PES Institute of Technology and Management

Department of Computer Science & Engineering

Laboratory Manual

Semester: VI

Subject: Machine Learning Lab

Subject Code: BCSL606

Compiled by

Mr. Sunilkumar H R
Assistant Professor, Dept. of CS&E,
PESITM, Shivamogga 577204

Verified by

Dr. Arjun U
Associate Professor and Head
Dept. of CS&E, PESITM, Shivamogga 577204

NH-206, Sagar Road, Shivamogga-577204

Ph: 08182-640733/640734 Fax: 08182-233797

www.pestrust.edu.in/pesitm

	Machine Learning lab			**Semester**	**6**
Course Code			**BCSL606**	**CIE Marks**	**50**
Teaching Hours/Week (L:T:P: S)		**0:0:2:0**		**SEE Marks**	**50**
Credits		**01**		**Exam Hours**	**100**
Examination type (SEE)		**Practical**			

Course objectives:

1. **To become familiar with data and visualize univariate, bivariate, and multivariate data using statistical techniques and dimensionality reduction.**
2. **To understand various machine learning algorithms such as similarity-based learning, regression, decision trees, and clustering.**
3. **To familiarize with learning theories, probability-based models and developing the skills required for decision-making in dynamic environments.**

Sl.NO	Experiments
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.** **Book 1: Chapter 2**
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.** **Book 1: Chapter 2**
3	**Develop a program to implement Principal Component Analysis (PCA) for reducing the dimensionality of the Iris dataset from 4 features to 2.** **Book 1: Chapter 2**
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.** **Book 1: Chapter 3**
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.** 1. **Label the first 50 points *[x1,. ,xsoj* aS follows: if (xi s 0.5), then x; e Classi, else x, e Classi** 2. **Classify the remaining points, *X51,.....,xi oo* using KNN. Perform this for *k——1,2,3,4,5,20,30*** **Book 2: Chapter - 2**
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** **Book 1: Chapter — 4**
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.** **Book 1: Chapter — 5**
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.** **Book 2: Chapter — 3**

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.** **Book 2: Chapter — 4**
10	**Develop a program to implement k-means clustering using Wisconsin Breast Cancer data set and visualize the clustering result.** **Book 2: Chapter - 4**

Course outcomes (Course Skill Set):

At the end of the course the student will be able to:

1. **Illustrate the principles of multivariate data and apply dimensionality reduction techniques.**
2. **Demonstrate similarity-based learning methods and perform regression analysis.**
3. **Develop decision trees for classification and regression problems, and Bayesian models for probabilistic learning.**
1. **Implement the clustering algorithms to share computing resources.**

Assessment Details (both CIE and SEE)

The weightage of Continuous Internal Evaluation (CIE) is 50% and for Semester End Exam (SEE) is 50%. The minimum passing mark for the CIE is 40% of the maximum marks (20 marks out of 50) and for the SEE minimum passing mark is 35% of the maximum marks (18 out of 50 marks). A student shall be deemed to have satisfied the academic requirements and earned the credits allotted to each subject/ course if the student secures a minimum of 40% (40 marks out of 100) in the sum total of the CIE (Continuous Internal Evaluation) and SEE (Semester End Examination) taken together

Continuous Internal Evaluation (CIE):

CIE marks for the practical course are 50 Marks.

The split-up of CIE marks for record/ journal and test are in the ratio 60:40.

Each experiment is to be evaluated for conduction with an observation sheet and record write-up. Rubrics for the evaluation of the journal/write-up for hardware/software experiments are designed by the faculty who is handling the laboratory session and are made known to students at the beginning of the practical session.

Record should contain all the specified experiments in the syllabus and each experiment write-up will be evaluated for 10 marks.

Total marks scored by the students are scaled down to 30 marks (60% of maximum marks).

Weightage to be given for neatness and submission of record/write-up on time.

Department shall conduct a test of 100 marks after the completion of all the experiments listed in the syllabus.

In a test, test write-up, conduction of experiment, acceptable result, and procedural knowledge will carry a weightage of 60% and the rest 40% for viva-voce.

The suitable rubrics can be designed to evaluate each student's performance and learning ability. The marks scored shall be scaled down to 20 marks (40% of the maximum marks).

The Sum of scaled-down marks scored in the report write-up/journal and marks of a test is the total CIE marks scored by the student.

Semester End Evaluation (SEE):

1. **SEE marks for the practical course are 50 Marks.**
2. **SEE shall be conducted jointly by the two examiners of the same institute, examiners are appointed by the Head of the Institute.**
3. **The examination schedule and names of examiners are informed to the university before the conduction of the examination. These practical examinations are to be conducted between the schedule mentioned in the academic calendar of the University.**
4. **All laboratory experiments are to be included for practical examination.**

5. **(Rubrics) Breakup of marks and the instructions printed on the cover page of the answer script to be strictly adhered to by the examiners. OR based on the course requirement evaluation rubrics shall be decided jointly by examiners.**

6. **Students can pick one question (experiment) from the questions lot prepared by the examiners jointly.**

7. **Evaluation of test write-up/ conduction procedure and result/viva will be conducted jointly by examiners.**

General rubrics suggested for SEE are mentioned here, writeup-20%, Conduction procedure and result in -60%, Viva-voce 20% of maximum marks. SEE for practical shall be evaluated for 100 marks and scored marks shall be scaled down to 50 marks (however, based on course type, rubrics shall be decided by the examiners)

Change of experiment is allowed only once and 15% of Marks allotted to the procedure part are to be made zero.

The minimum duration of SEE is 02 hours

Suggested Learning Resources:
Books:

1. **S Sridhar and M Vijayalakshmi, "Machine Learning", Oxford University Press, 2021.**
2. **M N Murty and Ananthanarayana V S, "Machine Learning: Theory and Practice", Universities Press (India) Pvt. Limited, 2024.**

Web links and Video Lectures (e-Resources):

1. **https://www.drssridhar.com/?page_id=1053**
2. **https://www.universitiespress.com/resources?id=9789393330697**
3. **https://onlinecourses.nptel.ac.in/noc23_cs18/preview**

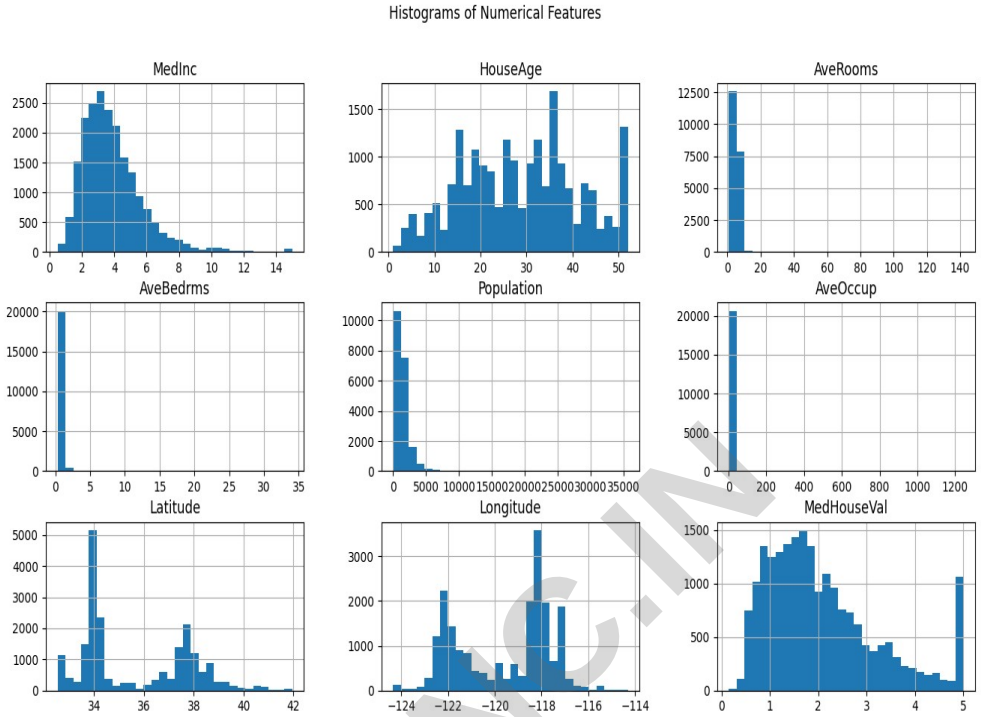
1. Develop a program to create histograms for all numerical features and analyze the distribution of each feature. Generate box plots for all numerical features and identify any outliers. Use California Housing dataset.

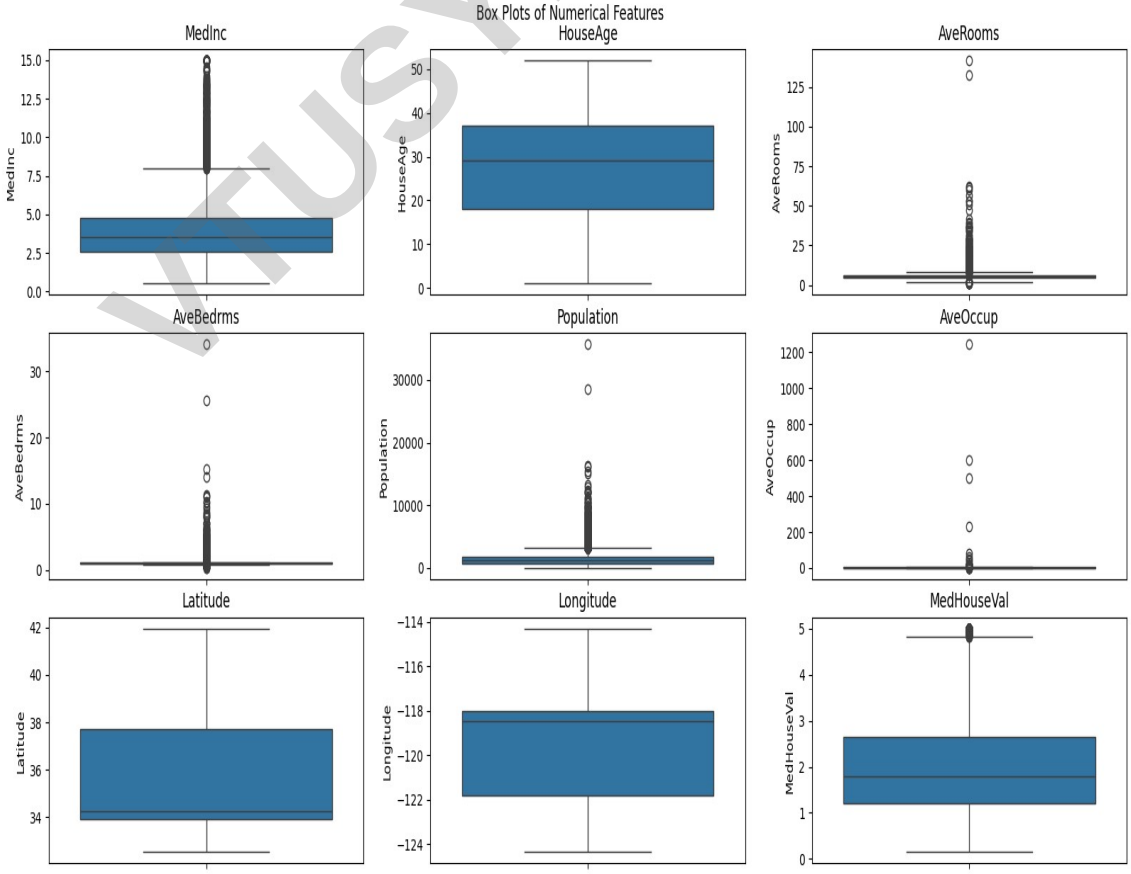
Python Code:

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

# Load the dataset
california_housing = fetch_california_housing()
data = pd.DataFrame(california_housing.data,
columns=california_housing.feature_names)
data['MedHouseVal'] = california_housing.target  # Adding the target variable to
the dataframe

# Plot histograms for all numerical features
data.hist(bins=30, figsize=(15, 10))
plt.suptitle('Histograms of Numerical Features')
plt.show()

# Plot box plots for all numerical features
plt.figure(figsize=(15, 10))
for i, column in enumerate(data.columns):
    plt.subplot(3, 3, i+1)
    sns.boxplot(y=data[column])
    plt.title(column)
plt.suptitle('Box Plots of Numerical Features')
plt.tight_layout()
plt.show()

# Identify outliers using the IQR method
Q1 = data.quantile(0.25)
Q3 = data.quantile(0.75)
IQR = Q3 - Q1

outliers = ((data < (Q1 - 1.5 * IQR)) | (data > (Q3 + 1.5 * IQR))).sum()
print("Number of outliers in each feature:")
print(outliers)
```

Output :

Number of outliers in each feature:
MedInc 681
HouseAge 0
AveRooms 511
AveBedrms 1424
Population 1196
AveOccup 711
Latitude 0
Longitude 0
MedHouseVal 1071
dtype: int64


Histograms of Numerical Features
MedInc
HouseAge
AveRooms
AveBedrms
Population
AveOccup
Latitude
Longitude
MedHouseVal

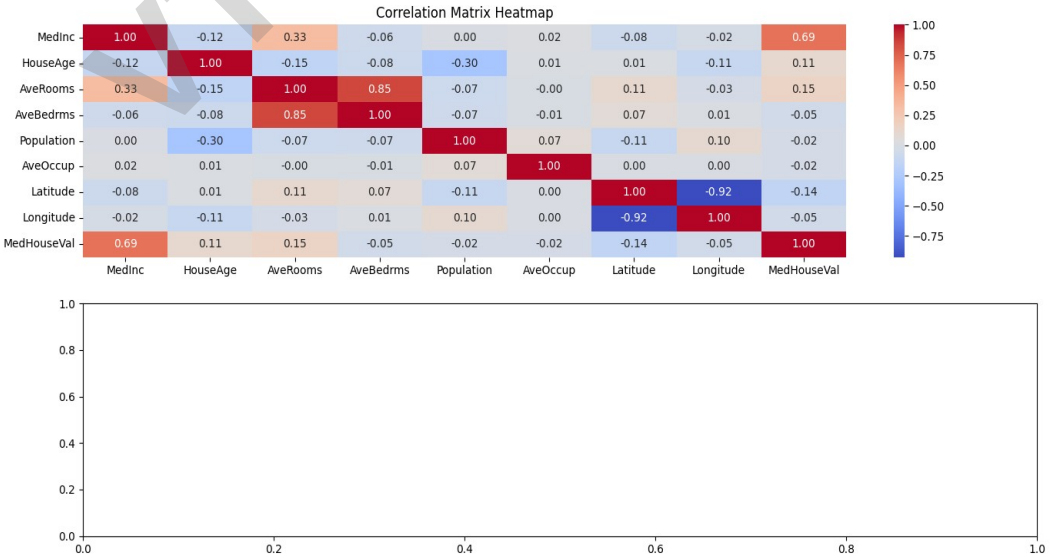

Box Plots of Numerical Features
MedInc
HouseAge
AveRooms
AveBedrms
Population
AveOccup
Latitude
Longitude
MedHouseVal

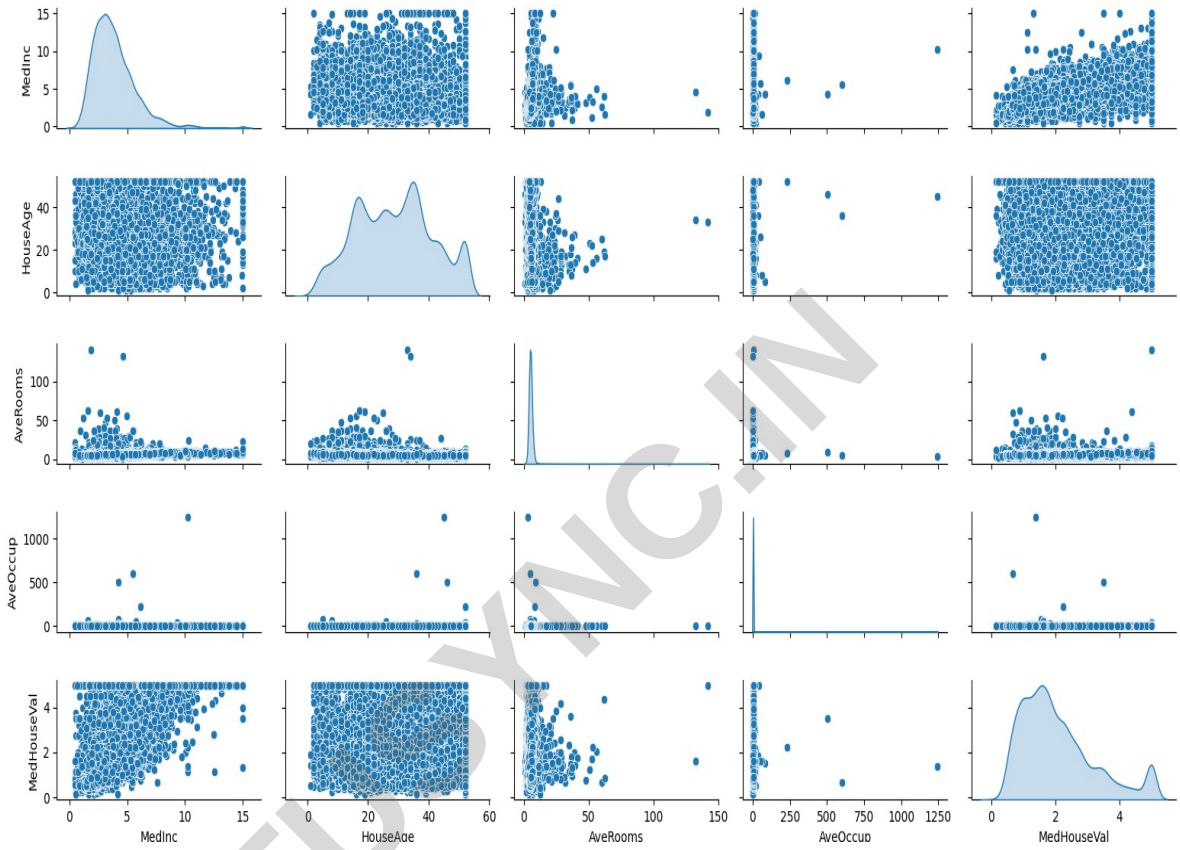
2. Develop a program to compute the correlation matrix to understand the relationships between pairs of features. Visualize the correlation matrix using a heatmap to know which variables have strong positive/negative correlations. Create a pair plot to visualize pairwise relationships between features.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

Python Code:

```
import matplotlib
matplotlib.use('TkAgg')  # Use TkAgg backend
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.datasets import fetch_california_housing

# Load the dataset
df = fetch_california_housing(as_frame=True).frame

# Set up the grid layout for plots (2 rows, 1 column)
fig, axes = plt.subplots(2, 1, figsize=(12, 12))

# Heatmap of correlation matrix (Top plot)
sns.heatmap(df.corr(), annot=True, fmt=".2f", cmap="coolwarm", cbar=True,
ax=axes[0])
axes[0].set_title("Correlation Matrix Heatmap")

# Pair plot for selected features (Bottom plot)
sns.pairplot(df[['MedInc', 'HouseAge', 'AveRooms', 'AveOccup', 'MedHouseVal']],
diag_kind="kde")
plt.subplots_adjust(hspace=0.4)  # Adjust space between subplots

# Show the combined figure
plt.show()
```

Output:



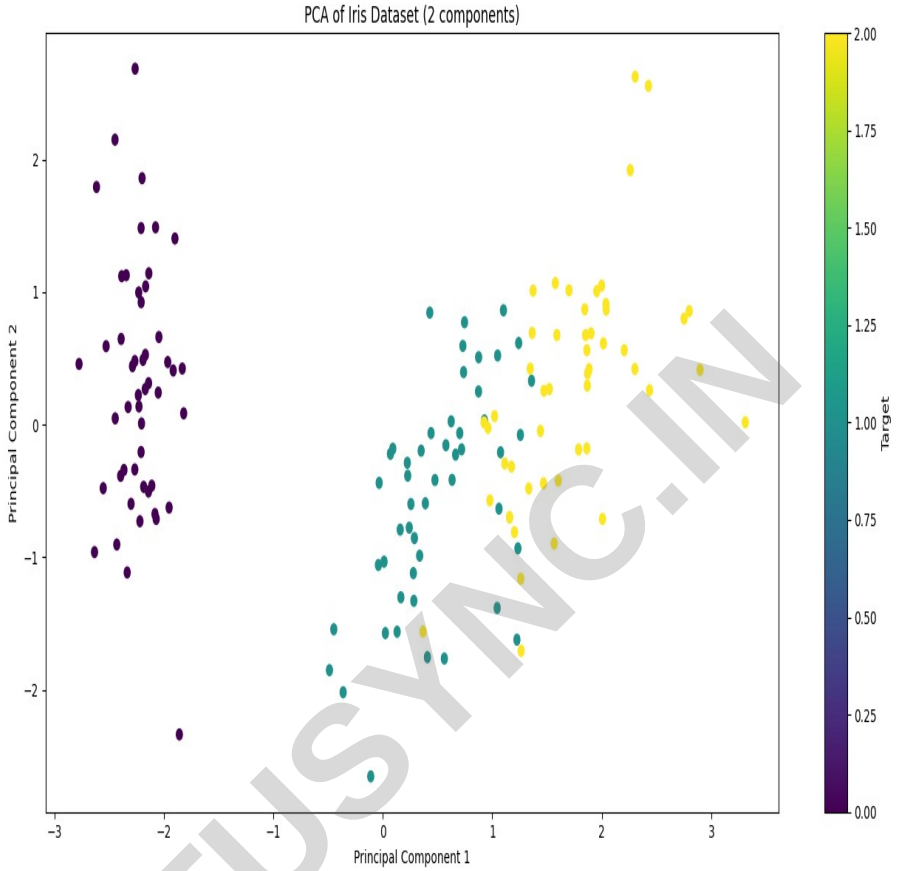

MedInc
HouseAge
AveRooms
AveOccup
MedHouseVal
MedInc
HouseAge
AveRooms
AveOccup
MedHouseVal

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

Python Code:

```
import matplotlib
matplotlib.use('TkAgg')  # Use TkAgg backend
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
from sklearn.datasets import load_iris
from sklearn.preprocessing import StandardScaler

# Load the Iris dataset
iris = load_iris()
df = pd.DataFrame(iris.data, columns=iris.feature_names)

# Standardize the features (important for PCA)
scaler = StandardScaler()
scaled_data = scaler.fit_transform(df)

# Apply PCA to reduce to 2 dimensions
pca = PCA(n_components=2)
pca_result = pca.fit_transform(scaled_data)

# Create a DataFrame for the 2 principal components
pca_df = pd.DataFrame(pca_result, columns=['PC1', 'PC2'])

# Visualize the result
plt.figure(figsize=(8, 6))
plt.scatter(pca_df['PC1'], pca_df['PC2'], c=iris.target, cmap='viridis')
plt.title("PCA of Iris Dataset (2 components)")
plt.xlabel("Principal Component 1")
plt.ylabel("Principal Component 2")
plt.colorbar(label='Target')
plt.show()
```

Output:



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.

Python Code:

```
mport pandas as pd

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

# Assume the last column is the class (target variable)
X = df.iloc[:, :-1]  # Features (all columns except the last)
y = df.iloc[:, -1]   # Class (the last column)

# Find-S algorithm
def find_s_algorithm(X, y):
    # Initialize the hypothesis to the most general hypothesis (all attributes can
be anything)
    hypothesis = ['?' for _ in range(X.shape[1])]

    # Loop through all examples in the dataset
    for i in range(len(X)):
        if y[i] == 'Yes':  # If the example is a positive example
            for j in range(len(X.columns)):
                # If the hypothesis is still general or the feature matches the
example, keep it
                if hypothesis[j] == '?' or hypothesis[j] == X.iloc[i, j]:
                    hypothesis[j] = X.iloc[i, j]
                # If the feature doesn't match, make it specific to the example
                else:
                    hypothesis[j] = '?'
    return hypothesis

# Get the most specific hypothesis
hypothesis = find_s_algorithm(X, y)

# Output the hypothesis
print("Hypothesis consistent with the positive examples:", hypothesis)
```

Training data stored as .csv file (a separate file)

Attribute1	Attribute2	Attribute3	Attribute4	Attribute5	Attribute6	Class
Sunny	Warm	Normal	Strong	Warm	Same	Yes
Sunny	Warm	High	Strong	Warm	Same	Yes
Rainy	Cold	High	Strong	Warm	Change	No
Sunny	Warm	High	Strong	Cool	Change	Yes

Output:

Hypothesis consistent with the positive examples: ['Sunny', 'Warm', 'High', 'Strong', '?', '?']

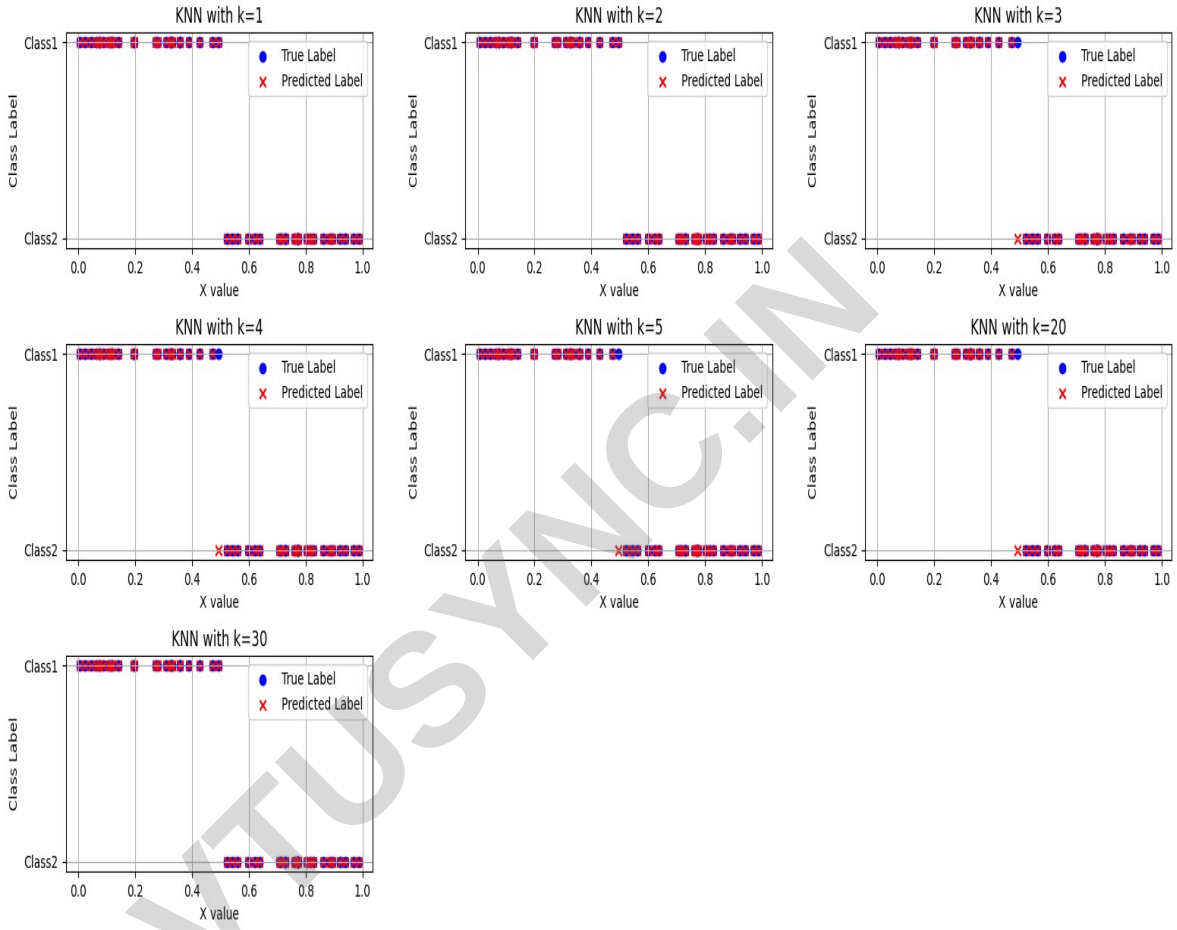
5. Develop a program to implement k-Nearest Neighbour algorithm to classify the randomly generated 100 values of x in the range of [0,1]. Perform the following based on dataset generated.
1.Label the first 50 points [x1,. ,xsoj aS follows: if (xi s 0.5), then x; e Classi, else x, e Classi
2.Classify the remaining points, X51,.....,xi oo using KNN. Perform this for k——1,2,3,4,5,20,30

Python Code:

```
import matplotlib
matplotlib.use('TkAgg')  # Use the TkAgg backend for stable display

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

# Step 1: Generate 100 random values of x in the range [0, 1]
np.random.seed(42)  # For reproducibility
x_values = np.random.rand(100, 1)  # 100 random values in the range [0,1]

# Step 2: Label the first 50 points as Class1 and the rest as Class2
y_labels = np.array(['Class1' if x <= 0.5 else 'Class2' for x in
x_values.flatten()])

# Split into training and testing sets
X_train = x_values[:50]  # First 50 points
y_train = y_labels[:50]  # First 50 labels
X_test = x_values[50:]   # Remaining 50 points
y_test = y_labels[50:]   # Remaining 50 labels

# Step 3: Classify using KNN for different k values
k_values = [1, 2, 3, 4, 5, 20, 30]
plt.figure(figsize=(12, 8))

for i, k in enumerate(k_values, 1):
    # Initialize the k-NN classifier with the current k value
    knn = KNeighborsClassifier(n_neighbors=k)

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

    # Predict the labels for the test set
    y_pred = knn.predict(X_test)

    # Plot the decision boundary and the points
    plt.subplot(3, 3, i)
    plt.scatter(X_test, y_test, color='blue', label='True Label')
    plt.scatter(X_test, y_pred, color='red', marker='x', label='Predicted Label')

    plt.title(f"KNN with k={k}")
    plt.xlabel("X value")
    plt.ylabel("Class Label")
    plt.legend(loc='best')
    plt.grid(True)

# Display the plots
plt.tight_layout()
plt.show()

# Step 4: Evaluate classification accuracy for each k value
for k in k_values:
```

```
knn = KNeighborsClassifier(n_neighbors=k)
knn.fit(X_train, y_train)
accuracy = knn.score(X_test, y_test)
print(f"Accuracy for k={k}: {accuracy:.2f}")
```

Output:



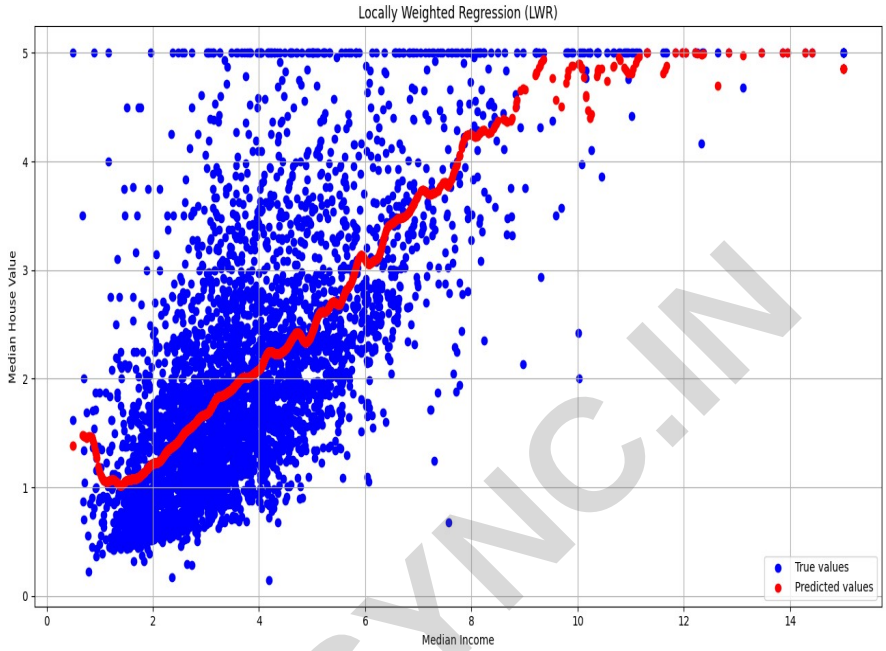
Accuracy for k=1: 1.00
Accuracy for k=2: 1.00
Accuracy for k=3: 0.98
Accuracy for k=4: 0.98
Accuracy for k=5: 0.98
Accuracy for k=20: 0.98
Accuracy for k=30: 1.00

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.

Python Code:

```
import matplotlib
matplotlib.use('TkAgg')  # Use TkAgg backend for interactive plotting
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import fetch_california_housing
from sklearn.model_selection import train_test_split

# Load the dataset and select a feature
data = fetch_california_housing(as_frame=True)
df = data.frame
X = df['MedInc'].values.reshape(-1, 1)
y = df['MedHouseVal'].values

# Split into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Locally Weighted Regression (LWR)
def locally_weighted_regression(X_train, y_train, X_test, tau=0.1):
    predictions = []
    for x in X_test:
        weights = np.exp(-np.sum((X_train - x) ** 2, axis=1) / (2 * tau ** 2))
        X_train_b = np.c_[np.ones((X_train.shape[0], 1)), X_train]

        # Solve the weighted least squares problem using np.linalg.lstsq for
efficiency
        theta, _, _, _ = np.linalg.lstsq(X_train_b * weights[:, np.newaxis],
y_train * weights, rcond=None)

        # Make prediction for the current test point
        X_test_b = np.c_[1, x]  # Add bias term
        predictions.append(X_test_b @ theta)
    return np.array(predictions)

# Predict values for the test set
y_pred = locally_weighted_regression(X_train, y_train, X_test, tau=0.1)

# Plot results
plt.scatter(X_test, y_test, color='blue', label='True values')
plt.scatter(X_test, y_pred, color='red', label='Predicted values')
plt.xlabel('Median Income')
plt.ylabel('Median House Value')
plt.title('Locally Weighted Regression (LWR)')
plt.legend()
plt.grid(True)

# Show plot (interactive window)
plt.show()

# Evaluate performance
mse = np.mean((y_pred - y_test) ** 2)
print(f"Mean Squared Error: {mse:.4f}")
```

Output:



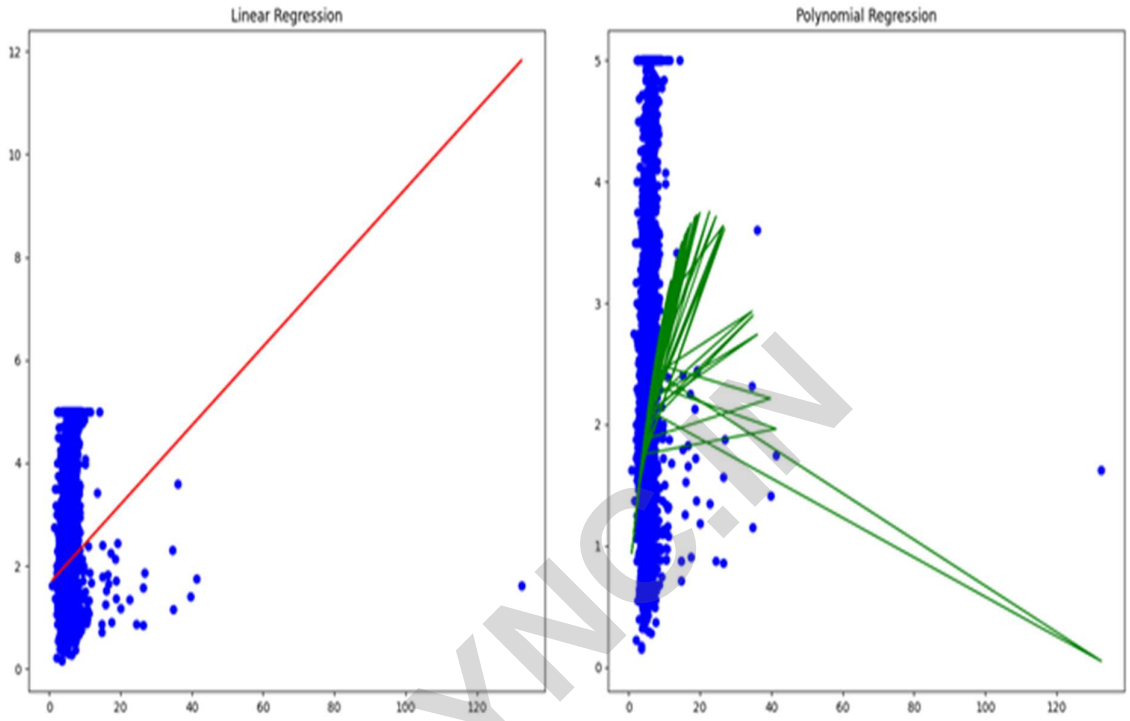
Mean Squared Error: 1.9598

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.

Python Code:

```
import numpy as np
import pandas as pd
import matplotlib
matplotlib.use('TkAgg')  # Use TkAgg backend
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import train_test_split
from sklearn.datasets import fetch_california_housing

# Load California Housing dataset for Linear Regression
data = fetch_california_housing(as_frame=True)
X = data.data[['AveRooms']]
y = data.target

# Split dataset
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Linear Regression
linear_reg = LinearRegression()
linear_reg.fit(X_train, y_train)
y_pred = linear_reg.predict(X_test)

# Polynomial Regression
poly = PolynomialFeatures(degree=3)
X_poly = poly.fit_transform(X_train)
poly_reg = LinearRegression()
poly_reg.fit(X_poly, y_train)
y_pred_poly = poly_reg.predict(poly.fit_transform(X_test))

# Plotting results
plt.subplot(1, 2, 1)
plt.scatter(X_test, y_test, color='blue')
plt.plot(X_test, y_pred, color='red')
plt.title('Linear Regression')

plt.subplot(1, 2, 2)
plt.scatter(X_test, y_test, color='blue')
plt.plot(X_test, y_pred_poly, color='green')
plt.title('Polynomial Regression')

plt.tight_layout()
plt.show()

# Output MSE
print(f"Linear Regression - MSE: {mean_squared_error(y_test, y_pred):.4f}")
print(f"Polynomial Regression - MSE: {mean_squared_error(y_test,
y_pred_poly):.4f}")
```

Output:



Linear Regression - MSE: 1.2923
Polynomial Regression - MSE: 1.2117

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.

Python Code:

```
# Import necessary libraries
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
import numpy as np

# Load the Breast Cancer dataset
data = load_breast_cancer()
X = data.data
y = data.target

# 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)

# Create a DecisionTreeClassifier instance and train it
clf = DecisionTreeClassifier(random_state=42)
clf.fit(X_train, y_train)

# Predict the test set results
y_pred = clf.predict(X_test)

# Evaluate the classifier performance
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy on Test Set: {accuracy:.4f}')

# Classify a new sample (randomly selected from the test set for demonstration)
new_sample = X_test[0].reshape(1, -1)  # Take the first sample from the test set
predicted_class = clf.predict(new_sample)

# Output the predicted class (0: malignant, 1: benign)
print(f'Predicted Class for New Sample: {"Benign" if predicted_class == 1 else
"Malignant"}')
```


Output:
Accuracy on Test Set: 0.9415
Predicted Class for New Sample: Benign

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.

Python Code:

```
import numpy as np
from scipy.io import loadmat
from sklearn.naive_bayes import GaussianNB
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

# Load the olivettifaces.mat file (ensure it's in the same directory or update the path)
data = loadmat('olivettifaces.mat')

# Inspect the keys in the dataset
print("Keys in the dataset:", data.keys())

# Use 'faces' as the feature matrix
X = data['faces']  # Features (faces), this is the matrix of images

# Assuming labels are the index of faces (0-40 for 40 individuals, 10 images per individual)
y = np.repeat(np.arange(40), 10)  # 40 classes (individuals), 10 images per class

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

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

# Make predictions
y_pred = model.predict(X_test)

# Calculate accuracy
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.4f}")
```


Output:

Keys in the dataset: dict_keys(['__header__', '__version__', '__globals__', 'faces', 'p', 'u', 'v'])
Accuracy: 0.7417

10. Develop a program to implement k-means clustering using Wisconsin Breast Cancer data set and visualize the clustering result.

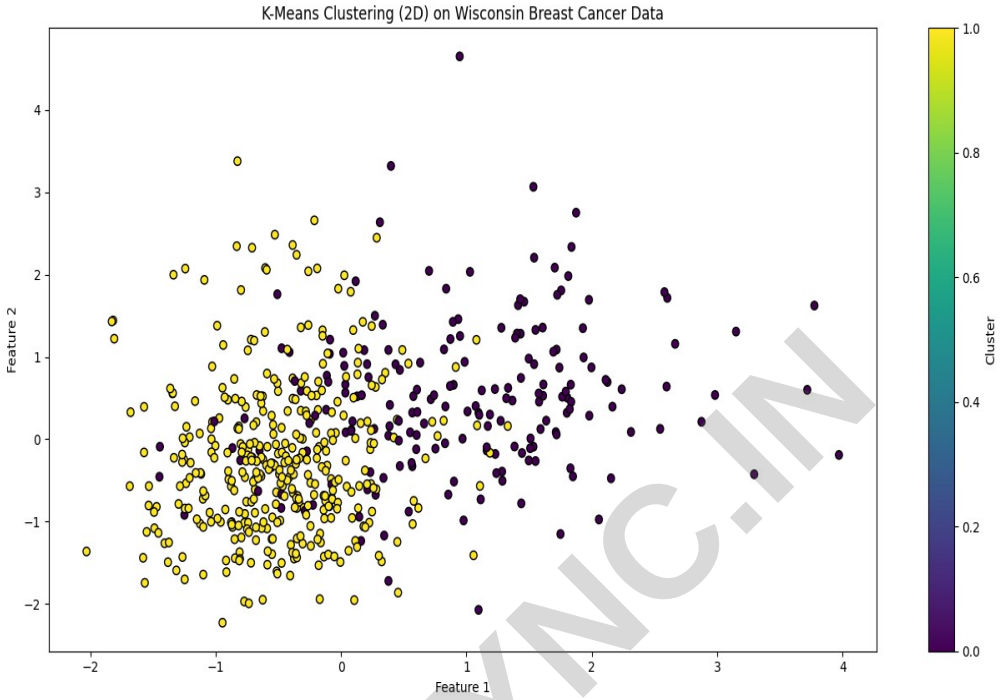
Python Code:

```
import matplotlib
matplotlib.use('TkAgg')  # Use the TkAgg backend for interactive GUI rendering

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.datasets import load_breast_cancer
from sklearn.preprocessing import StandardScaler

# Load the breast cancer dataset
data = load_breast_cancer()
X = data.data
y = data.target

# Standardize the data
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Apply KMeans clustering
kmeans = KMeans(n_clusters=2, random_state=42)
y_kmeans = kmeans.fit_predict(X_scaled)

# Visualize the clustering result
plt.figure(figsize=(10, 6))
plt.scatter(X_scaled[:, 0], X_scaled[:, 1], c=y_kmeans, cmap='viridis',
edgecolors='k')
plt.title('K-Means Clustering (2D) on Wisconsin Breast Cancer Data')
plt.xlabel('Feature 1')
plt.ylabel('Feature 2')
plt.colorbar(label='Cluster')

# Show the plot interactively using TkAgg
plt.show()

# Optionally, print cluster centers
print("Cluster centers:\n", kmeans.cluster_centers_)
```


Output:

Cluster centers:

[[0.98649786 0.49202131 1.01866689 0.97479724 0.5871176 1.014073
1.14492245 1.17028266 0.60339021 0.22927434 0.86311672 0.04416341
0.86446528 0.8137762 0.01228944 0.69281919 0.63976499 0.77166695
0.13798752 0.40384985 1.05221312 0.51705679 1.07769473 1.01391704
0.59804381 0.95285513 1.05144274 1.15328841 0.5994129 0.61362004]
[-0.48677585 -0.24278217 -0.50264928 -0.48100231 -0.28970632 -0.50038248
-0.56494861 -0.57746231 -0.29773585 -0.11313275 -0.42589487 -0.02179192
-0.4265603 -0.40154836 -0.00606408 -0.34186354 -0.31568456 -0.38077004
-0.06808833 -0.19927499 -0.51920227 -0.25513563 -0.53177588 -0.50030552
-0.29509774 -0.47017523 -0.51882214 -0.56907669 -0.29577329 -0.30278364]]


K-Means Clustering (2D) on Wisconsin Breast Cancer Data
Feature 1
Feature 2
Cluster
4
3
2
1
0
−1
−2
−2
−1
0
1
2
3
4
1.0
0.8
0.6
0.4
0.2
0.0