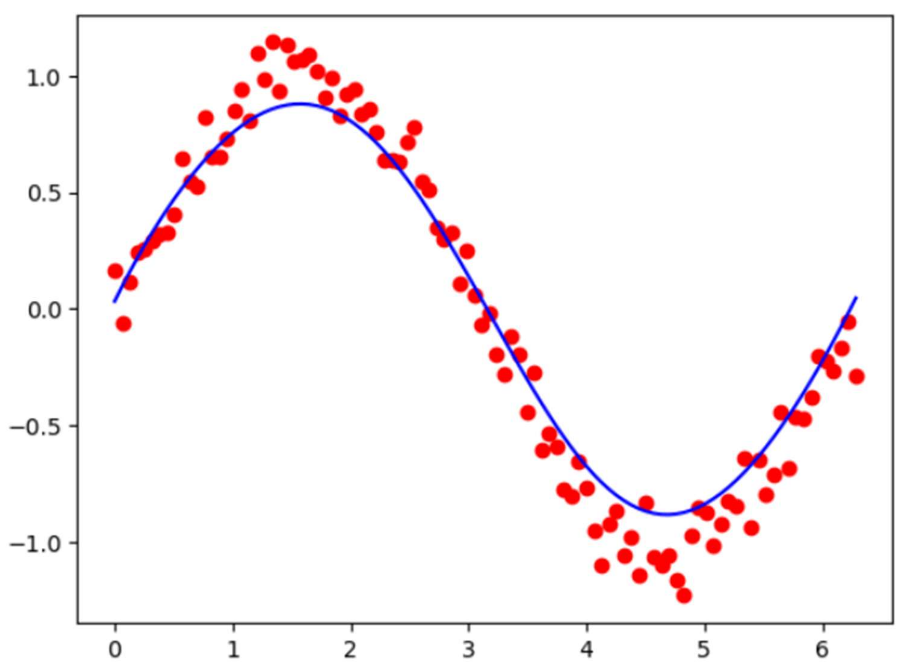
```
 1 1 1 0 0 1 1 1 1 1 1 0 1]
```




```
k=2 Predicted: [0 0 1 1 0 0 0 1 1 1 1 1 1 1 0 0 1 1 0 1 1 0 0 1 0 0 1 1 1 0 0 1 0 1 1 1 0
 1 1 1 0 0 1 1 1 1 1 1 0 1]
```

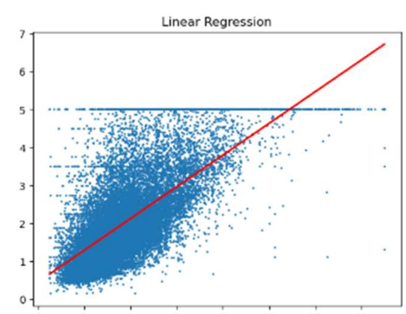
6. Implement the non-parametric Locally Weighted Regression algorithm in order to fit data points. Select appropriate data set for your experiment and draw graphs.

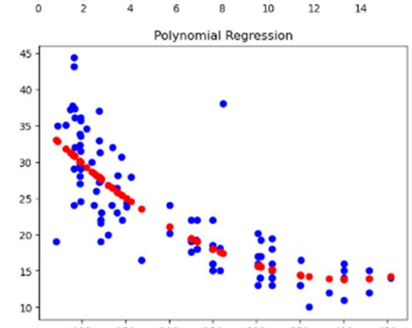
```
import numpy as np
import matplotlib.pyplot as plt

x = np.linspace(0, 2 * np.pi, 100)
y = np.sin(x) + np.random.normal(0, 0.1, 100)

def predict(x0):
    w = np.exp(-(x - x0)**2 / (2 * 0.5**2))
    W = np.diag(w)
    X = np.c_[np.ones(len(x)), x]
    theta = np.linalg.pinv(X.T @ W @ X) @ X.T @ W @ y
    return np.array([1, x0]) @ theta

# Compute predictions
y_pred = np.array([predict(x0) for x0 in x])

# Plot
plt.scatter(x, y, c='r')
plt.plot(x, y_pred, c='b')
plt.show()
```



7. ***Develop a program to demonstrate the working of Linear Regression and Polynomial Regression. Use Boston Housing Dataset for Linear Regression and Auto MPG Dataset (for vehicle fuel efficiency prediction) for Polynomial Regression.***

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.datasets import fetch_california_housing
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures
from sklearn.pipeline import make_pipeline

X, y = fetch_california_housing(return_X_y=True)
model = LinearRegression().fit(X[:, [0]], y)
plt.scatter(X[:, 0], y, s=1)
plt.plot(X[:, 0], model.predict(X[:, [0]]), c='r')
plt.title("Linear Regression")
plt.show()

url = "https://archive.ics.uci.edu/ml/machine-learning-databases/auto-mpg/auto-mpg.data"
columns = ['mpg','cyl','disp','hp','wt','acc','yr','ori','name']
df = pd.read_csv(url, names=columns, sep=r'\s+', na_values='?').dropna()  # updated sep

Xv2 = df[['hp']].astype(float).values
yv2 = df['mpg'].values

poly_model = make_pipeline(PolynomialFeatures(degree=2), LinearRegression())
poly_model.fit(Xv2, yv2)
yp2 = poly_model.predict(Xv2)

plt.scatter(Xv2, yv2, c='b', s=10, label='Actual')
plt.scatter(Xv2, yp2, c='r', s=10, label='Predicted')
plt.title("Polynomial Regression")
plt.xlabel("Horsepower")
plt.ylabel("MPG")
plt.legend()
plt.show()
```

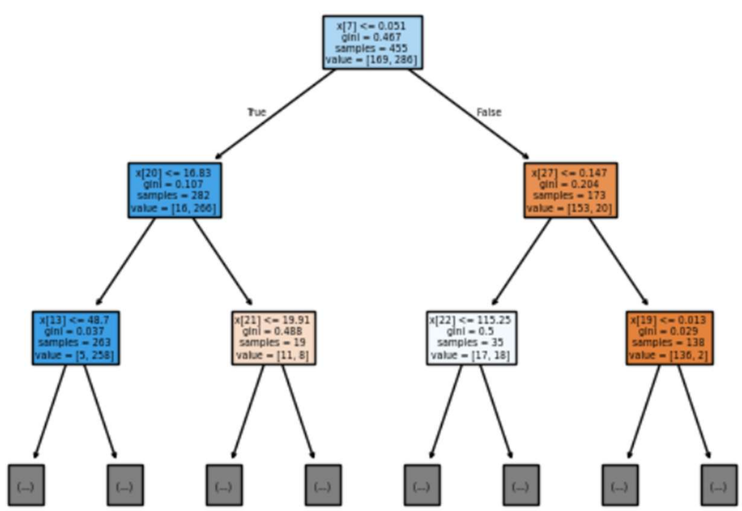




8. Develop a program to demonstrate the working of the decision tree algorithm. Use Breast Cancer Data set for building the decision tree and apply this knowledge to classify a new sample.

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

X, y = load_breast_cancer(return_X_y=True)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
clf = DecisionTreeClassifier(max_depth=4, random_state=42)
clf.fit(X_train, y_train)
plot_tree(clf, max_depth=2, filled=True)
new_sample = np.array([X_test[0]])
prediction = clf.predict(new_sample)
print("Prediction for new sample:", "Malignant" if prediction[0] == 0 else "Benign")
print("Accuracy: {:.2f}%".format(clf.score(X_test, y_test) * 100))
plt.show()
```

```
Prediction for new sample: Malignant
Accuracy: 93.86%
```



9. Develop a program to implement the Naive Bayesian classifier considering Olivetti Face Data set for training. Compute the accuracy of the classifier, considering a few test data sets.

```
import matplotlib.pyplot as plt
from sklearn.datasets import fetch_olivetti_faces
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score, classification_report,
confusion_matrix

X, y = fetch_olivetti_faces(return_X_y=True)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=42)

model = GaussianNB().fit(X_train, y_train)

y_pred = model.predict(X_test)

print(f'Accuracy: {accuracy_score(y_test,
y_pred)*100:.2f}%\n\nClassification
Report:\n{classification_report(y_test, y_pred,
zero_division=1)}\nConfusion Matrix:\n{confusion_matrix(y_test,
y_pred)}\n\nCross-validation accuracy: {cross_val_score(model, X, y,
cv=5).mean()*100:.2f}%')

fig, ax = plt.subplots(3, 5, figsize=(12, 8)); [a.imshow(i.reshape(64, 64),
cmap='gray') or a.set_title(f"T:{t},P:{p}") or a.axis('off') for a, i, t, p in
zip(ax.ravel(), X_test, y_test, y_pred)]

plt.show()
```