



SDM INSTITUTE OF TECHNOLOGY

**DEPARTMENT OF COMPUTER SCIENCE &
ENGINEERING**

VI SEMESTER

LAB MANUAL

COURSE: MACHINE LEARNING LAB

COURSE CODE: BCSL606

2024-2025

Dr.Thyagaraju G S

Pradeep Rao K B

Contents

Sl. No	Topic	Page No
1	Syllabus	3
2	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.	7
3	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.	19
4	Develop a program to implement Principal Component Analysis (PCA) for reducing the dimensionality of the Iris dataset from 4 features to 2.	22
5	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.	24
6	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 $\{x_1, \dots, x_{50}\}$ as follows: if $(x_i \leq 0.5)$, then $x_i \in \text{Class1}$, else $x_i \in \text{Class2}$ b. Classify the remaining points, x_{51}, \dots, x_{100} using KNN. Perform this for $k=1,2,3,4,5,20,30$	26
7	Implement the non-parametric Locally Weighted Regression algorithm in order to fit data points. Select appropriate data set for your experiment and draw graphs	28
8	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.	30

9	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.	34
10	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.	36
11	Develop a program to implement k-means clustering using Wisconsin Breast Cancer data set and visualize the clustering result.	40
12	Viva Questions	42

Syllabus

Template for Practical Course and if AEC is a practical Course Annexure-V

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:			
<ul style="list-style-type: none"> To become familiar with data and visualize univariate, bivariate, and multivariate data using statistical techniques and dimensionality reduction. To understand various machine learning algorithms such as similarity-based learning, regression, decision trees, and clustering. 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. <ol style="list-style-type: none"> Label the first 50 points $\{x_1, \dots, x_{50}\}$ as follows: if $(x_i \leq 0.5)$, then $x_i \in \text{Class}_1$, else $x_i \in \text{Class}_2$ Classify the remaining points, x_{51}, \dots, x_{100} 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		

@#11012025

9	<p>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.</p> <p>Book 2: Chapter – 4</p>
10	<p>Develop a program to implement k-means clustering using Wisconsin Breast Cancer data set and visualize the clustering result.</p> <p>Book 2: Chapter – 4</p>
<p>Course outcomes (Course Skill Set): At the end of the course the student will be able to:</p> <ul style="list-style-type: none"> • Illustrate the principles of multivariate data and apply dimensionality reduction techniques. • Demonstrate similarity-based learning methods and perform regression analysis. • Develop decision trees for classification and regression problems, and Bayesian models for probabilistic learning. • Implement the clustering algorithms to share computing resources. 	
<p>Assessment Details (both CIE and SEE)</p> <p>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</p>	
<p>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.</p> <ul style="list-style-type: none"> • 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). <p>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.</p>	

Semester End Evaluation (SEE):

- SEE marks for the practical course are 50 Marks.
- SEE shall be conducted jointly by the two examiners of the same institute, examiners are appointed by the Head of the Institute.
- 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.
- All laboratory experiments are to be included for practical examination.
- (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.
- Students can pick one question (experiment) from the questions lot prepared by the examiners jointly.
- 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):

- https://www.drssidhar.com/?page_id=1053
- <https://www.universitiespress.com/resources?id=9789393330697>
- https://onlinecourses.nptel.ac.in/noc23_cs18/preview

EXPERIMENT 1

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
import seaborn as sns
from sklearn.datasets import fetch_california_housing

# Load the California Housing dataset
california_data = fetch_california_housing(as_frame=True)
data = california_data.frame

# Display basic information about the dataset
print("Dataset Overview:")
print(data.info())
print("\nFirst few rows:")
print(data.head())

# Create histograms for all numerical features
def plot_histograms(df):
    for column in df.columns:
        plt.figure(figsize=(10, 6))
        plt.hist(df[column], bins=30, color='skyblue', edgecolor='black')
        plt.title(f'Histogram of {column}', fontsize=16)
        plt.xlabel(column, fontsize=14)
        plt.ylabel('Frequency', fontsize=14)
        plt.grid(axis='y', alpha=0.75)
        plt.show()
```

```

# Create box plots for all numerical features
def plot_boxplots(df):
    for column in df.columns:
        plt.figure(figsize=(10, 6))
        sns.boxplot(x=df[column], color='lightgreen')
        plt.title(f'Box Plot for {column}', fontsize=14)
        plt.xlabel(column, fontsize=12)
        plt.show()

# Detect and display outliers using IQR
# Outliers are defined as values below Q1 - 1.5*IQR or above Q3 + 1.5*IQR
def detect_outliers(df):
    outlier_summary = { }
    for column in df.columns:
        Q1 = df[column].quantile(0.25)
        Q3 = df[column].quantile(0.75)
        IQR = Q3 - Q1
        lower_bound = Q1 - 1.5 * IQR
        upper_bound = Q3 + 1.5 * IQR
        outliers = df[(df[column] < lower_bound) | (df[column] > upper_bound)]
        outlier_summary[column] = outliers.shape[0]
        print(f"{column}: {outliers.shape[0]} outliers detected.")
    return outlier_summary

# Plot histograms
plot_histograms(data)
# Plot box plots
plot_boxplots(data)
# Detect and summarize outliers
print("\nOutlier Detection Summary:")
outlier_summary = detect_outliers(data)

```

Output

Dataset Overview:

```
<class 'pandas.core.frame.DataFrame'>
```

```
RangeIndex: 20640 entries, 0 to 20639
```

```
Data columns (total 9 columns):
```

#	Column	Non-Null Count	Dtype
0	MedInc	20640 non-null	float64
1	HouseAge	20640 non-null	float64
2	AveRooms	20640 non-null	float64
3	AveBedrms	20640 non-null	float64
4	Population	20640 non-null	float64
5	AveOccup	20640 non-null	float64
6	Latitude	20640 non-null	float64
7	Longitude	20640 non-null	float64
8	MedHouseVal	20640 non-null	float64

```
dtypes: float64(9)
```

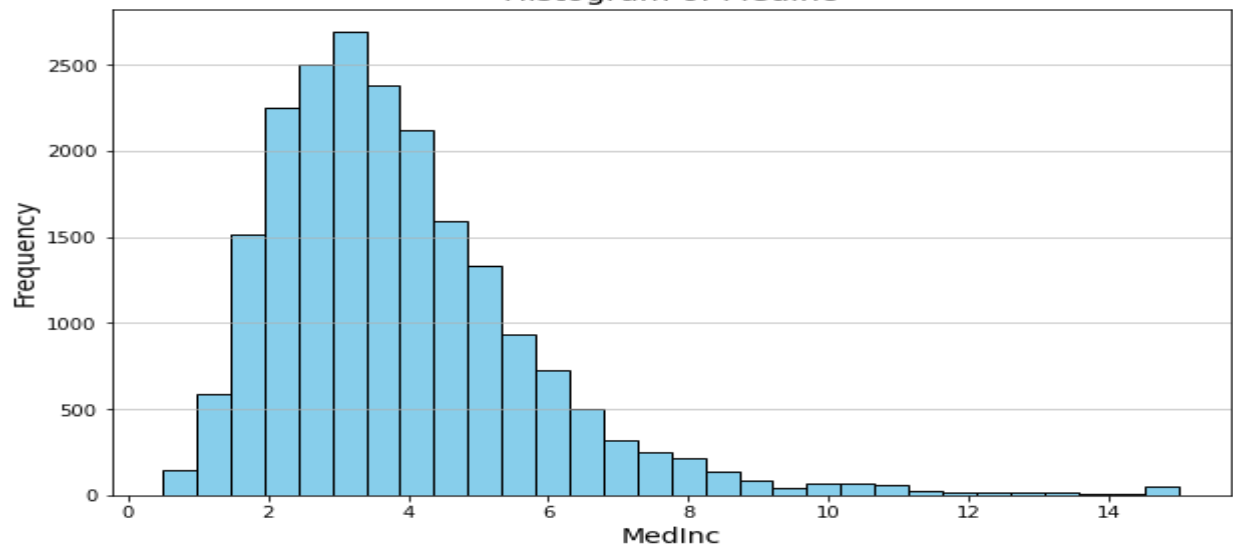
```
memory usage: 1.4 MB
```

```
None
```

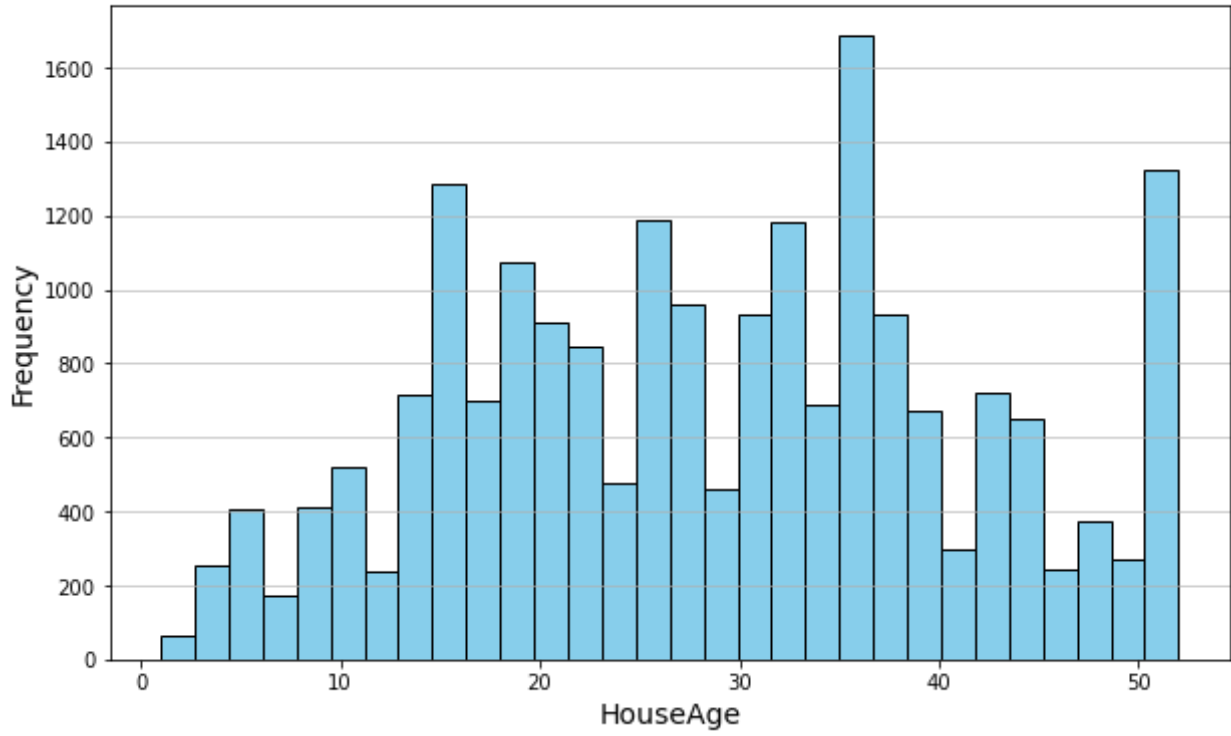
First few rows:

	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	Longitude	MedHouseVal
0	8.3252	41.0	6.984127	1.023810	322.0	2.555556	37.88	-122.23	4.526
1	8.3014	21.0	6.238137	0.971880	2401.0	2.109842	37.86	-122.22	3.585
2	7.2574	52.0	8.288136	1.073446	496.0	2.802260	37.85	-122.24	3.521
3	5.6431	52.0	5.817352	1.073059	558.0	2.547945	37.85	-122.25	3.413
4	3.8462	52.0	6.281853	1.081081	565.0	2.181467	37.85		

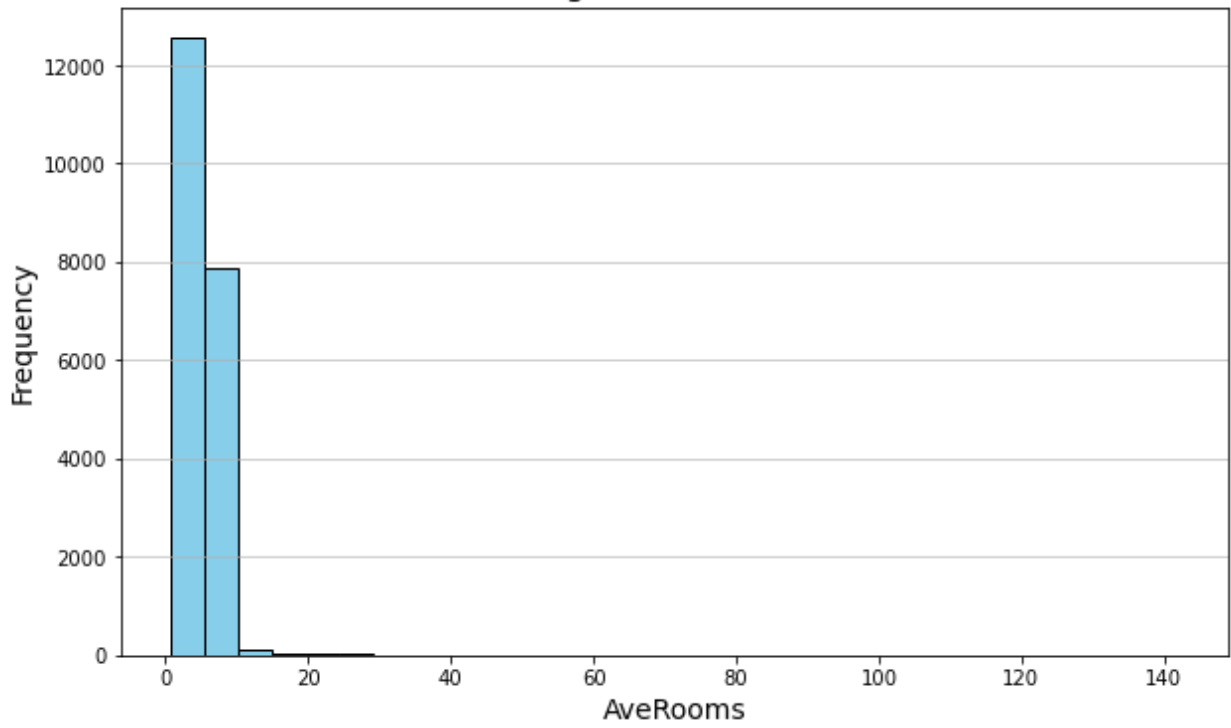
Histogram of MedInc



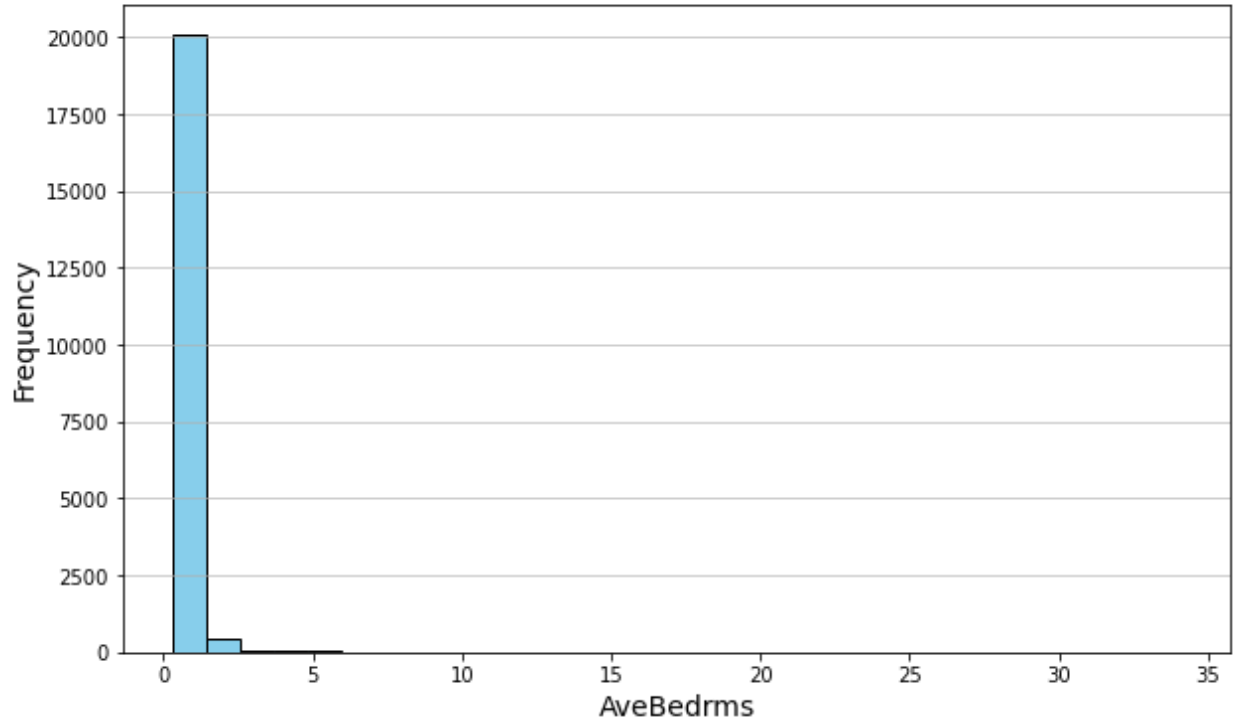
Histogram of HouseAge



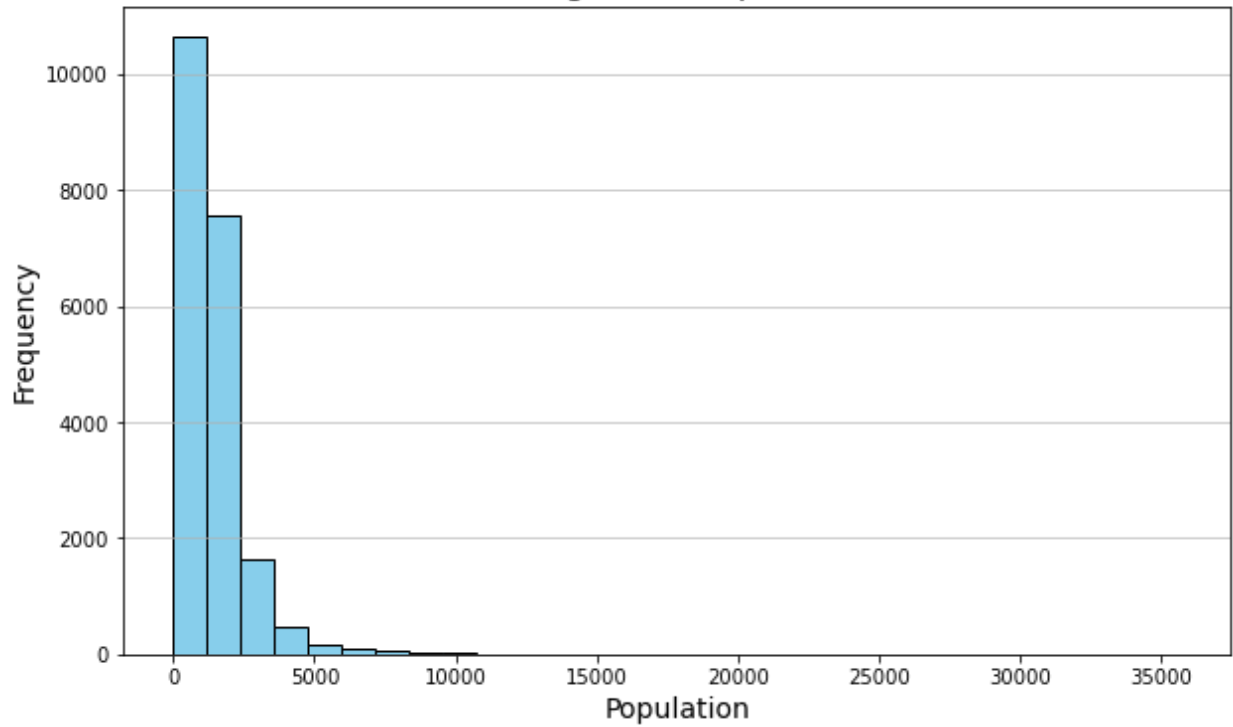
Histogram of AveRooms



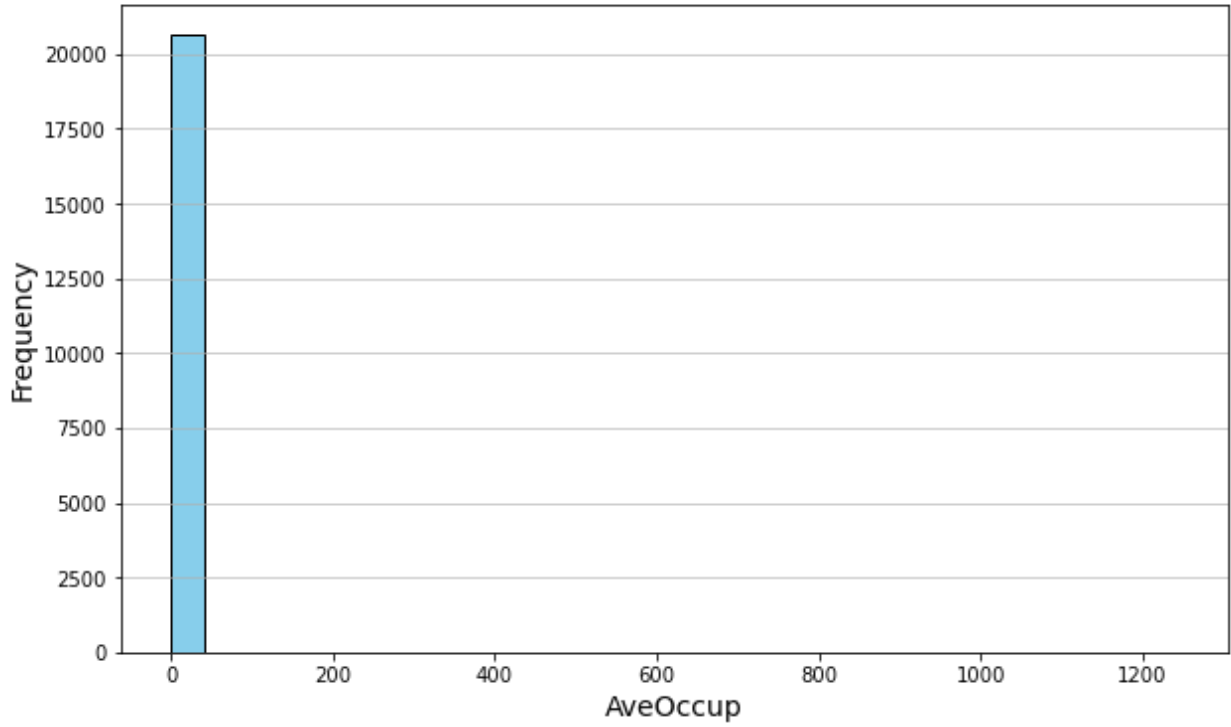
Histogram of AveBedrms



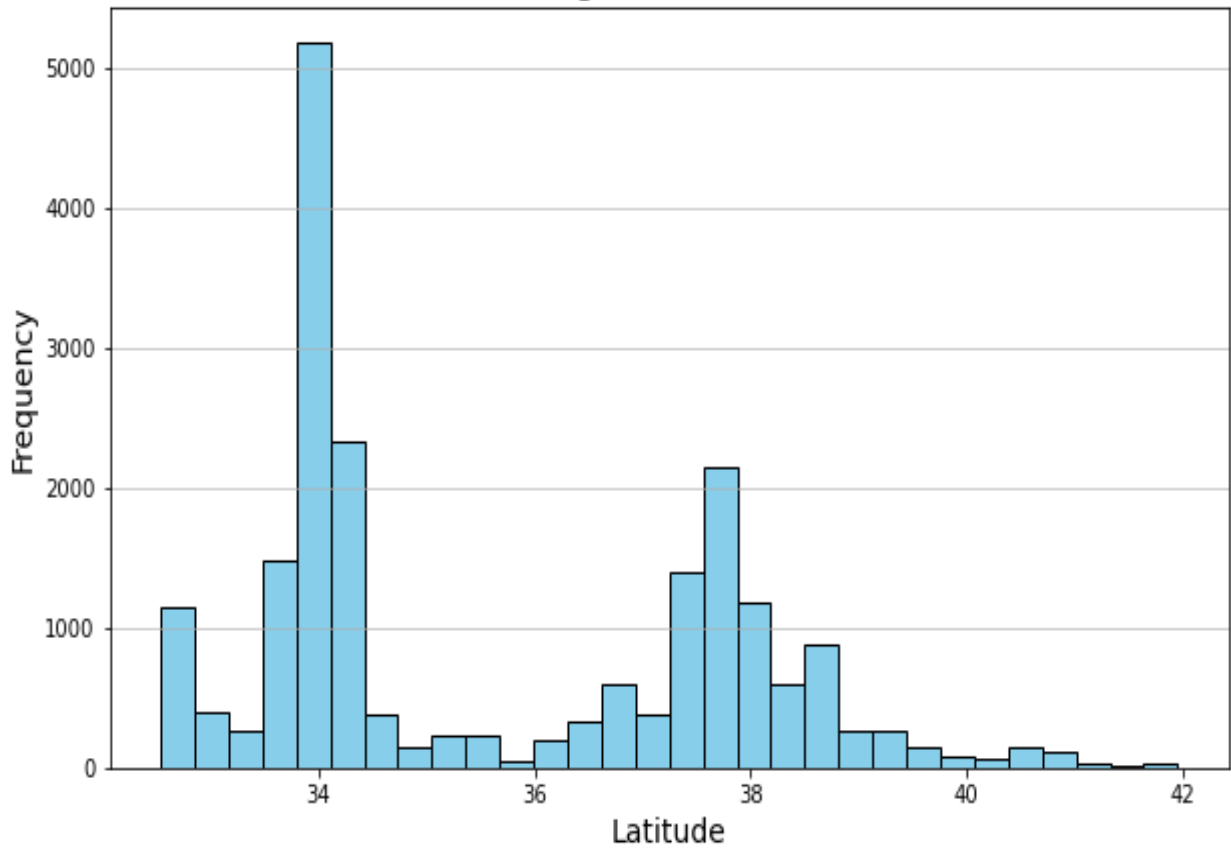
Histogram of Population

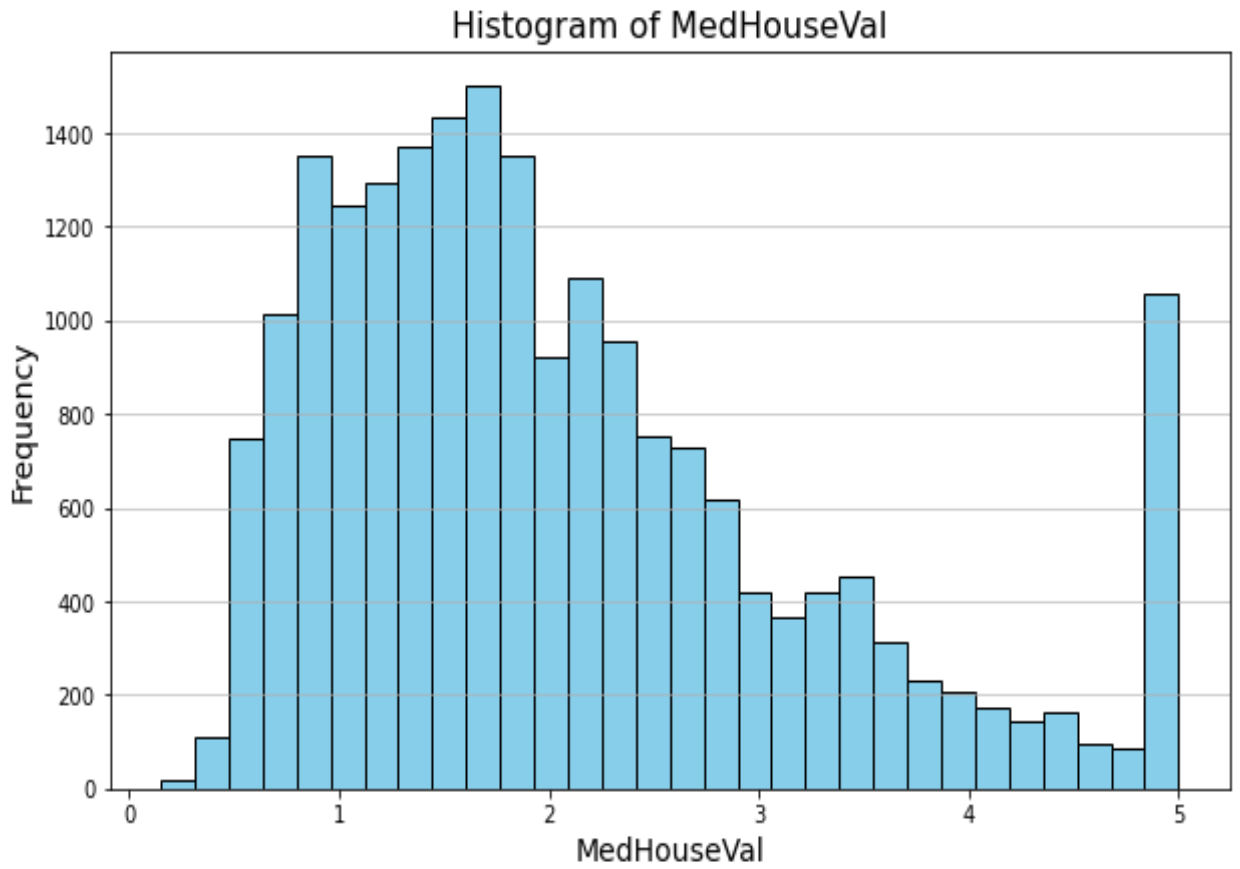
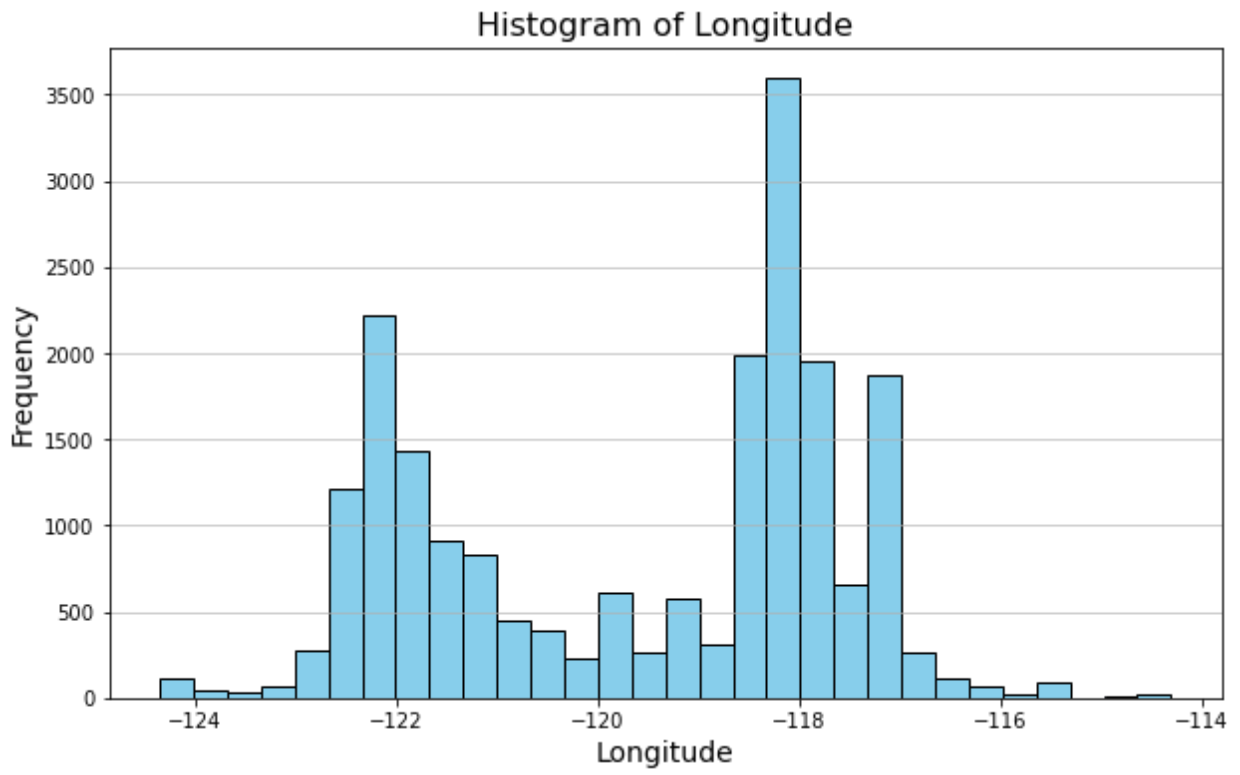


Histogram of AveOccup

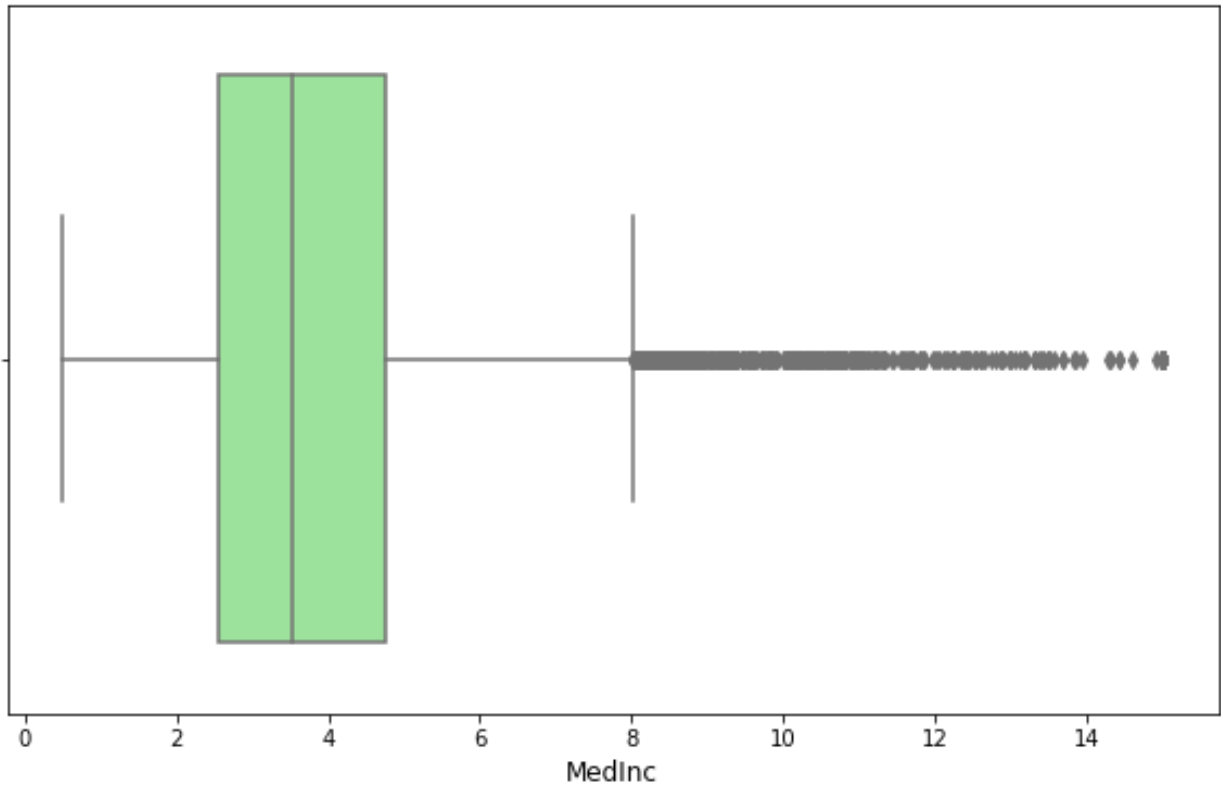


Histogram of Latitude

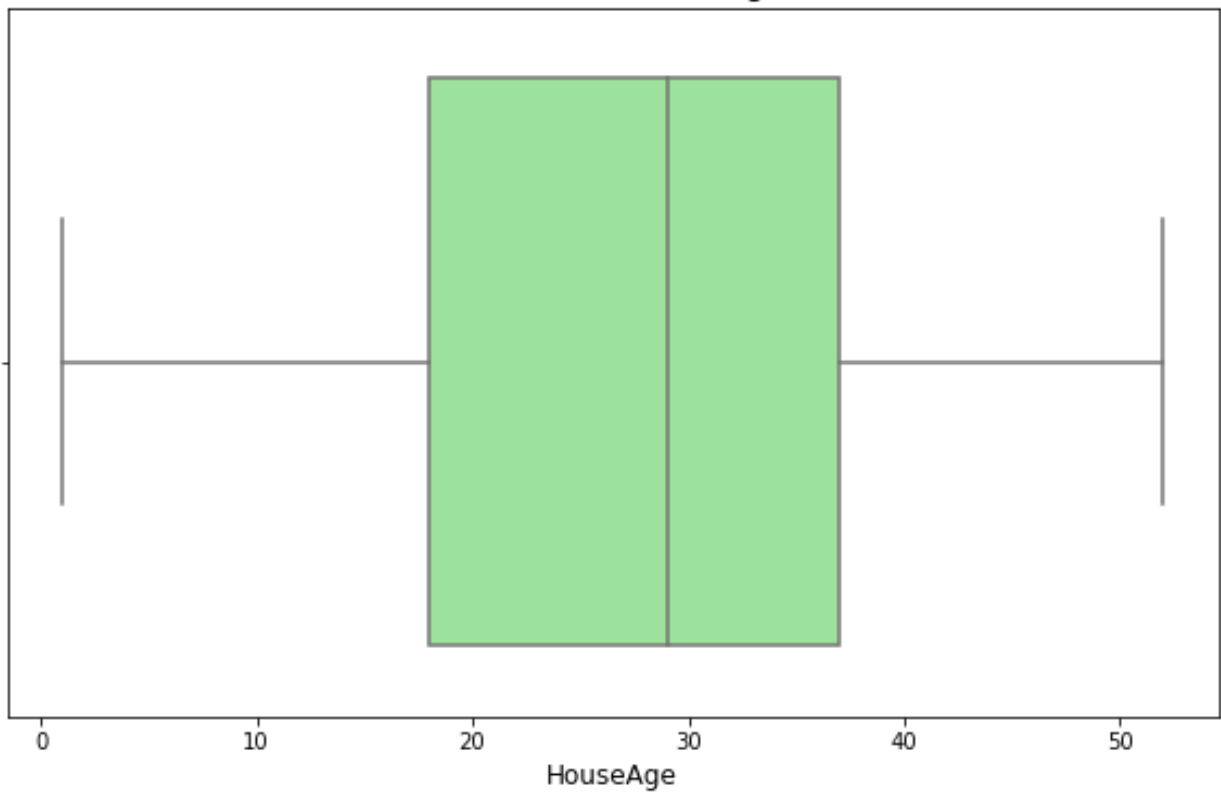




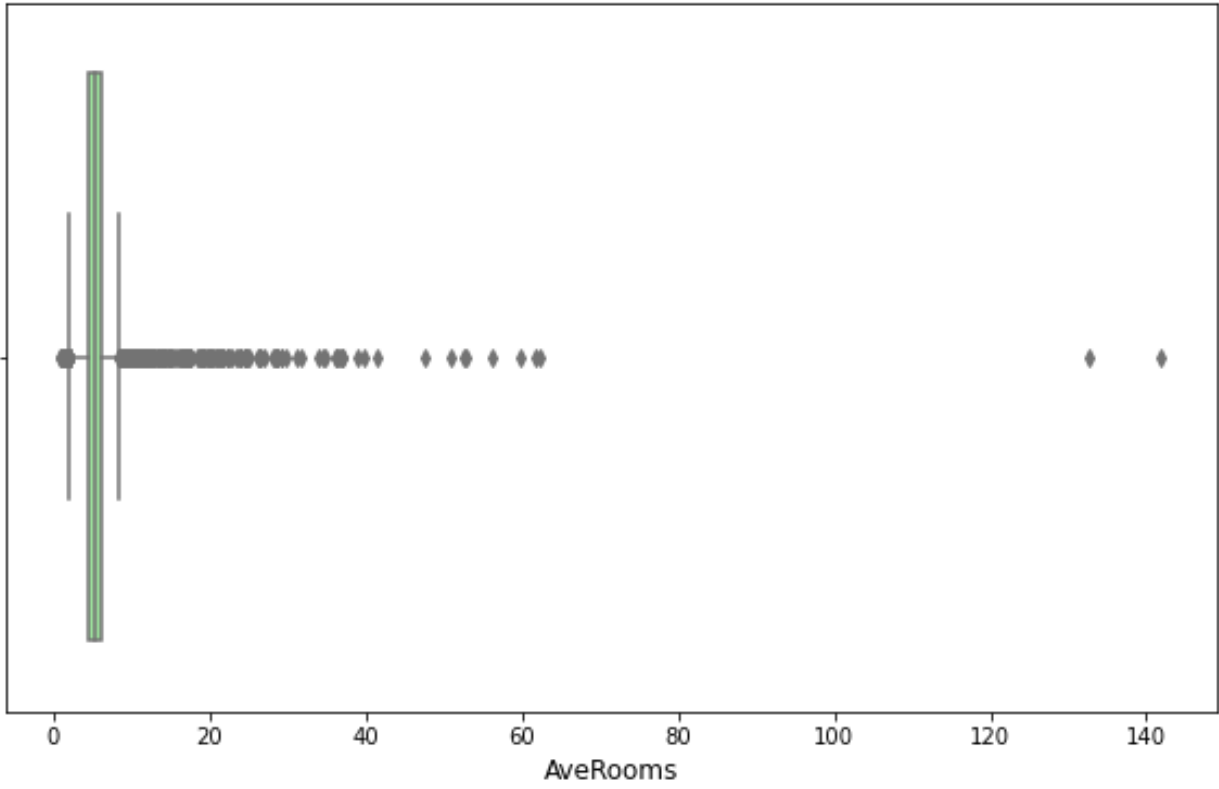
Box Plot for MedInc



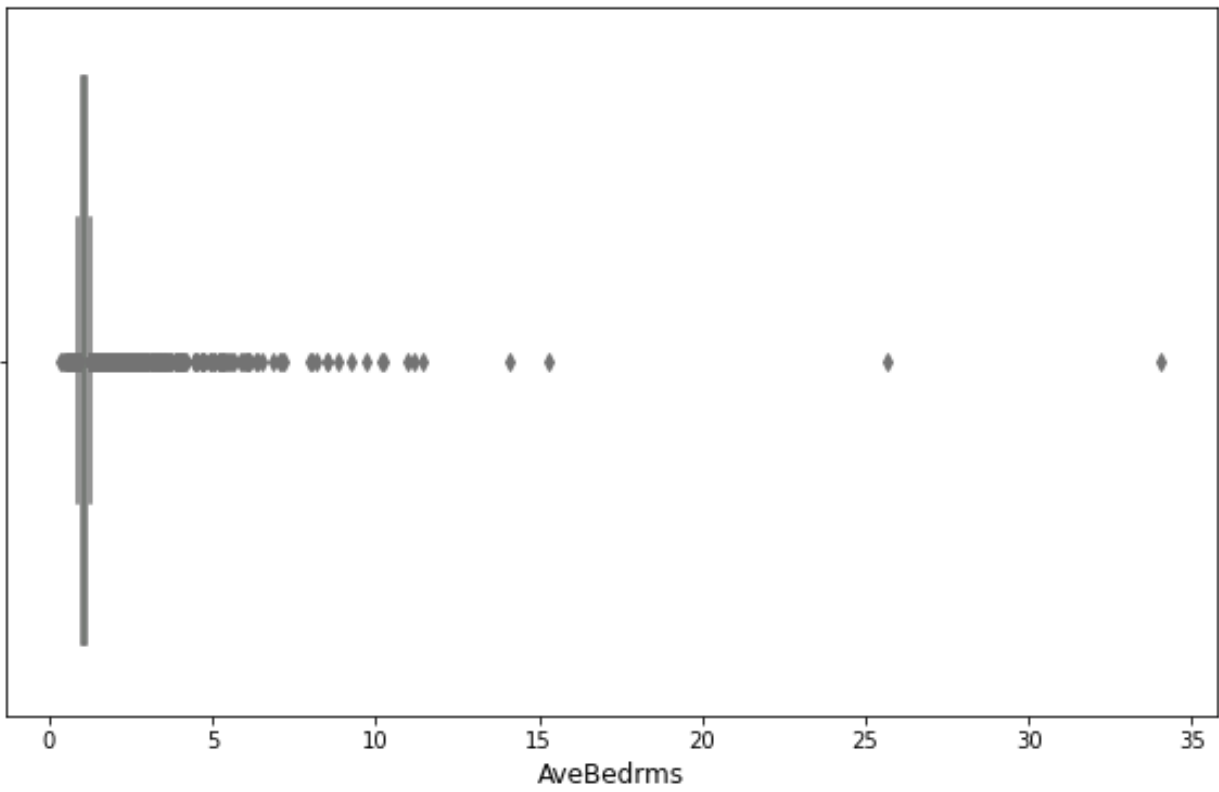
Box Plot for HouseAge



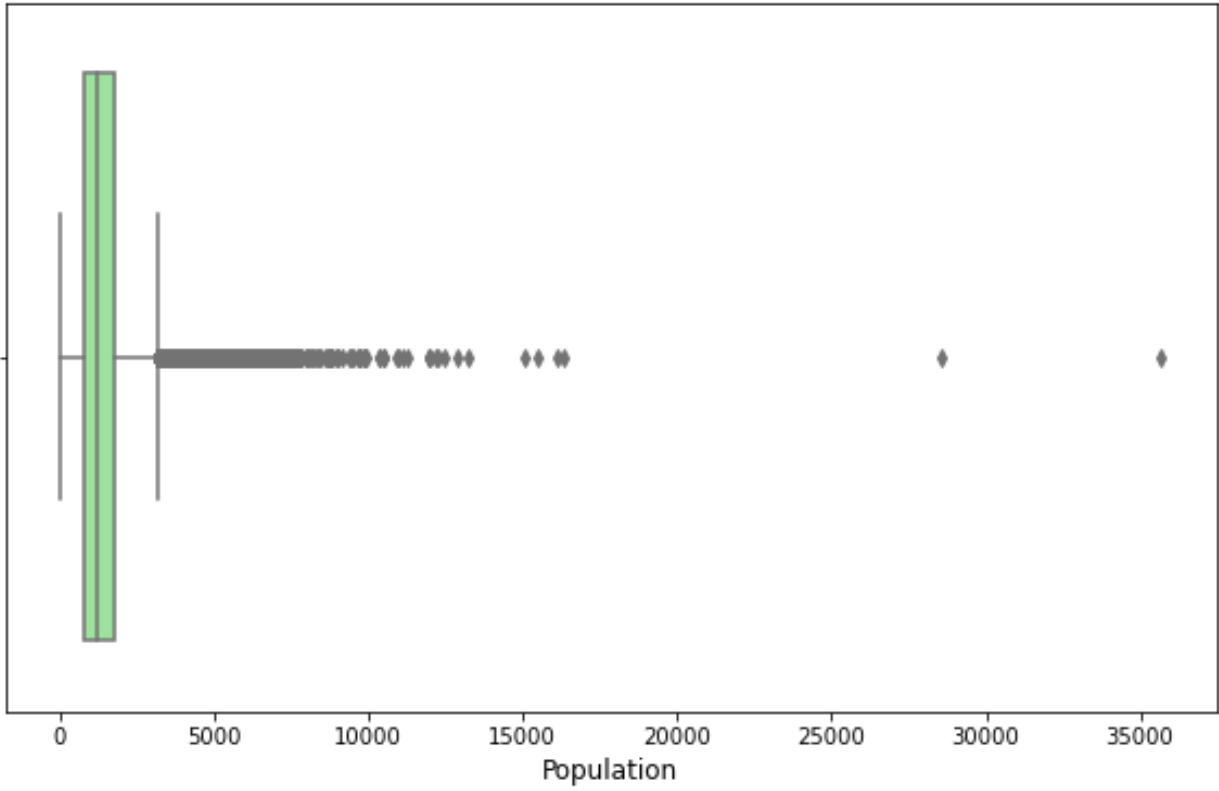
Box Plot for AveRooms



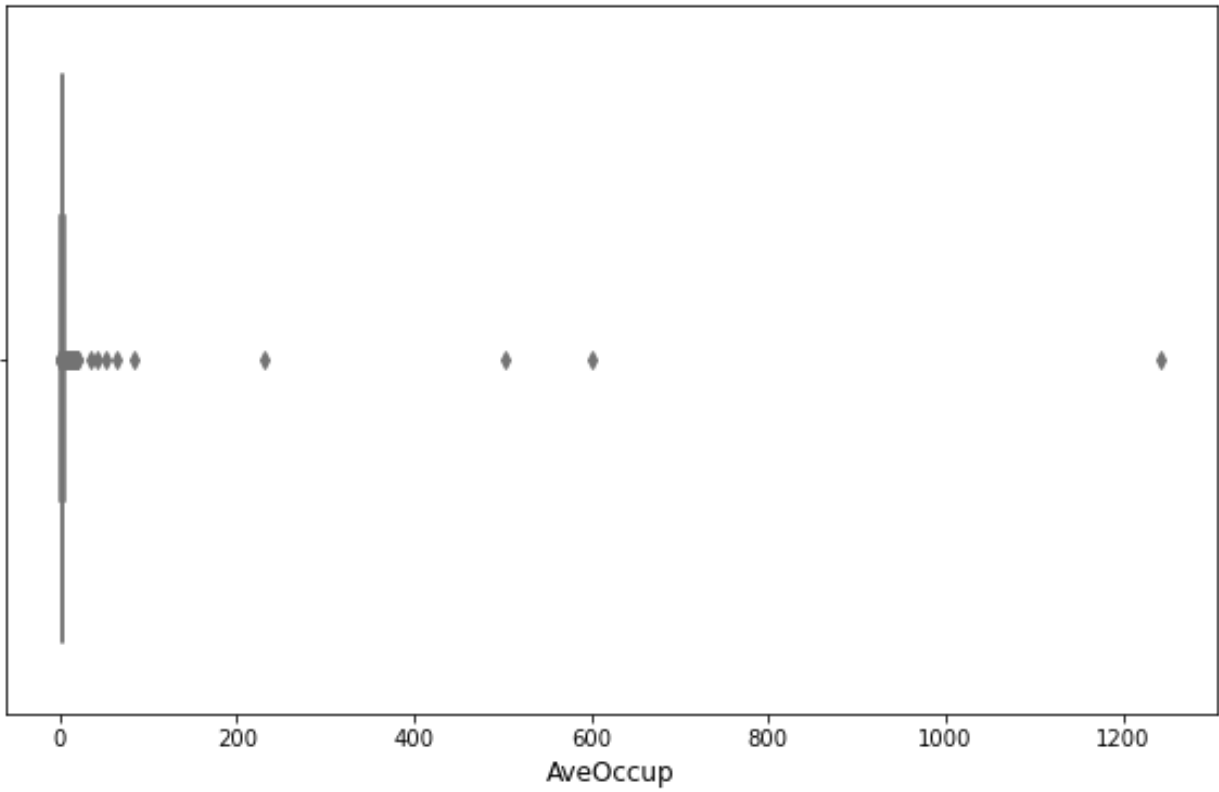
Box Plot for AveBedrms



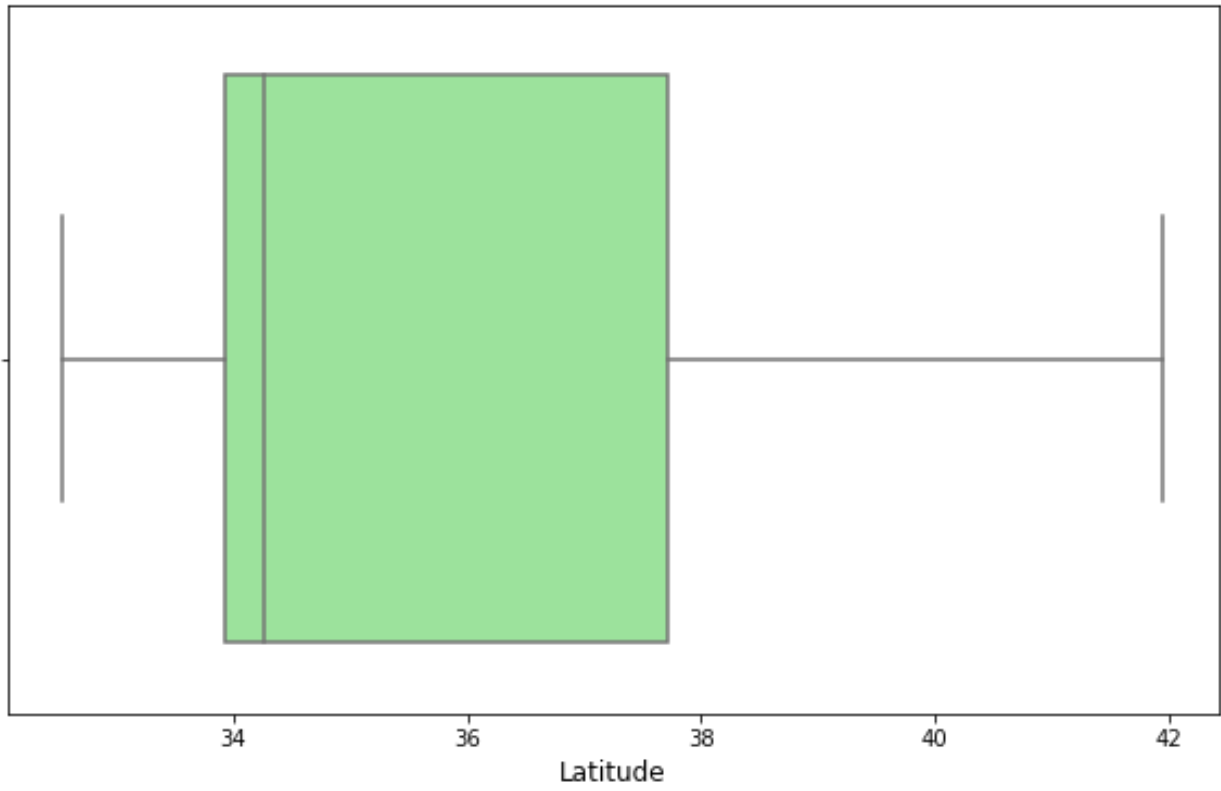
Box Plot for Population



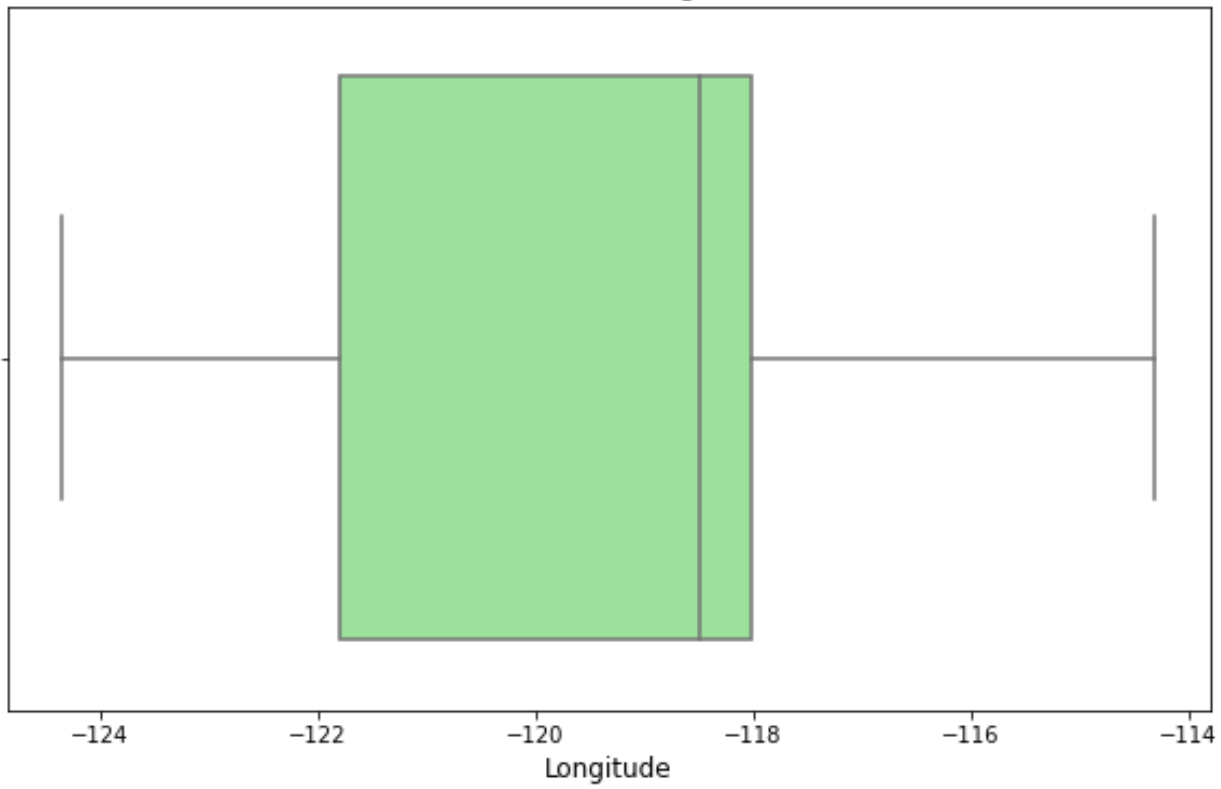
Box Plot for AveOccup

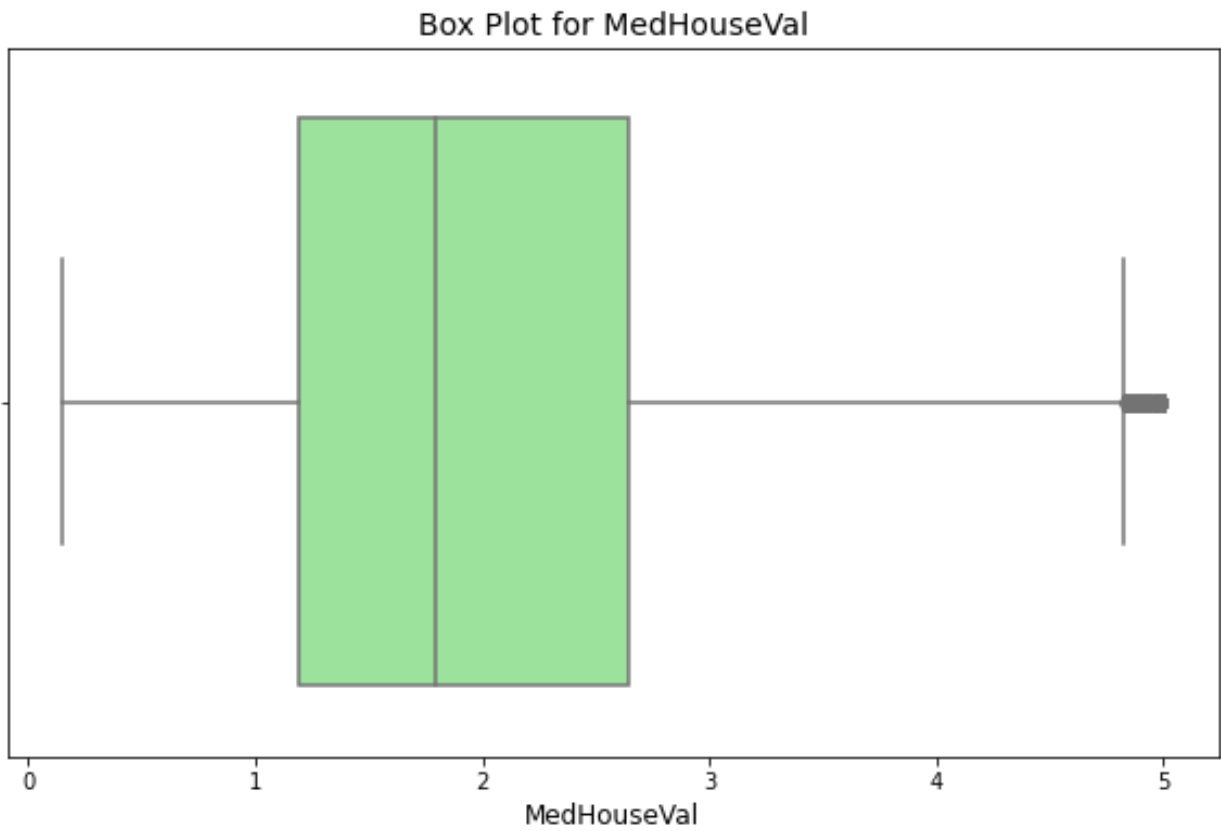


Box Plot for Latitude



Box Plot for Longitude





Outlier Detection Summary:

MedInc: 681 outliers detected.

HouseAge: 0 outliers detected.

AveRooms: 511 outliers detected.

AveBedrms: 1424 outliers detected.

Population: 1196 outliers detected.

AveOccup: 711 outliers detected.

Latitude: 0 outliers detected.

Longitude: 0 outliers detected.

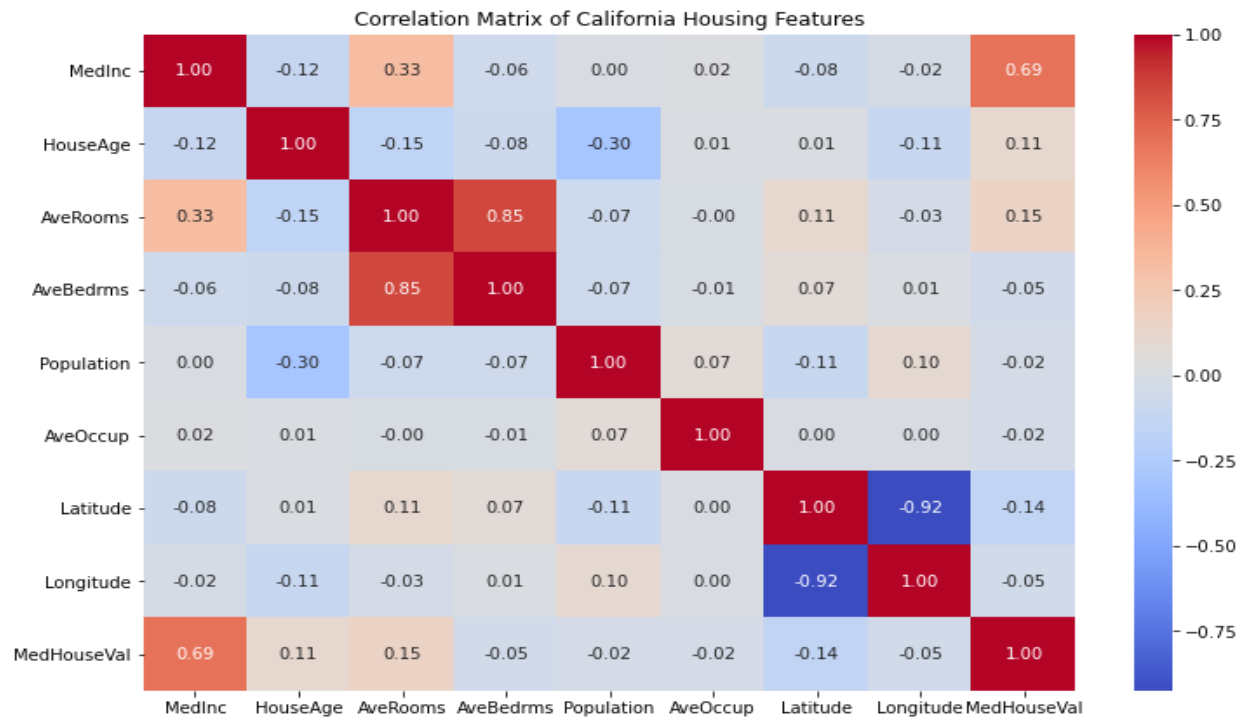
MedHouseVal: 1071 outliers detected.

EXPERIMENT 2

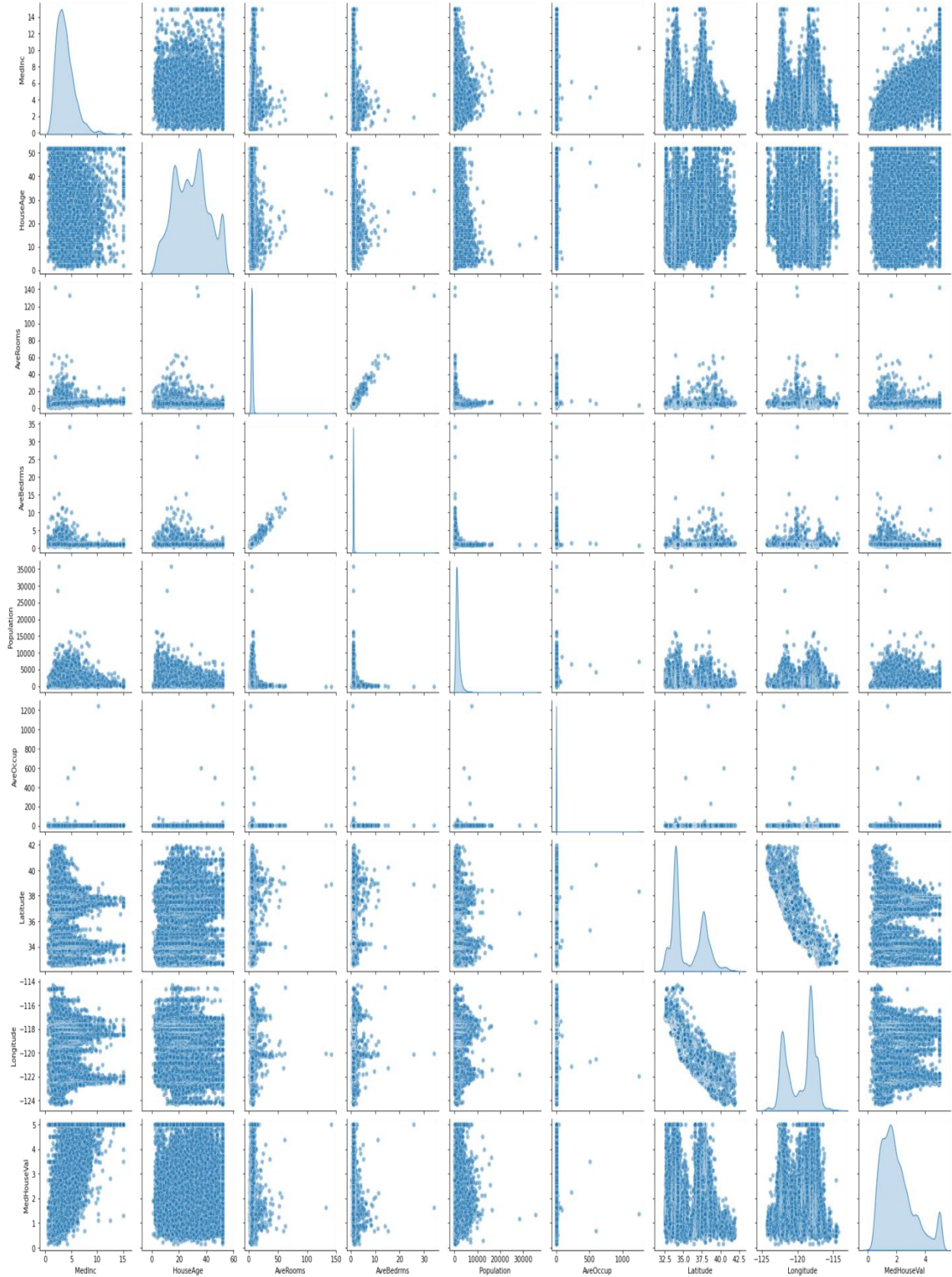
2. Develop a program to Compute the correlation matrix to understand the relationships between pairs of features. Visualize the correlation matrix using a heatmap to know which variables have strong positive/negative correlations. Create a pair plot to visualize pairwise relationships between features.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

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.datasets import fetch_california_housing
# Load the California Housing dataset
data = fetch_california_housing(as_frame=True)
df = data.frame
# Compute the correlation matrix
correlation_matrix = df.corr()
# Visualize the correlation matrix using a heatmap
plt.figure(figsize=(12, 8))
sns.heatmap(correlation_matrix, annot=True, cmap="coolwarm", fmt=".2f")
plt.title("Correlation Matrix of California Housing Features")
plt.show()
# Create a pair plot to visualize pairwise relationships
sns.pairplot(df, diag_kind="kde", plot_kws={"alpha": 0.5})
plt.suptitle("Pair Plot of California Housing Features", y=1.02)
plt.show()
```

Output



Pair Plot of California Housing Features



EXPERIMENT 3

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.preprocessing import StandardScaler
import matplotlib.pyplot as plt

# Load the Iris dataset
iris = load_iris()
data = iris.data
target = iris.target
feature_names = iris.feature_names

# Standardize the data
scaler = StandardScaler()
data_standardized = scaler.fit_transform(data)

# Compute the covariance matrix
covariance_matrix = np.cov(data_standardized.T)

# Compute eigenvalues and eigenvectors
eigenvalues, eigenvectors = np.linalg.eig(covariance_matrix)

# Sort eigenvalues and eigenvectors in descending order of eigenvalues
sorted_indices = np.argsort(eigenvalues)[::-1]
eigenvalues = eigenvalues[sorted_indices]
eigenvectors = eigenvectors[:, sorted_indices]

# Select the top 2 eigenvectors
principal_components = eigenvectors[:, :2]

# Transform the data to the new 2D space
data_reduced = np.dot(data_standardized, principal_components)

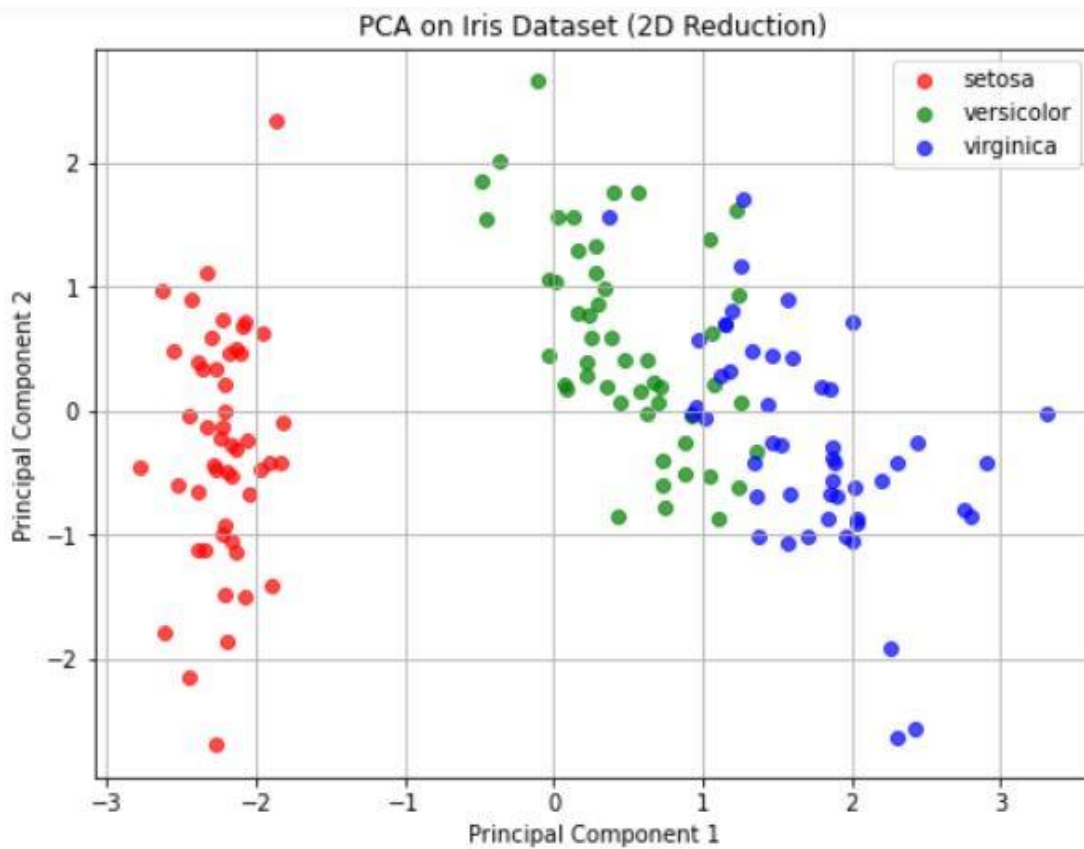
# Create a DataFrame for visualization
```

```

reduced_df = pd.DataFrame(data_reduced, columns=['PC1', 'PC2'])
reduced_df['Target'] = target
# Plot the reduced data
plt.figure(figsize=(8, 6))
colors = ['red', 'green', 'blue']
for i, color in enumerate(colors):
    plt.scatter(reduced_df[reduced_df['Target'] == i]['PC1'],
                reduced_df[reduced_df['Target'] == i]['PC2'],
                color=color, label=iris.target_names[i], alpha=0.7)
plt.title('PCA on Iris Dataset (2D Reduction)')
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.legend()
plt.grid()
plt.show()

```

Output



EXPERIMENT 4

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 csv
with open('enjoysport.csv', 'r') as file:
    data = [row for row in csv.reader(file)] #list comprehension
    print("The total number of training instances are:",len(data)-1,"\n",data[1:])

num_attribute = len(data[0])-1
# Initial hypothesis
hypothesis = ['0']*num_attribute
for i in range(0, len(data)):
    if data[i][num_attribute] == 'yes':
        for j in range(0, num_attribute):
            if hypothesis[j] == '0' or hypothesis[j] == data[i][j]:
                hypothesis[j] = data[i][j]
            else:
                hypothesis[j] = '?'
    print("\nThe hypothesis for the training instance { } is : \n".format(i),hypothesis)

print("\nThe Maximally specific hypothesis for the training instances is ", hypothesis)
```

Output

The total number of training instances are: 4

```
[[['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']]]
```

The hypothesis for the training instance 0 is :

```
['0', '0', '0', '0', '0', '0']
```

The hypothesis for the training instance 1 is :

['sunny', 'warm', 'normal', 'strong', 'warm', 'same']

The hypothesis for the training instance 2 is :

['sunny', 'warm', '?', 'strong', 'warm', 'same']

The hypothesis for the training instance 3 is :

['sunny', 'warm', '?', 'strong', 'warm', 'same']

The hypothesis for the training instance 4 is :

['sunny', 'warm', '?', 'strong', '?', '?']

The Maximally specific hypothesis for the training instances is ['sunny', 'warm', '?', 'strong', '?', '?']

EXPERIMENT 5

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 $\{x_1, \dots, x_{50}\}$ as follows: if $(x_i \leq 0.5)$, then $x_i \in \text{Class1}$, else $x_i \in \text{Class0}$
- b. Classify the remaining points, x_{51}, \dots, x_{100} using KNN. Perform this for $k=1,2,3,4,5,20,30$

```
import numpy as np
from collections import Counter
def generate_data():
    # Generate 100 random values between 0 and 1
    x = np.random.rand(100)
    return x
def label_data(x):
    # Label the first 50 points based on the rule
    labels = np.zeros(50, dtype=int) # Class 0 for xi <= 0.5
    labels[x[:50] > 0.5] = 1 # Class 1 for xi > 0.5
    return labels
def knn_classify(x_train, y_train, x_test, k):
    predictions = []
    for x in x_test:
        # Calculate distances from x to all training points
        distances = np.abs(x_train - x)
        # Get the indices of the k nearest neighbors
        nearest_indices = distances.argsort()[:k]
        # Get the labels of the k nearest neighbors
        nearest_labels = y_train[nearest_indices]
        # Determine the majority class
        majority_class = Counter(nearest_labels).most_common(1)[0][0]
        predictions.append(majority_class)
    return np.array(predictions)
```

```

def main():
    # Generate data
    x = generate_data()
    print("Generated points",x)
    # Label the first 50 points
    x_train = x[:50]
    y_train = label_data(x)
    print("labels of first 50 points",y_train)
    # Classify the remaining points using KNN
    x_test = x[50:]
    k_values = [1, 2, 3, 4, 5, 20, 30]
    print("KNN Classification Results:")
    for k in k_values:
        if k > len(x_train):
            print(f"k={k} is larger than the training set size and is skipped.")
            continue
        y_pred = knn_classify(x_train, y_train, x_test, k)
        print(f"k={k}: Predicted Classes: {y_pred}")
main()

```

Output

```

Generated points [0.74722035 0.75293165 0.35157064 0.11841091 0.32885605 0.07889337
0.20110261 0.01545388 0.8885711 0.92273204 0.19312117 0.12069055
0.83082426 0.65544183 0.65193073 0.56487092 0.66320956 0.90142367
0.76538335 0.12920818 0.22374779 0.68030336 0.58157345 0.62923884
0.46901236 0.48244588 0.51581747 0.38687083 0.30091563 0.63100299
0.80330766 0.39982626 0.39493118 0.11055241 0.01196966 0.82556546
0.30515055 0.13269029 0.1769821 0.76510273 0.08966337 0.38265816
0.9706964 0.92234415 0.00567237 0.35738169 0.34243248 0.71423301
0.09659369 0.99099885 0.26925519 0.40038672 0.47370594 0.83908304
0.46438589 0.83447033 0.21368933 0.43521261 0.27924869 0.64597382
0.19086124 0.83156813 0.79270232 0.60888411 0.599663 0.41156996
0.7374764 0.65266158 0.50391039 0.10616744 0.38778425 0.00698569
0.36526828 0.96938636 0.94572554 0.18282409 0.94485379 0.57457604
0.40866847 0.05020679 0.47256857 0.62967589 0.35570877 0.9600593
0.180249 0.98360184 0.29926218 0.99330739 0.56266349 0.61413277
0.15862791 0.12450683 0.94360382 0.31096008 0.70567067 0.14349801
0.40897897 0.0194384 0.65569158 0.9338463 ]
labels of first 50 points [1 1 0 0 0 0 0 0 1 1 0 0 1 1 1 1 1 1 1 0 0 1 1 1 0 0 1 1 1 0 0 1 0 0 0 0 0 0 1 0
0 0 1 0 0 1 1 0 0 0 1 0 1]
KNN Classification Results:
k=1: Predicted Classes: [0 0 0 1 0 1 0 0 0 1 0 1 1 1 1 0 1 1 1 0 0 0 0 1 1 0 0 0 1 1 0 0 0 1 0 1 0 1 0 1 0
1 1 1 0 0 1 0 1 0 0 0 1 1]
k=2: Predicted Classes: [0 0 0 1 0 1 0 0 0 1 0 1 1 1 1 0 1 1 1 0 0 0 0 1 1 0 1 1 0 0 0 1 0 1 0 1 0 1 0
1 1 1 0 0 1 0 1 0 0 0 1 1]
k=3: Predicted Classes: [0 0 0 1 0 1 0 0 0 1 0 1 1 1 1 0 1 1 0 0 0 0 0 1 1 0 1 1 0 0 0 1 0 1 0 1 0 1 0
1 1 1 0 0 1 0 1 0 0 0 1 1]
k=4: Predicted Classes: [0 0 0 1 0 1 0 0 0 1 0 1 1 1 1 0 1 1 1 0 0 0 0 1 1 0 1 1 0 0 0 1 0 1 0 1 0 1 0
1 1 1 0 0 1 0 1 0 0 0 1 1]
k=5: Predicted Classes: [0 0 0 1 0 1 0 0 0 1 0 1 1 1 1 0 1 1 1 0 0 0 0 1 1 0 1 1 0 0 0 1 0 1 0 1 0 1 0
1 1 1 0 0 1 0 1 0 0 0 1 1]
k=20: Predicted Classes: [0 0 0 1 0 1 0 0 0 1 0 1 1 1 1 0 1 1 0 0 0 0 0 1 1 0 1 1 0 0 0 1 0 1 0 1 0 1 0
1 1 1 0 0 1 0 1 0 0 0 1 1]
k=30: Predicted Classes: [0 0 0 1 0 1 0 0 0 1 0 1 1 1 1 0 1 1 1 0 0 0 0 1 1 0 1 1 0 0 0 1 0 1 0 1 0 1 0
1 1 1 0 0 1 0 1 0 0 0 1 1]

```

EXPERIMENT 6

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 matplotlib.pyplot as plt
import pandas as pd
import numpy as np
def kernel(point,xmat,k):
    m,n = np.shape(xmat)
    weights = np.mat(np.eye((m)))
    for j in range(m):
        diff=point - X[j]
        weights[j,j] = np.exp(diff*diff.T/(-2.0*k**2))
    return weights

def localWeight(point,xmat,yamat,k):
    wei=kernel(point,xmat,k)
    W=(X.T*(wei*X)).I*(X.T*(wei*yamat.T))
    return W

def localweightregression(xmat,yamat,k):
    m,n=np.shape(xmat)
    ypred=np.zeros(m)
    for i in range(m):
        ypred[i]=xmat[i]*localWeight(xmat[i],xmat,yamat,k)
    return ypred

def graphplot(X,ypred):
    sortindex=X[:,1].argsort(0)
    xsort=X[sortindex][:,0]
    fig=plt.figure()
    ax=fig.add_subplot(1,1,1)
```

```

ax.scatter(bill,tip,color='green')
ax.plot(xsort[:,1],ypred[sortindex],color='red',linewidth=4)
plt.xlabel('Total Bill')
plt.ylabel('Tip')
plt.show();

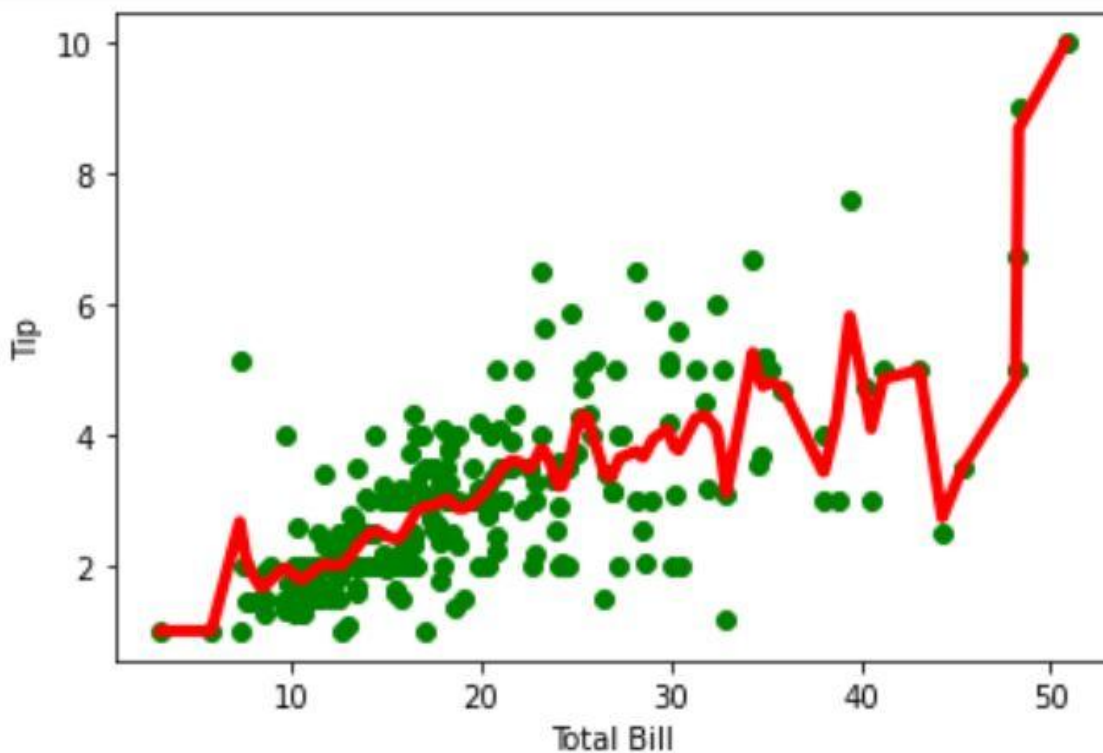
```

```

data=pd.read_csv('data10_tips.csv')
bill=np.array(data.total_bill)
tip=np.array(data.tip)
mbill=np.mat(bill)
mtip=np.mat(tip)
m=np.shape(mbill)[1]
one=np.mat(np.ones(m))
X=np.hstack((one.T,mbill.T))
ypred=localweightregression(X,mtip,0.5)
graphplot(X,ypred)

```

Output



EXPERIMENT 7

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 numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
from sklearn.datasets import load_boston
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
boston_dataset = load_boston()
boston_dataset.keys()
boston = pd.DataFrame(boston_dataset.data, columns=boston_dataset.feature_names)
boston.head()
boston['MEDV'] = boston_dataset.target
X = pd.DataFrame(np.c_[boston['LSTAT'], boston['RM']], columns = ['LSTAT', 'RM'])
Y = boston['MEDV']
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.2, random_state=5)
lin_model = LinearRegression()
lin_model.fit(X_train, Y_train)
y_train_predict = lin_model.predict(X_train)
rmse = (np.sqrt(mean_squared_error(Y_train, y_train_predict)))
r2 = r2_score(Y_train, y_train_predict)
print("-----")
print("Linear Regression")
print("The model performance for training set")
print("-----")
print('RMSE is {}'.format(rmse))
print('R2 score is {}'.format(r2))
```

```

print("\n")
# model evaluation for testing set
y_test_predict = lin_model.predict(X_test)
# root mean square error of the model
rmse = (np.sqrt(mean_squared_error(Y_test, y_test_predict)))
# r-squared score of the model
r2 = r2_score(Y_test, y_test_predict)
print("The model performance for testing set")
print("-----")
print('RMSE is {}'.format(rmse))
print('R2 score is {}'.format(r2))
plt.scatter(Y_test, y_test_predict)
plt.xlabel("Actual Prices")
plt.ylabel("Predicted Prices")
plt.show()

# Polynomial regression
print("-----")
print("Polynomial Regression")
url = 'https://archive.ics.uci.edu/ml/machine-learning-databases/auto-mpg/auto-mpg.data'
auto_df = pd.read_csv(url, delim_whitespace=True)
auto_df.columns = ['mpg', 'cylinders', 'displacement', 'horsepower', 'weight', 'acceleration',
'model_year', 'origin', 'car_name']
auto_df.horsepower = pd.to_numeric(auto_df.horsepower, errors='coerce')
auto_df.car_name = auto_df.car_name.astype('string')
auto_df = auto_df[auto_df.horsepower.notnull()]
X_train, X_test, y_train, y_test = train_test_split(auto_df[['displacement']], auto_df['mpg'],
test_size=0.25, random_state=101)
transformer = PolynomialFeatures(degree=2, include_bias=True)
X_p_train = transformer.fit_transform(X_train)
model = LinearRegression()
model.fit(X_p_train, y_train)

```

```

X_p_test = transformer.fit_transform(X_test)
y_predict = model.predict(X_p_test)
plt.scatter(X_test, y_test, color='dodgerblue')
sorted_Xy = sorted(zip(X_test.displacement, y_predict))
X_s, y_s = zip(*sorted_Xy)
plt.plot(X_s, y_s, color='red')
plt.xlabel('Displacement')
plt.ylabel('MPG')
plt.show()
r2=r2_score(y_test, y_predict)
mse=mean_squared_error(y_test, y_predict, squared=False)
print(f"R squared value: {r2}")
print(f"Mean Squared Error: {mse}")

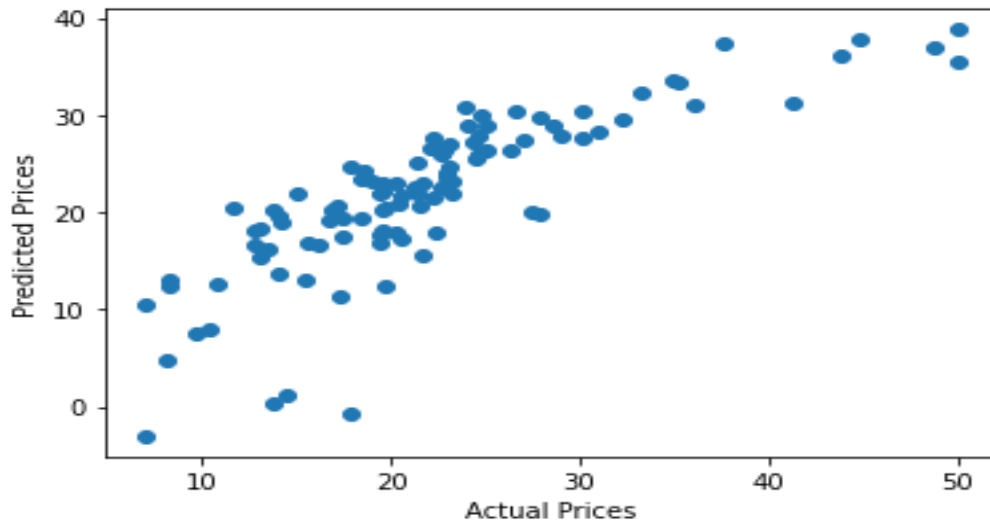
```

Output

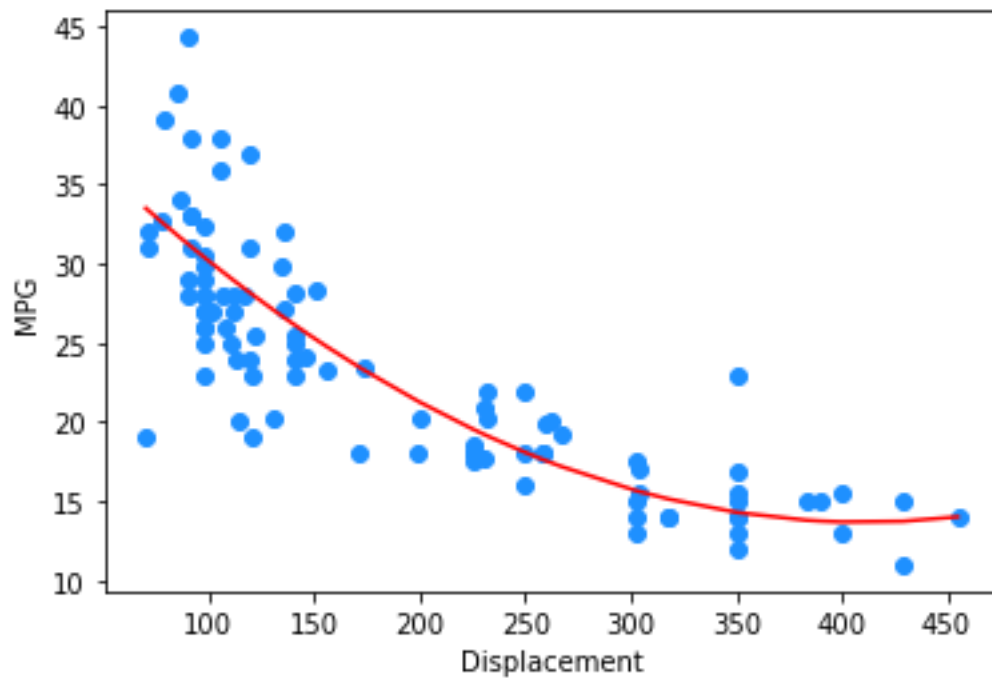
```

-----
Linear Regression
The model performance for training set
-----
RMSE is 5.6371293350711955
R2 score is 0.6300745149331701
The model performance for testing set
-----
RMSE is 5.137400784702911
R2 score is 0.6628996975186953

```



Polynomial Regression



R squared value: 0.702509211169476

Mean Squared Error: 4.008353173218597

EXPERIMENT 8

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

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

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

# Split the dataset 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 train a decision tree classifier
clf = DecisionTreeClassifier(random_state=42)
clf.fit(X_train, y_train)

# Evaluate the model on the test data
y_pred = clf.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy on test data: {accuracy:.2f}")

# Visualize the decision tree
```

```

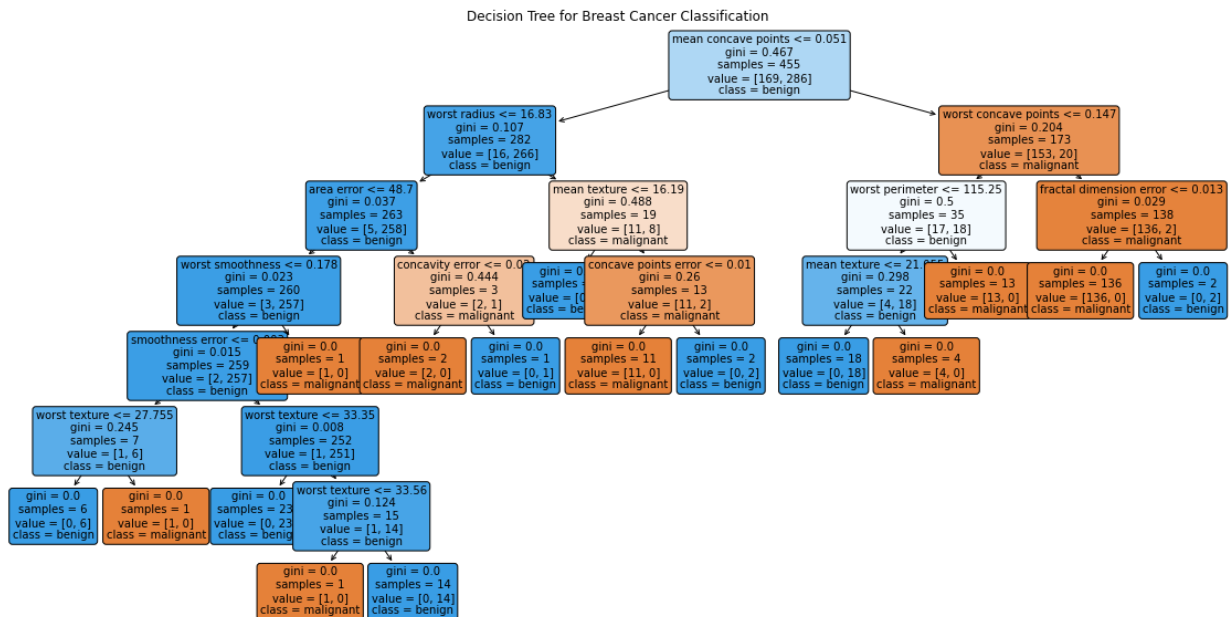
plt.figure(figsize=(20, 10))
plot_tree(clf, feature_names=feature_names, class_names=target_names, filled=True,
rounded=True, fontsize=10)
plt.title("Decision Tree for Breast Cancer Classification")
plt.show()

# Classify a new sample
# Example new sample: mean radius=17.99, mean texture=10.38, mean perimeter=122.80, mean
area=1001.0, etc.
new_sample = np.array([17.99, 10.38, 122.8, 1001.0, 0.1184, 0.2776, 0.3001, 0.1471, 0.2419,
0.07871,1.095, 0.9053, 8.589, 153.4, 0.006399, 0.04904, 0.05373, 0.01587, 0.03003, 0.006193,
25.38, 17.33, 184.6, 2019.0, 0.1622, 0.6656, 0.7119, 0.2654, 0.4601, 0.1189]).reshape(1, -1)
# Predict the class of the new sample
new_prediction = clf.predict(new_sample)
print(f"Predicted class for the new sample: {target_names[new_prediction[0]]}")

```

Output

Accuracy on test data: 0.95



Predicted class for the new sample: malignant

EXPERIMENT 9

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 numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import fetch_olivetti_faces
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score
# Load the Olivetti Face dataset
data = fetch_olivetti_faces(shuffle=True, random_state=42)
# Extract the data and labels
X = data.data # Feature data (images flattened into vectors)
y = data.target # Labels (target classes for each face)
# Split the dataset 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)

# Initialize the Gaussian Naive Bayes classifier
nb_classifier = GaussianNB()
# Train the Naive Bayes classifier
nb_classifier.fit(X_train, y_train)
# Display all the test images with their actual labels
num_test_images = X_test.shape[0]
num_cols = 10 # Number of columns in the grid
num_rows = num_test_images // num_cols + (num_test_images % num_cols != 0) # Calculate
number of rows
plt.figure(figsize=(15, 15))
# Loop through the test set and display each image with actual label
for i in range(num_test_images):
    plt.subplot(num_rows, num_cols, i + 1)
```

```

plt.imshow(X_test[i].reshape(64, 64), cmap='gray') # Reshape to 64x64 image
plt.title(f"True: {y_test[i]}")
plt.axis('off')
plt.tight_layout() # Adjust layout to prevent overlap
plt.show()
# Predict the labels on the test set
y_pred = nb_classifier.predict(X_test)
# Display test images with predicted and actual labels
plt.figure(figsize=(15, 15))

# Loop through the test set and display each image with predicted and actual label
for i in range(num_test_images):
    plt.subplot(num_rows, num_cols, i + 1)
    plt.imshow(X_test[i].reshape(64, 64), cmap='gray') # Reshape to 64x64 image
    plt.title(f"Pred: {y_pred[i]}\nTrue: {y_test[i]}")
    plt.axis('off')
plt.tight_layout() # Adjust layout to prevent overlap
plt.show()

# Compute the accuracy of the classifier
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy of the Naive Bayes Classifier: {accuracy * 100:.2f}%")

```

Output





Accuracy of the Naive Bayes Classifier: 77.50%

EXPERIMENT 10

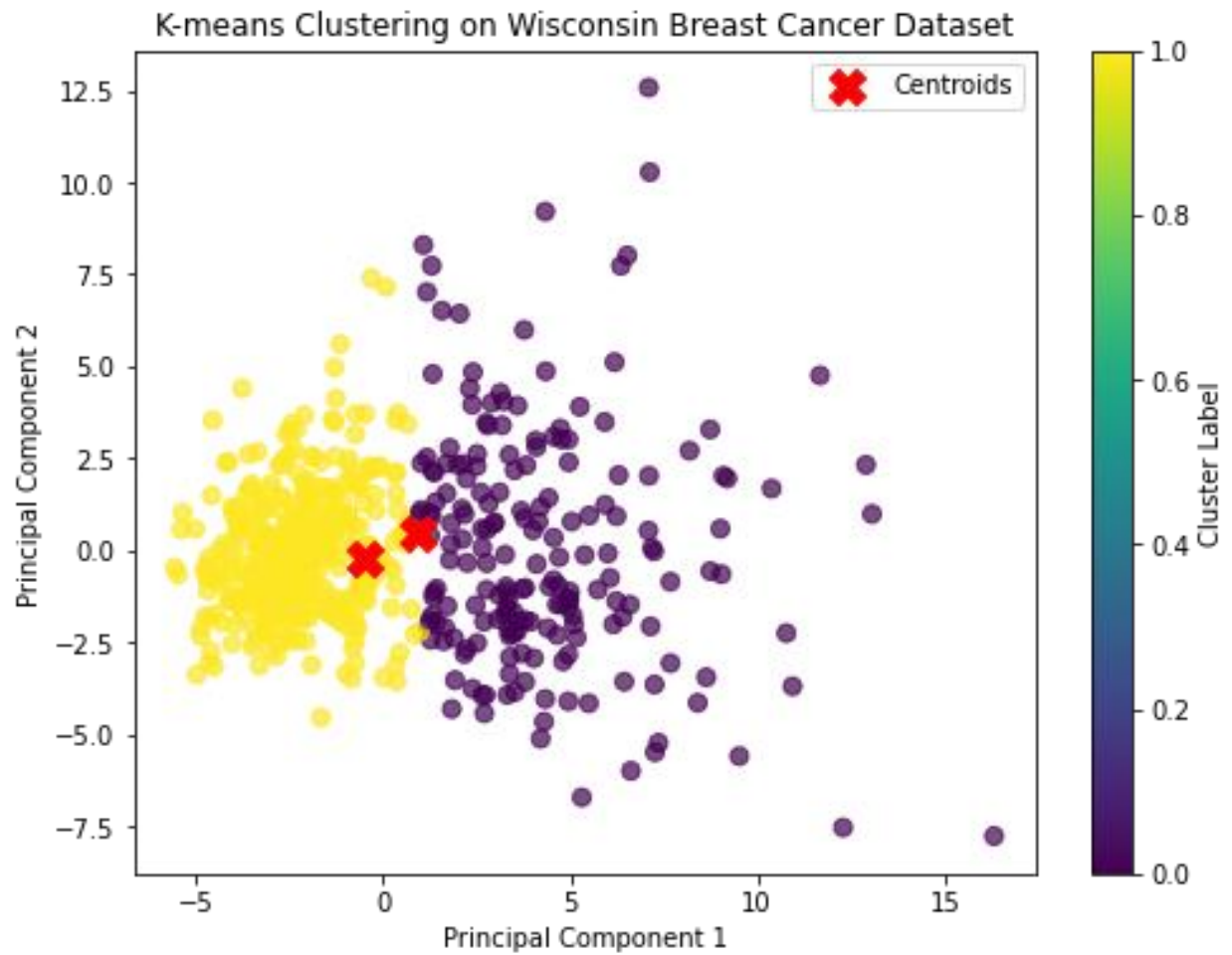
10. Develop a program to implement k-means clustering using Wisconsin Breast Cancer data set and visualize the clustering result.

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

# Load the Wisconsin Breast Cancer dataset
cancer_data = datasets.load_breast_cancer()
X = cancer_data.data
y = cancer_data.target
# Standardize the data
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
# Apply K-means clustering
kmeans = KMeans(n_clusters=2, random_state=42)
y_kmeans = kmeans.fit_predict(X_scaled)
# Reduce dimensionality to 2D using PCA for visualization
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X_scaled)
# Plot the clusters
plt.figure(figsize=(8, 6))
plt.scatter(X_pca[:, 0], X_pca[:, 1], c=y_kmeans, cmap='viridis', s=50, alpha=0.7)
plt.scatter(kmeans.cluster_centers_[0, :], kmeans.cluster_centers_[0, 1], c='red', marker='X',
s=200, label='Centroids')
plt.title('K-means Clustering on Wisconsin Breast Cancer Dataset')
plt.xlabel('Principal Component 1')
```

```
plt.ylabel('Principal Component 2')
plt.legend()
plt.colorbar(label='Cluster Label')
plt.show()
```

Output



Viva Questions

1. Why do you mean by outliers ?
2. Distinguish between histogram and box plot?
3. Explain the characteristics of California Housing dataset.
4. How histogram can be created using python?
5. How box plot can be created using python?
6. What do you mean by correlation matrix?
7. How can you interpret the pair plots to understand the relationship between two variables?
8. What is the significance of the Pearson correlation coefficient in a correlation matrix?
9. What is the use of heatmap and pairplot?
10. List and explain about functions used in creating heatmap and pairplot?
11. What is the purpose of Principal Component Analysis (PCA) in machine learning?
12. Explain the characteristics of iris dataset.
13. What do you mean by dimensionality reduction?
14. List the steps of PCA algorithm?
15. List the steps of Find S algorithm?
16. What is the use of Find S algorithm?
17. List the limitations of Find S algorithm?
18. How does the Find S algorithm generalize the hypothesis?
19. What is the k-NN algorithm?
20. List the steps of k-NN algorithm?
21. What do you mean by instance-based learning?
22. How do you calculate Euclidean distance between two points ?
23. How does increasing k affect the model's performance in k-NN algorithm?
24. What is Locally Weighted Regression?
25. List the steps of locally weighted regression?
26. Explain weight function in locally weighted regression?
27. Explain hypothesis function in locally weighted regression?
28. Explain cost function in locally weighted regression?
29. What is linear regression?
30. What is polynomial regression?

31. What is the difference between Linear and Polynomial Regression?
32. How does Polynomial Regression handle non-linearity?
33. List the characteristics of Boston Housing Dataset and Auto MPG Dataset.
34. What are the different types of nodes present in decision tree?
35. What is entropy?
36. List the steps used in constructing a decision tree using ID3 algorithm?
37. List the characteristics of breast cancer data set?
38. List the disadvantages of decision trees?
39. What is the Naive Bayes algorithm?
40. List the steps of Naive Bayes algorithm?
41. List the characteristics of Olivetti Face Data set.
42. List the applications of naive bayes algorithm?
43. How do you calculate accuracy of the naive bayes classifier?
44. What is clustering?
45. What is k-means clustering?
46. List the steps of k-means clustering?
47. What are the disadvantages of k-means clustering?
48. List the characteristics of Wisconsin Breast Cancer data set.
49. How clustering results can be visualized?