

# Machine Learning

Practical File

BACHELOR OF TECHNOLOGY

Information Technology

SUBMITTED BY

Muskan kaur

URN- 2104534

CRN- 2121075



GURU NANAK DEV ENGINEERING COLLEGE

LUDHIANA-141006, INDIA

## Contents

- 1 BASIC PYTHON PROGRAM

## 2 NUMPY LIBRARY

2.1 Numpy library to add two number

2.2 Numpy library to find mean of an array

## 3 PANADA LIBRARY

3.1 Using Panda for dataframing.

3.2 series in Panda Library

## 4 MATPLOTLIB

4.1 Plot

4.2 Scatter Plot 4.3 Pie Chart

## 5 SCIKIT-LEARN

# I SUPERVISED LEARNING

## 6 CLASSIFICATION

6.1 Decision Tree

6.2 K-Nearest Neighbor

6.3 Kernel Support Vector Machine

6.4 Confusion Matrix in Machine Learning

## 7 REGRESSION

7.1 Simple linear regression

7.2 Polynomial regression

7.3 Support vector regression

7.4 Decision trees

7.5 Random forest

# II UNSUPERVISED LEARNING

## 8 CLUSTERING

8.1 K-mean Clustering

8.2 K-mode Clustering

### III PROGRAMS

#### 9 SIMPLE LINEAR REGRESSION

9.1 Importing the library

9.2 Importing the dataset

9.3 Splitting the dataset into the Training set and Test set

9.4 Training the Simple Linear Regression model on the Training set

9.5 Predicting the Test set result

9.6 Visualising the Training set result

9.7 Visualising the Testing set result

#### 10 K- MEANS CLUSTERING.

10.1 Importing the libraries

10.2 Importing the dataset

10.3 Using the elbow method to find the optimal number of clusters

10.4 Training the K-Means model on the dataset 10.5 Visualising the clusters

#### 11 Polynomial Regression

11.1 Importing the library

11.2 Importing the dataset

11.3 Training the Linear Regression model on the whole dataset

11.4 Training the Polynomial Regression model on the whole dataset

11.5 Visualising the Linear Regression results

11.6 Visualising the Polynomial Regression results

11.7 Visualising the Polynomial Regression results (for higher resolution and smoother curve)

#### 12 Support Vector Regression (SVR)

12.1 Importing the library

- 12.2 Importing the dataset
- 12.3 Feature Scaling
- 12.4 Training the SVR model on the whole dataset
- 12.5 Predicting a new result
- 12.6 Visualising the SVR results
- 12.7 Visualising the SVR results (for higher resolution and smoother curve)
- 13 Hierarchical Clustering.
- 13.1 Importing the libraries
- 13.2 Importing the dataset
- 13.3 Using the dendrogram to find the optimal numbers of clusters
- 13.4 Training the Hierarchical Clustering model on the dataset
- 13.5 Visualising the cluster

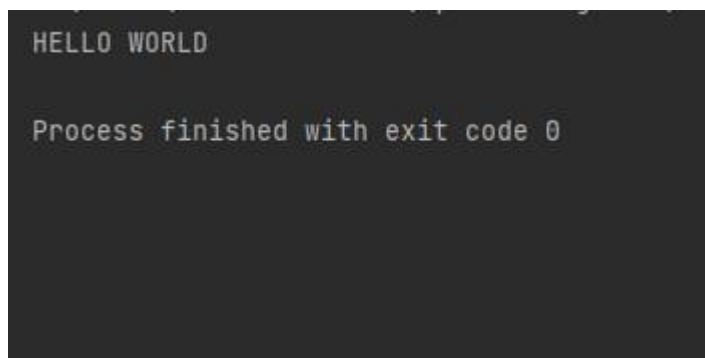
## 1 BASIC PYTHON PROGRAM

### 1.1 Print HELLO WORLD .

CODE :

```
print("HELLO WORLD")
```

OUTPUT :



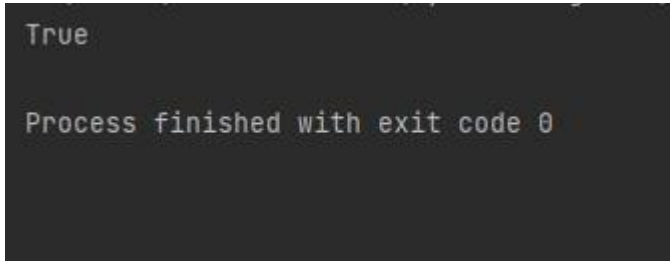
```
HELLO WORLD

Process finished with exit code 0
```

### 1.2 Else If Condition CODE :

```
a=10 if(1):print("True")
else:print("False")
```

OUTPUT :



```
True

Process finished with exit code 0
```

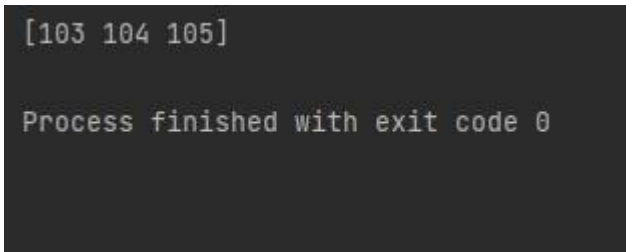
**2. NUMPY LIBRARY :** NumPy is a Python library used for working with arrays. It also has functions for working in domain of linear algebra, fourier transform, and matrices.

2.1 Numpy library to add two number.

CODE :

```
import numpy as np
a=np.array([98,99,100])
b=np.array([5])
c=np.array(a+b) print(c)
```

OUTPUT :



```
[103 104 105]

Process finished with exit code 0
```

2.2 Numpy library to find mean of an array.

CODE :

```
import numpy as np
marks=[12,13,15,17,20] marks_arr=np.array(marks)
```

```
m=np.mean(marks_arr) print("mean of  
list",m)
```

OUTPUT :

```
mean of list 15.4  
  
Process finished with exit code 0
```

**3. PANADA LIBRARY** : Pandas is an open source Python package that is most widely used for data science/data Analysis and machinelearning tasks.

**3.1 Using Panda for dataframing.**

CODE :

```
import pandas as pd s=pd.array([23,43,76,98,12,45],[45,76,43,98,7,1])  
print(pd.DataFrame(s))
```

OUTPUT :

```
      0  
0  23  
1  43  
2  76  
3  98  
4  12  
5  45
```

**3.2 Series in Panda Library.**

CODE :

```
import pandas as pd phonebook={  
    'kiran':6734787422,  
    'kamal':7737366548,  
    'preeti':9865444679 }  
print(pd.Series(phonebook)) Output:
```

```
kiran    6734787422
kamal    7737366548
preeti   9865444679
dtype: int64

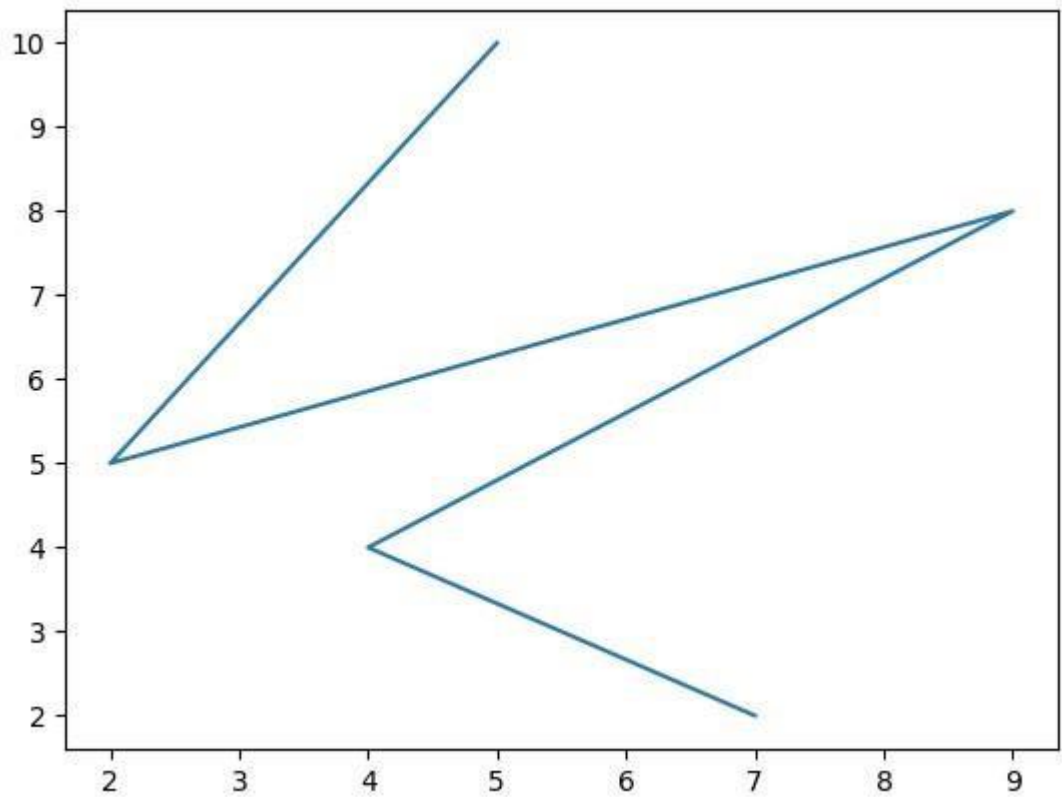
Process finished with exit code 0
```

4 **MATPLOTLIB** : Matplotlib is an amazing visualization library in Python for 2D plots of arrays. Matplotlib is a multiplatform data visualization library built on NumPy arrays and designed to work with the broader SciPy stack

#### 4.1 Plot.

```
from matplotlib import pyplot as plt
x=[5,2,9,4,7] y=[10,5,8,4,2] plt.plot(x,y)
plt.show()
```

OUTPUT :

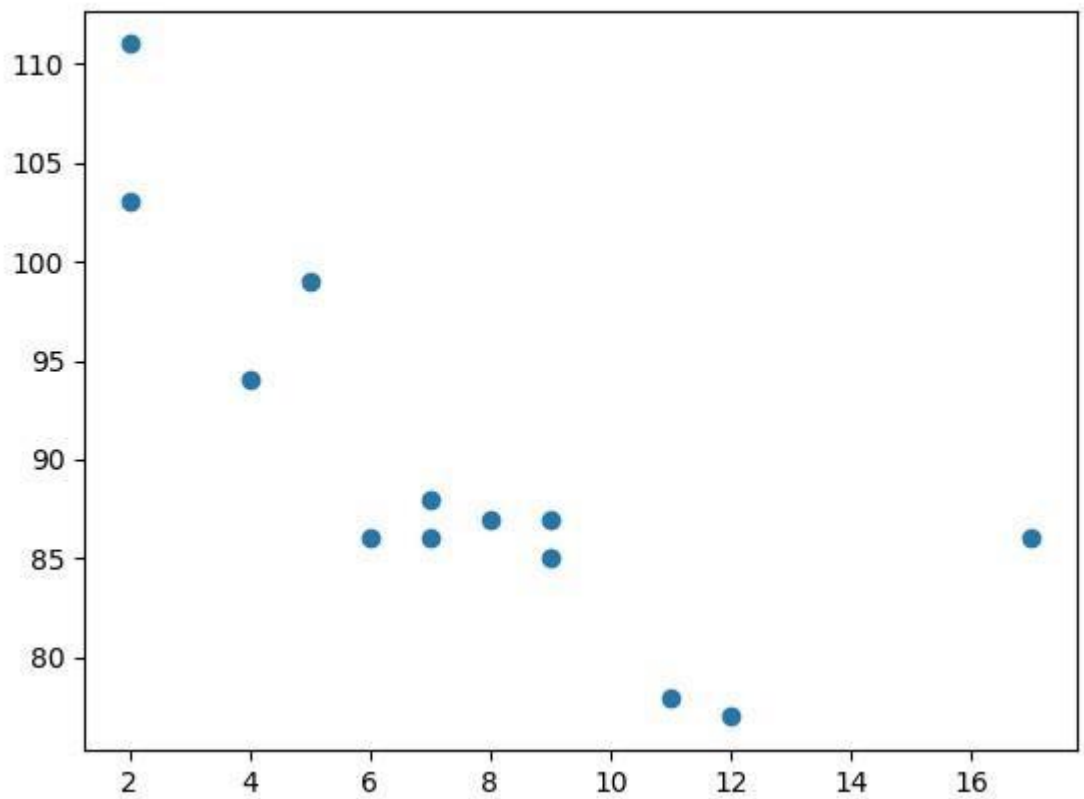


#### 4.2 Scatter Plot.

```
import numpy as np x = np.array([5,7,8,7,2,17,2,9,4,11,12,9,6]) y =  
np.array([99,86,87,88,111,86,103,87,94,78,77,85,86]) plt.scatter(x, y) plt.show()
```

OUTPUT:

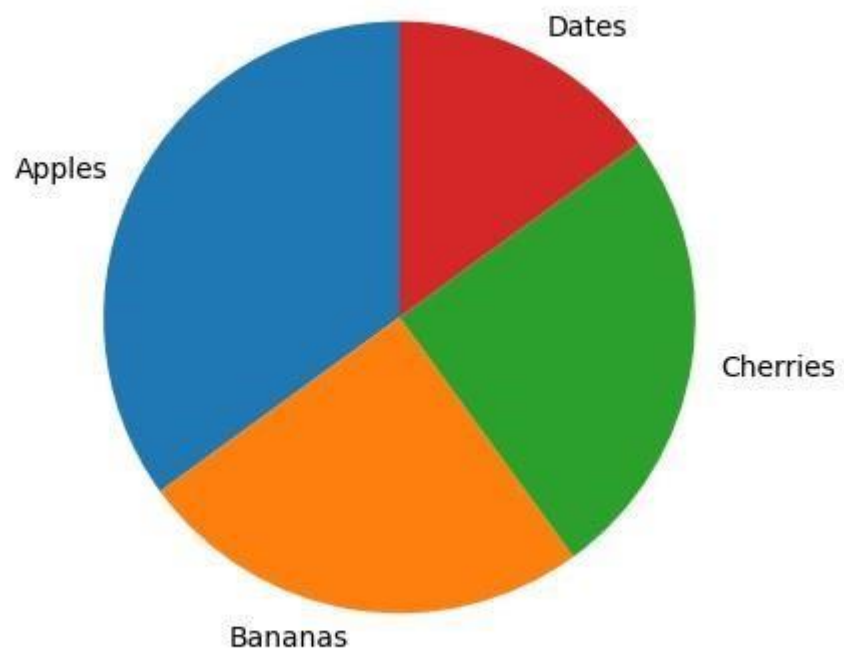




#### 4.3 Pie Chart.

```
import numpy as np y = np.array([35, 25, 25, 15]) mylabels =  
["Apples", "Bananas", "Cherries", "Dates"] plt.pie(y, labels = mylabels,  
startangle = 90) plt.show()
```

OUTPUT:



Artificial Intelligence

July 20, 2022



Artificial Intelligence (AI), the ability of a digital computer or computer-controlled robot to perform tasks commonly associated with intelligent beings. The term is frequently applied to the project of developing systems endowed with the intellectual processes characteristic of humans, such as the ability to reason, discover meaning, generalize, or learn from past experience. Since the development of the digital computer in the 1940s, it has been demonstrated that computers can be programmed to carry out very complex tasks as, for example, discovering proofs for mathematical theorems or playing chess with great proficiency. Still, despite continuing advances in computer processing speed and memory capacity, there are as yet no programs that can match human ability over wider domains or in tasks requiring much everyday knowledge. 5

## SCIKIT-LEARN

Scikit-learn (Sklarn) is the most useful and robust library for machine learning in Python. It provides a selection of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction via a

consistence interface in Python. This library, which is largely written in Python, is built upon NumPy, SciPy and Matplotlib.

## Features

Rather than focusing on loading, manipulating and summarising data, Scikit-learn library is focused on modeling the data. Some of the most popular groups of models provided by Sklearn are as follows =

1. Supervised Learning algorithms = Almost all the popular supervised learning algorithms, like Linear Regression, Support Vector Machine (SVM), Decision Tree etc., are the part of scikitlearn.
2. Unsupervised Learning algorithms = On the other hand, it also has all the popular unsupervised learning algorithms from clustering, factor analysis, PCA (Principal Component Analysis) to unsupervised neural networks.
3. Clustering= This model is used for grouping unlabeled data.
4. Cross Validation = It is used to check the accuracy of supervised models on unseen data.
5. Dimensionality Reduction = It is used for reducing the number of attributes in data which can be further used for summarisation, visualisation and feature selection.
6. Ensemble methods = As name suggest, it is used for combining the predictions of multiple supervised models.

## Pros and cons of scikit-learn Pros:

The library is distributed under the BSD license, making it free with minimum legal and licensing restrictions.

It is easy to use.

The scikit-learn library is very versatile and handy and serves real-world purposes like the prediction of consumer behavior, the creation of neuroimages, etc.

Scikit-learn is backed and updated by numerous authors, contributors, and a vast international online community.

The scikit-learn website provides elaborate API documentation for users who want to integrate the algorithms with their platforms.

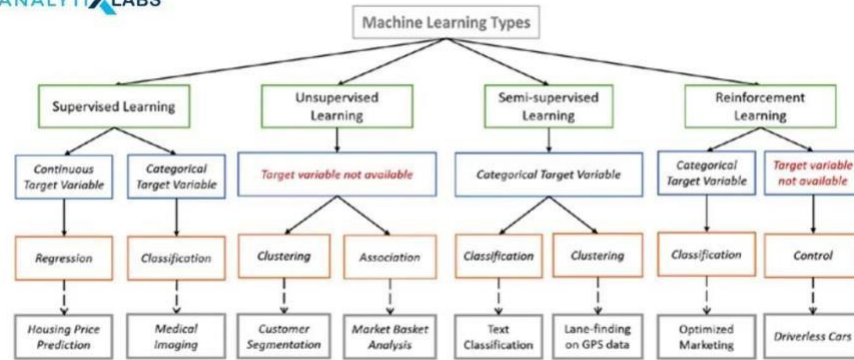
Con:

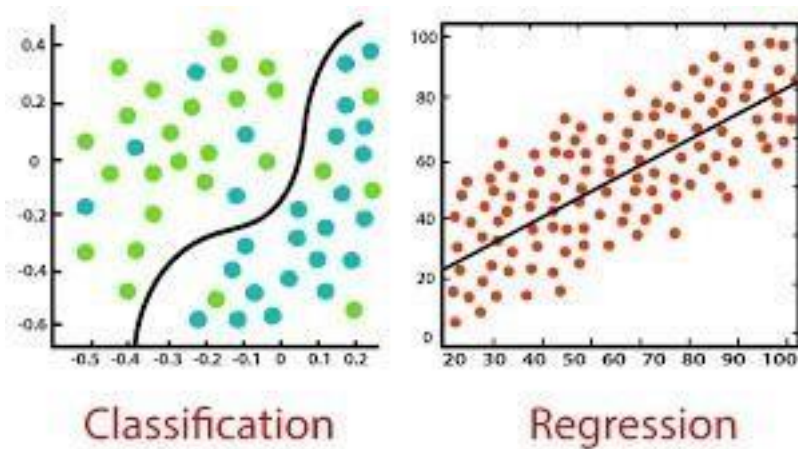
It is not the best choice for in-depth learning.

It is not optimized for graph algorithms, and it is not very good at string processing. For example, scikitlearn does not provide a built-in way to produce a simple word cloud. Scikit-learn doesn't have a strong linear algebra library, hence scipy and numpy are used.

## TYPES OF MACHINE LEARNING

1. SUPERVISED LEARNING.
2. UNSUPERVISED LEARNING.



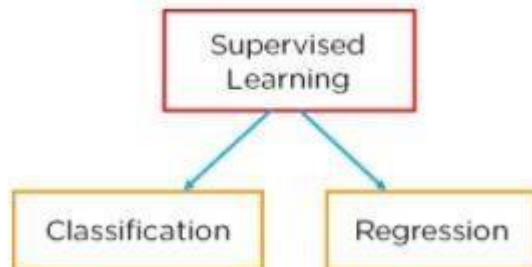


Part I

## SUPERVISED LEARNING.

Suppose you are trying to learn a new concept in maths and after solving a problem, you may refer to the solutions to see if you were right or not. Once you are confident in your ability to solve a particular type of problem, you will stop referring to the answers and solve the questions put before you by yourself.

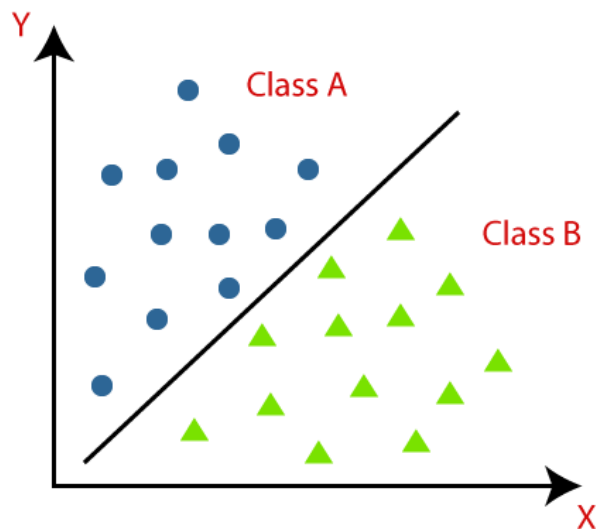
This is also how Supervised Learning works with machine learning models. In Supervised Learning, the model learns by example. Along with our input variable, we also give our model the corresponding correct labels. While training, the model gets to look at which label corresponds to our data and hence can find patterns between our data and those labels.



6 CLASSIFICATION.

Classification is a process of categorizing a given set of data into classes. It can be performed on both structured or unstructured data. The process starts with predicting the class of given data points. The classes are often referred to as target, label or categories.

Classification is defined as the process of recognition, understanding, and grouping of objects and ideas into preset categories a.k.a subpopulations. With the help of these pre-categorized training datasets, classification in machine learning programs leverage a wide range of algorithms to classify future datasets into respective and relevant categories.

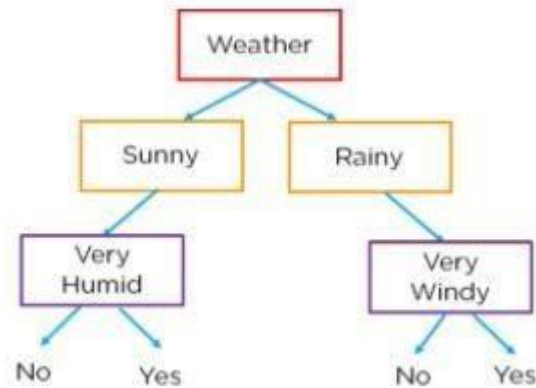


Classification Models

### 6.1 Decision Tree

A Decision Tree is an algorithm that is used to visually represent decision-making. A Decision Tree can be made by asking a yes/no question and splitting the answer to lead to another decision. The question is at the node and it places the resulting decisions below at the leaves. The tree depicted below is used to decide if we can play tennis.



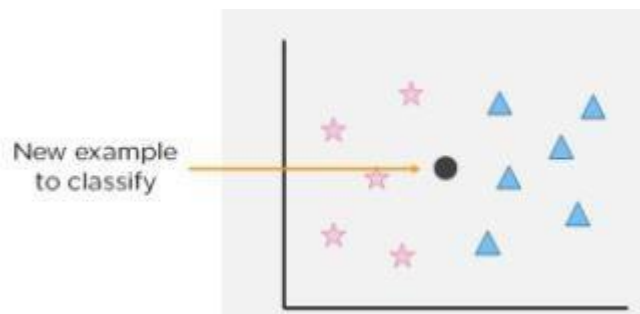


### Decision Tree

In the above figure, depending on the weather conditions and the humidity and wind, we can systematically decide if we should play tennis or not. In decision trees, all the False statements lie on the left of the tree and the True statements branch off to the right. Knowing this, we can make a tree which has the features at the nodes and the resulting classes at the leaves.

### 6.2 K-Nearest Neighbor

K-Nearest Neighbor is a classification and prediction algorithm that is used to divide data into classes based on the distance between the data points. K-Nearest Neighbor assumes that data points which are close to one another must be similar and hence, the data point to be classified will be grouped with the closest cluster. data-classified



Data to be classified

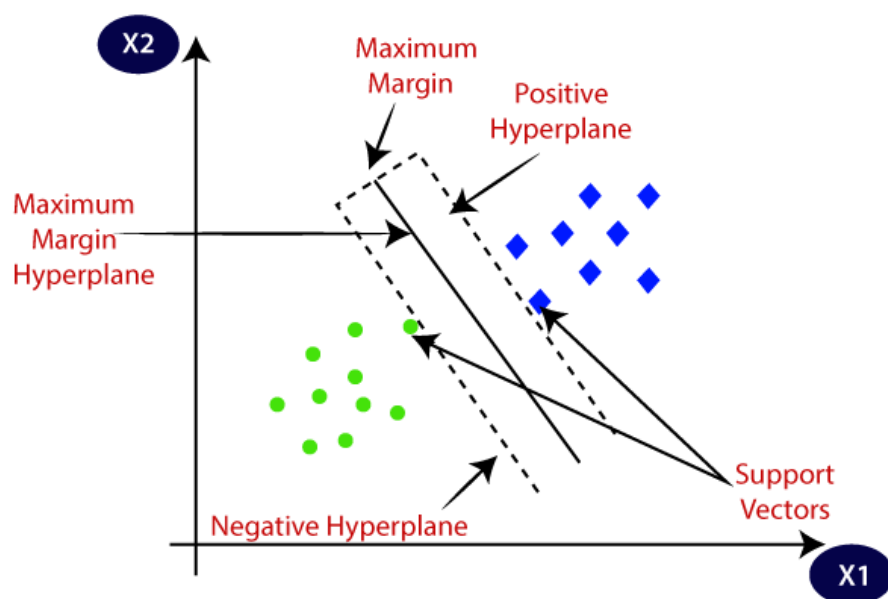


Classification using K-Nearest Neighbours

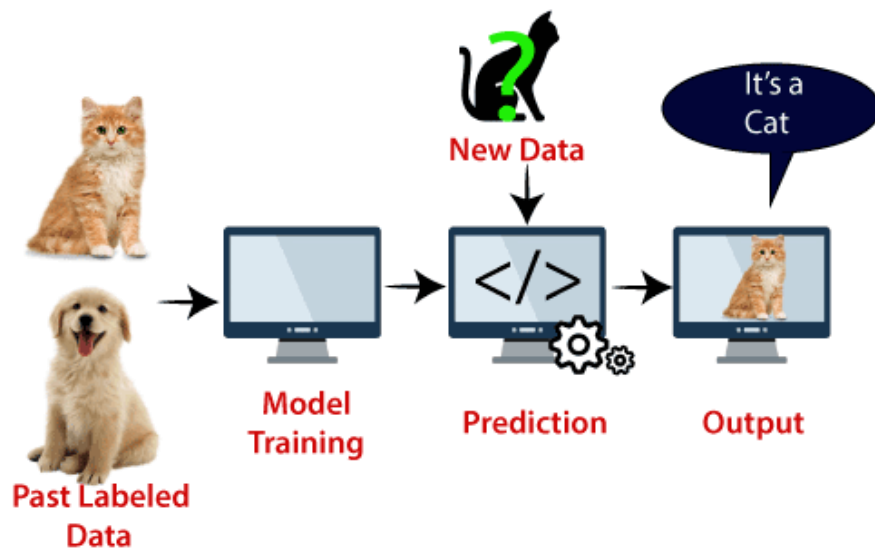
### 6.3 Kernel Support Vector Machine

The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane.

SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called as support vectors, and hence algorithm is termed as Support Vector Machine. Consider the below diagram in which there are two different categories that are classified using a decision boundary or hyperplane: Support Vector Machine Algorithm



Example: SVM can be understood with the example that we have used in the KNN classifier. Suppose we see a strange cat that also has some features of dogs, so if we want a model that can accurately identify whether it is a cat or dog, so such a model can be created by using the SVM algorithm. We will first train our model with lots of images of cats and dogs so that it can learn about different features of cats and dogs, and then we test it with this strange creature. So as support vector creates a decision boundary between these two data (cat and dog) and choose extreme cases (support vectors), it will see the extreme case of cat and dog. On the basis of the support vectors, it will classify it as a cat. Consider the below diagram:



#### 6.4 Confusion Matrix in Machine Learning

**Confusion Matrix in Machine Learning** The confusion matrix is a matrix used to determine the performance of the classification models for a given set of test data. It can only be determined if the true values for test data are known. The matrix itself can be easily understood, but the related terminologies may be confusing. Since it shows the errors in the model performance in the form of a matrix, hence also known as an error matrix. Some features of Confusion matrix are given below:

For the 2 prediction classes of classifiers, the matrix is of 2\*2 table, for 3 classes, it is 3\*3 table, and so on.

The matrix is divided into two dimensions, that are predicted values and actual values along with the total number of predictions.

Predicted values are those values, which are predicted by the model, and actual values are the true values for the given observations.

It looks like the below table:

n = total predictions	Actual: No	Actual: Yes
Predicted: No	True Negative	False Positive
Predicted: Yes	False Negative	True Positive

### Confusion Matrix in Machine Learning

The above table has the following cases:

True Negative: Model has given prediction No, and the real or actual value was also No.

True Positive: The model has predicted yes, and the actual value was also true.

False Negative: The model has predicted no, but the actual value was Yes, it is also called as Type-II error.

False Positive: The model has predicted Yes, but the actual value was No. It is also called a Type-I error.

n = 100	Actual: No	Actual: Yes	
Predicted: No	TN: 65	FP: 3	68
Predicted: Yes	FN: 8	TP: 24	32
	73	27	

### Need for Confusion Matrix in Machine learning?

It evaluates the performance of the classification models, when they make predictions on test data, and tells how good our classification model is.

It not only tells the error made by the classifiers but also the type of errors such as it is either type-I or type-II error.

With the help of the confusion matrix, we can calculate the different parameters for the model, such as accuracy, precision, etc.

Example: We can understand the confusion matrix using an example.

Suppose we are trying to create a model that can predict the result for the disease that is either a person has that disease or not. So, the confusion matrix for this is given as: Confusion Matrix in Machine Learning

From the above example, we can conclude that:

The table is given for the two-class classifier, which has two predictions "Yes" and "NO." Here, Yes denotes that patient has the disease, and No denotes that patient does not have that disease.

The classifier has made a total of 100 predictions. Out of 100 predictions, 89 are true predictions, and 11 are incorrect predictions.

The model has given prediction "yes" for 32 times, and "No" for 68 times. Whereas the actual "Yes" was 27, and actual "No" was 73 times.

## 7 REGRESSION

Regression models are used to predict a continuous value. Predicting prices of a house given the features of house like size, price etc is one of the common examples of Regression. It is a supervised technique. A detailed explanation on types of Machine Learning and some important concepts is given in my previous article. Types of Regression

1. Simple Linear Regression
2. Polynomial Regression
3. Support Vector Regression
4. Decision Tree Regression
5. Random Forest Regression

### a Simple linear regression

This is one of the most common and interesting type of Regression technique. Here we predict a target variable Y based on the input variable X. A linear relationship should exist between target variable and predictor and so comes the name Linear Regression.

#### b Polynomial regression

In polynomial regression, we transform the original features into polynomial features of a given degree and then apply Linear Regression on it

#### c Support vector regression

In SVR, we identify a hyperplane with maximum margin such that the maximum number of data points are within that margin. SVRs are almost similar to the SVM classification algorithm. We will discuss the SVM algorithm in detail in my next article.

Instead of minimizing the error rate as in simple linear regression, we try to fit the error within a certain threshold. Our objective in SVR is to basically consider the points that are within the margin. Our best fit line is the hyperplane that has the maximum number of points. d

#### Decision trees

Decision trees can be used for classification as well as regression. In decision trees, at each level, we need to identify the splitting attribute. In the case of regression, the ID3 algorithm can be used to identify the splitting node by reducing the standard deviation (in classification information gain is used).

A decision tree is built by partitioning the data into subsets containing instances with similar values (homogenous). Standard deviation is used to calculate the homogeneity of a numerical sample. If the numerical sample is completely homogeneous, its standard deviation is zero.

#### e Random forest

Random forest is an ensemble approach where we take into account the predictions of several decision regression trees. Select K random points

Identify n where n is the number of decision tree regressors to be created.

Repeat steps 1 and 2 to create several regression trees.

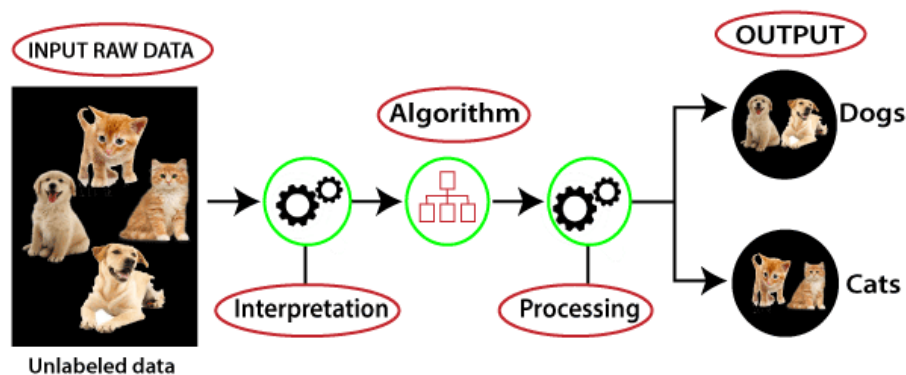
The average of each branch is assigned to the leaf node in each decision tree.

To predict output for a variable, the average of all the predictions of all decision trees are taken into consideration.

Random Forest prevents over fitting (which is common in decision trees) by creating random subsets of the features and building smaller trees using these subsets. **Part II**

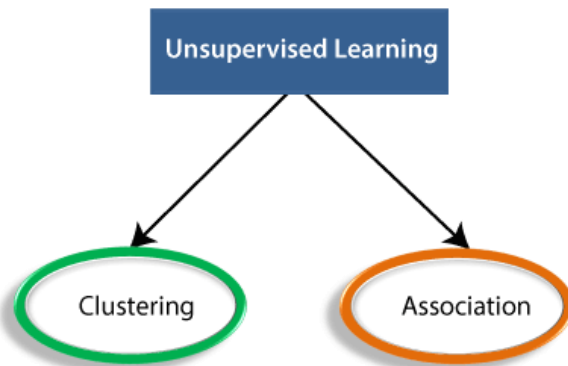
## UNSUPERVISED LEARNING.

As the name suggests, unsupervised learning is a machine learning technique in which models are not supervised using training dataset. Instead, models find the hidden patterns and insights from the given data. Unsupervised learning is a type of machine learning in which models are trained using unlabeled dataset and are allowed to act on that data without any supervision. Unsupervised learning cannot be directly applied to a regression or classification problem because unlike supervised learning, we have the input data but no corresponding output data. The goal of unsupervised learning is to find the underlying structure of dataset, group that data according to similarities, and represent that dataset in a compressed format.



## Types of Unsupervised Learning.





**Clustering:** Clustering is a method of grouping the objects into clusters such that objects with most similarities remain in a group and have less or no similarities with the objects of another group. Cluster analysis finds the commonalities between the data objects and categorizes them as per the presence and absence of those commonalities.

**Association:** An association rule is an unsupervised learning method which is used for finding the relationships between variables in the large database. It determines the set of items that occurs together in the dataset. Association rule makes marketing strategy more effective. Such as people who buy X item (suppose a bread) are also tend to purchase Y (Butter/Jam) item. A typical example of Association rule is Market Basket Analysis.

## 8 CLUSTERING

Clustering is the process of grouping similar entities together. The goal of this unsupervised machine learning technique is to find similarities in the data point and group similar data points together.

Grouping similar entities together help profile the attributes of different groups. In other words, this will give us insight into underlying patterns of different groups. There are many applications of grouping unlabeled data, for example, you can identify different groups/segments of customers and market each group in a different

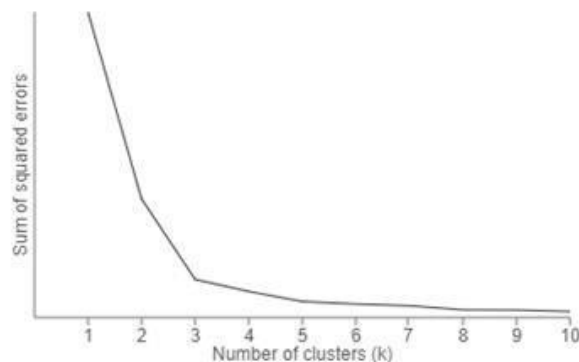
way to maximize the revenue. Another example is grouping documents together which belong to the similar topics etc. Clustering is also used to reduce the dimensionality of the data when you are dealing with a copious number of variables.

There are many algorithms developed to implement this technique but for this post, let's stick to the most popular and widely used algorithms in machine learning.

### 8.1 K-mean Clustering

It starts with K as the input which is how many clusters you want to find.

1. Place K centroids in random locations in your space.
2. Now, using the euclidean distance between data points and centroids, assign each data point to the cluster which is close to it.
3. Recalculate the cluster centers as a mean of data points assigned to it.
4. Repeat 2 and 3 until no further changes occur.
5. Now, you might be thinking that how do I decide the value of K in the first step.



One of the methods is called Elbow method can be used to decide an optimal number of clusters. Here you would run K-mean clustering on a range of K values and plot the percentage of variance explained on the Y-axis and K on X-axis.

In the picture below you would notice that as we add more clusters after 3 it doesn't give much better modeling on the data. The first cluster adds much information, but at some point, the marginal gain will start dropping.

## 8.2 K-mode Clustering

Most of the real world datasets are in categorical form. Let's say, if we are working on analysing the social media, we have categorical data like gender (male or female), profession and so on. So deal with all this categorical data or cluster the categorical variables we use K Modes Clustering. It is widely used algorithm for grouping the categorical data because it is easy to implement and efficiently handles large amount of data. . It defines clusters based on the number of matching categories between data points. (This is in contrast to the more well-known kmeans algorithm, which clusters numerical data based on Euclidean distance.) How is it used???

The k-modes clustering algorithm is an extension of k-means clustering algorithm. The k-means algorithm is the most widely used centre based partitioning clustering algorithm. Huang extends the kmeans clustering algorithm to k-modes clustering algorithm to group the categorical data.

The modifications done in the k-means are -

- (i) using a simple matching dissimilarity measure for categorical objects,
  - (ii) replacing means of clusters by modes, and
  - (iii) using a frequency-based method to update the modes.
- Part III

# PROGRAMS

## 9 SIMPLE LINEAR REGRESSION

### 9.1 Importing the library:

- ```
[1] 1. import numpy as np
    2. import matplotlib.pyplot as plt
```

3. import pandas as pd

## 9.2 Importing the dataset

```
[2] 1. dataset = pd.read_csv('Salary_Data.csv')
    2. X = dataset.iloc[:, :-1].values
    3. Y = dataset.iloc[:, :-1].values
```

## 9.3 Splitting the dataset into the Training set and Test set

```
[3] 1. from sklearn.model_selection import train_test_split
    2. X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=1/3,random_state= 0)
```

## 9.4 Training the Simple Linear Regression model on the Training set

```
[4] 1. from sklearn.linear_model import LinearRegression
    2. regressor = LinearRegression()
    3. regressor.fit(X_train, y_train)
```

## 9.5 Predicting the Test set result. [5] 1.

```
y_pred = regressor.predict(X_test)
```

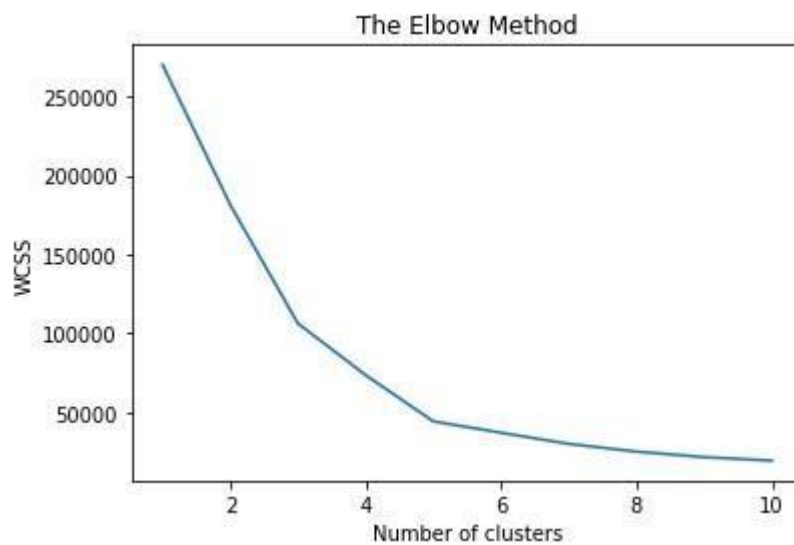
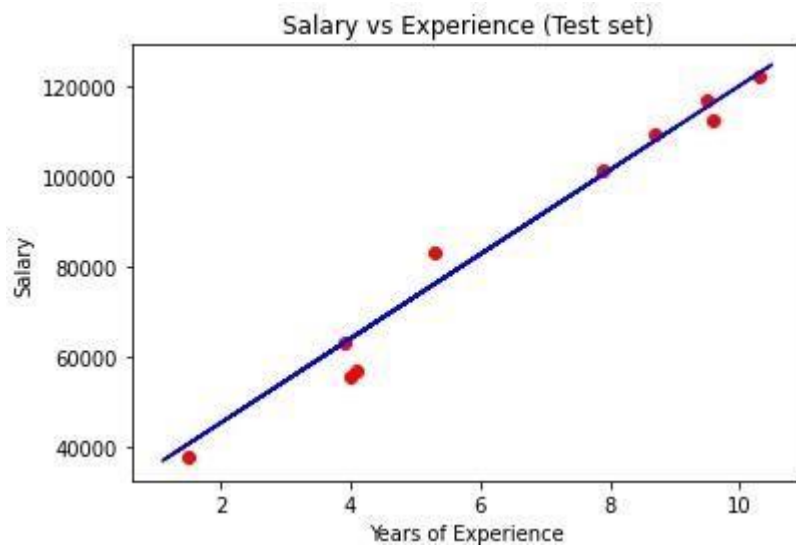
## 9.6 Visualising the Training set result

```
[6] 1. plt.scatter(X_train, y_train,color = 'red' )
    2. plt.plot(X_train,regressor.predict(X_train) , color = 'blue')
    3. plt.title('Salary vs Experience (Training Set)' )
    4. plt.xlabel('Years of Experience' )
    5. plt.ylabel('Salary')
    6. plt.show()
```



### 9.7 Visualising the Testing set result.

- ```
[7] 1. plt.scatter(X_test, y_test, color = 'red')
    2. plt.plot(X_train, regressor.predict(X_train), color = 'blue')
    3. plt.title('Salary vs Experience (Test set)')
    4. plt.xlabel('Years of Experience')
    5. plt.ylabel('Salary')
    6. plt.show()
```



## 10 K- MEANS CLUSTERING.

### 10.1 Importing the libraries

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

### 10.2 Importing the dataset

```
dataset = pd.read_csv('Mall_Customers.csv')
X = dataset.iloc[:, [3, 4]].values
```

### 10.3 Using the elbow method to find the optimal number of clusters

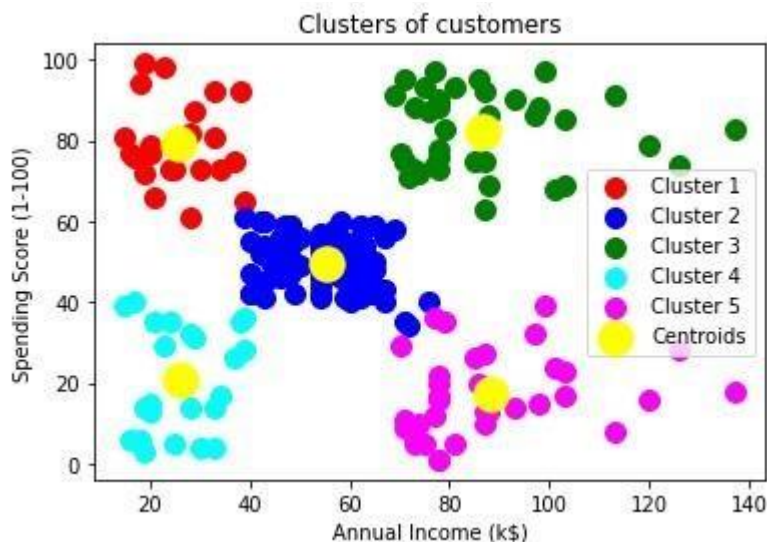
```
from sklearn.cluster import KMeans wcss = [] for i in range(1, 11):  
    kmeans = KMeans(n_clusters = i, init = 'k-means++', random_state = 42)  
    kmeans.fit(X) wcss.append(kmeans.inertia_)  
plt.plot(range(1, 11), wcss) plt.title('The Elbow  
Method') plt.xlabel('Number of clusters')  
plt.ylabel('WCSS') plt.show()
```

### 10.4 Training the K-Means model on the dataset kmeans =

```
KMeans(n_clusters = 5, init = 'k-means++', random_state = 42) y_kmeans =  
kmeans.fit_predict(X)
```

### 10.5 Visualising the clusters

```
plt.scatter(X[y_kmeans == 0, 0], X[y_kmeans == 0, 1], s = 100, c = 'red', label = 'Cluster 1')  
plt.scatter(X[y_kmeans == 1, 0], X[y_kmeans == 1, 1], s = 100, c = 'blue', label = 'Cluster 2')  
plt.scatter(X[y_kmeans == 2, 0], X[y_kmeans == 2, 1], s = 100, c = 'green', label = 'Cluster 3')  
plt.scatter(X[y_kmeans == 3, 0], X[y_kmeans == 3, 1], s = 100, c = 'cyan', label = 'Cluster 4')  
plt.scatter(X[y_kmeans == 4, 0], X[y_kmeans == 4, 1], s = 100, c = 'magenta', label = 'Cluster 5')  
plt.scatter(kmeans.cluster_centers_[0, 0], kmeans.cluster_centers_[0, 1], s = 300, c = 'yellow', label = 'Centroid 1')  
plt.scatter(kmeans.cluster_centers_[1, 0], kmeans.cluster_centers_[1, 1], s = 300, c = 'yellow', label = 'Centroid 2')  
plt.scatter(kmeans.cluster_centers_[2, 0], kmeans.cluster_centers_[2, 1], s = 300, c = 'yellow', label = 'Centroid 3')  
plt.scatter(kmeans.cluster_centers_[3, 0], kmeans.cluster_centers_[3, 1], s = 300, c = 'yellow', label = 'Centroid 4')  
plt.scatter(kmeans.cluster_centers_[4, 0], kmeans.cluster_centers_[4, 1], s = 300, c = 'yellow', label = 'Centroid 5')  
plt.title('Clusters of customers') plt.xlabel('Annual Income (k$)') plt.ylabel('Spending Score (1-100)') plt.legend() plt.show()
```



## 11 Polynomial Regression

### 11.1 the library: import numpy as np import

```
matplotlib.pyplot as plt
```

```
import pandas as pd
```

## 11.2 Importing the dataset

```
dataset = pd.read_csv('position_Salaries.csv')  
X = dataset.iloc[:, 1:-1].values  
Y = dataset.iloc[:, -1].values
```

## 11.3 Training the Linear Regression model on the whole dataset

```
from sklearn.model_selection import LinearRegression  
lin_reg = LinearRegression()  
lin_reg.fit(X, y)
```

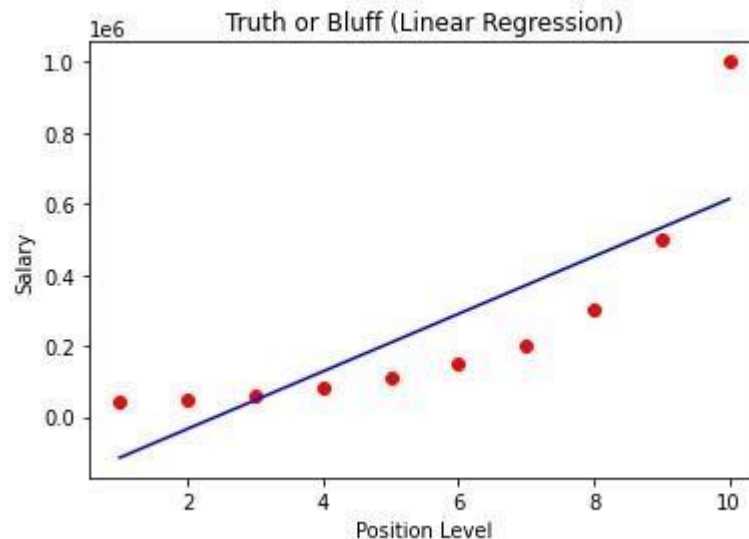
`LinearRegression(copy_X=True, t_intercept=True, n_jobs=None, normalize=False)`

## 11.4 Training the Polynomial Regression model on the whole dataset

```
from sklearn.preprocessing import PolynomialFeatures  
poly_reg = PolynomialFeatures(degree = 4)  
lin_reg_2 = LinearRegression()  
lin_reg_2.fit(X_poly, y)
```

## 11.5 Visualising the Linear Regression results

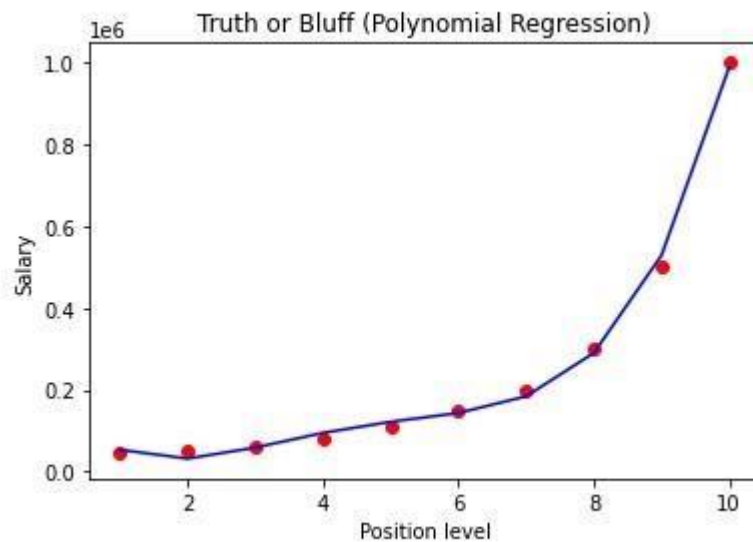
```
plt.scatter(X, y, color = 'red')  
plt.plot(X, lin_reg.predict(X), color = 'blue')  
plt.title('Truth or Bluff (Linear Regression)')  
plt.xlabel('Position Level')  
plt.ylabel('Salary')  
plt.show()
```



11.6 Visualising the Polynomial Regression results

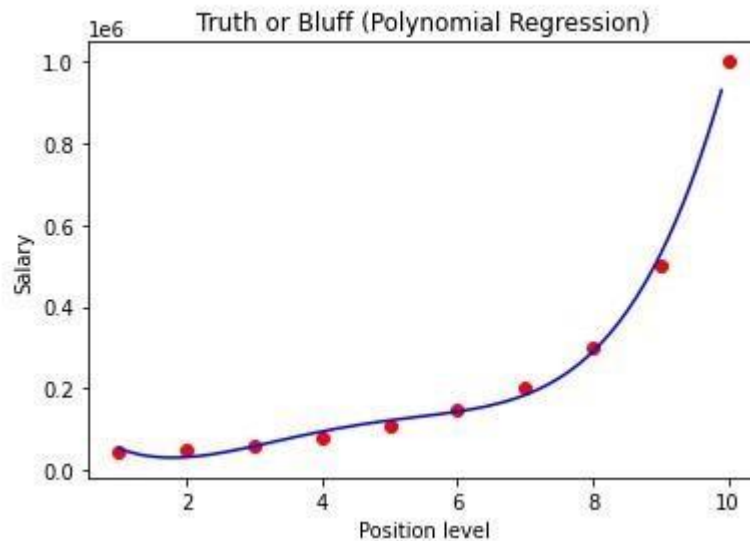
```
plt.scatter(X, y, color = 'red')  
plt.plot(X, lin_reg_2.predict(poly_reg.fit_transform(X)), color = 'blue')  
plt.title('Truth or Bluff (Polynomial Regression)')  
plt.xlabel('Position level')  
plt.ylabel('Salary')  
plt.show()
```





### 11.7 Visualising the Polynomial Regression results (for higher resolution and smoother curve)

```
X_grid = np.arange(min(X), max(X), 0.1)
X_grid = X_grid.reshape((len(X_grid), 1))
plt.scatter(X, y, color = 'red')
plt.plot(X_grid, lin_reg_2.predict(poly_reg.fit_transform(X_grid)), color = 'blue')
plt.title('Truth or Bluff (Polynomial Regression)')
plt.xlabel('Position level')
plt.ylabel('Salary')
plt.show()
```



## 12 Support Vector Regression (SVR)

### 12.1 Importing the library:

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

### 12.2 Importing the dataset dataset =

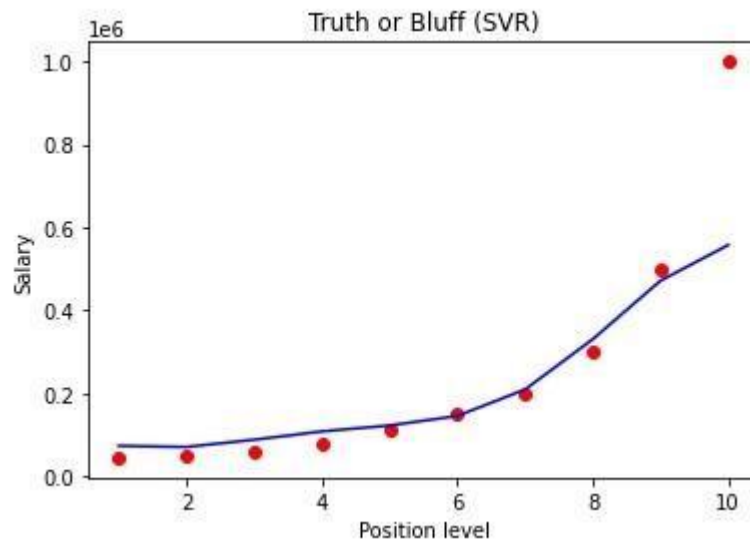
```
pd.read_csv('position_Salaries.csv')
X = dataset.iloc[:, 1:-1].values
Y = dataset.iloc[:, -1].values
```

### 12.3 Feature Scaling

```
from sklearn.preprocessing import StandardScaler
sc_X = StandardScaler()
sc_y = StandardScaler()
X = sc_X.fit_transform(X)
y = sc_y.fit_transform(y)
```

### 12.4 Training the SVR model on the whole dataset

```
from sklearn.svm import SVR
regressor = SVR(kernel='rbf')
regressor.fit(X, y)
```



### 12.5 Predicting a new result

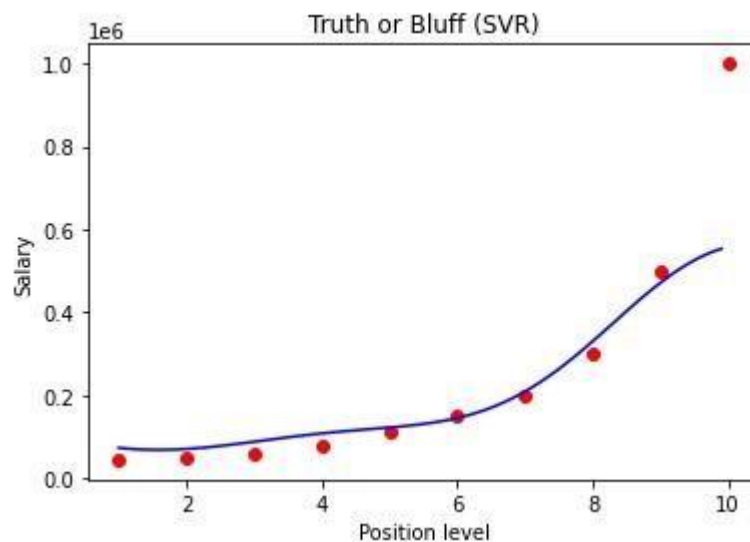
```
sc_y.inverse_transform(regressor.predict(sc_X.transform([[6.5]])))
```

## 12.6 Visualising the SVR results

```
plt.scatter(sc_X.inverse_transform(X), sc_y.inverse_transform(y), color = 'red') plt.plot(sc_X.inverse_transform(X),  
sc_y.inverse_transform(regressor.predict(X)), color = 'b' plt.title('Truth or Bluff (SVR)') plt.xlabel('Position level')  
plt.ylabel('Salary') plt.show()
```

## 12.7 Visualising the SVR results (for higher resolution and smoother curve)

```
X_grid = np.arange(min(sc_X.inverse_transform(X)), max(sc_X.inverse_transform(X)), 0.1) X_gr plt.plot(X_grid,  
sc_y.inverse_transform(regressor.predict(sc_X.transform(X_grid)))), color = plt.title('Truth or Bluff (SVR)')  
plt.xlabel('Position level') plt.ylabel('Salary') plt.show()
```



## 13 Hierarchical Clustering.

### 13.1 Importing the libraries

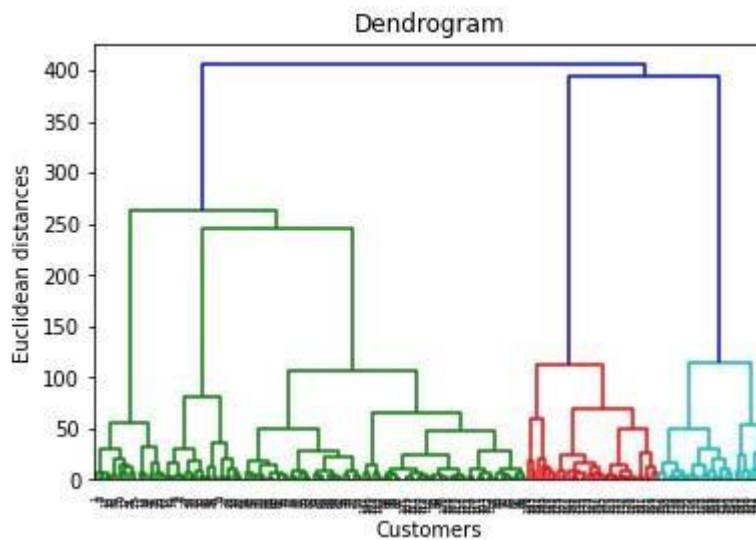
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

### 13.2 Importing the dataset

```
dataset = pd.read_csv('Mall_Customers.csv')
X = dataset.iloc[:, [3, 4]].values
```

### 13.3 Using the dendrogram to find the optimal number of clusters

```
import scipy.cluster.hierarchy as sch
dendrogram = sch.dendrogram(sch.linkage(X, method = 'ward'))
plt.title('Dendrogram')
plt.xlabel('Customers')
plt.ylabel('Euclidean distances')
plt.show()
```



### 13.4 Training the Hierarchical Clustering model on the dataset from

```
from sklearn.cluster import AgglomerativeClustering
hc = AgglomerativeClustering(n_clusters = 5, affinity = 'euclidean', linkage = 'ward')
y_hc = hc.fit_predict(X)
```

### 13.5 Visualising the clusters

```
plt.scatter(X[y_hc == 0, 0], X[y_hc == 0, 1], s = 100, c = 'red', label = 'Cluster 1')
plt.scatter(X[y_hc == 1, 0], X[y_hc == 1, 1], s = 100, c = 'blue', label = 'Cluster 2')
plt.scatter(X[y_hc == 2, 0], X[y_hc == 2, 1], s = 100, c = 'green', label = 'Cluster 3')
```

```
'Cluster 3') plt.scatter(X[y_hc == 3, 0], X[y_hc == 3, 1], s = 100, c = 'cyan', label = 'Cluster 4') plt.scatter(X[y_hc == 4, 0],  
X[y_hc == 4, 1], s = 100, c = 'magenta', label = 'Cluster 5') plt.title('Clusters of customers') plt.xlabel('Annual Income  
(k$)') plt.ylabel('Spending Score (1-100)') plt.legend() plt.show()
```

