Practical No. 3

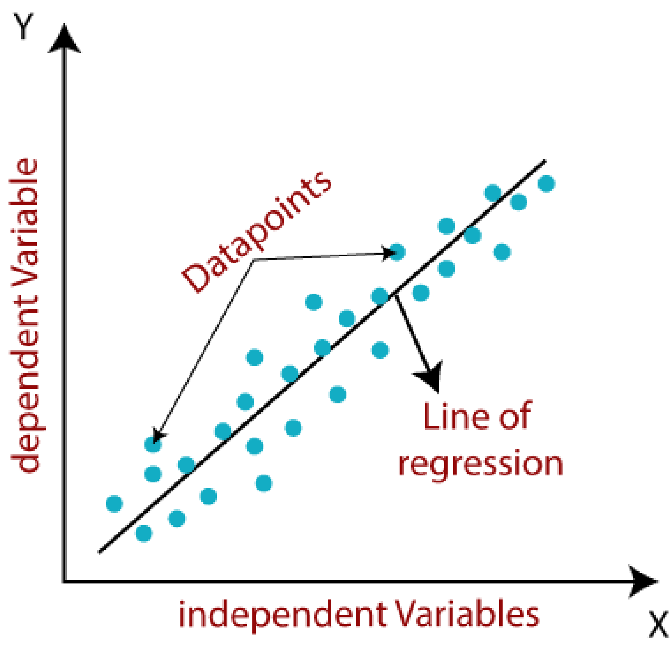
Title: Implementation of Linear Regression, Logistic regression, KNN- classification.

Aim: Understanding basics of Linear Regression, Logistic regression and KNN-classification.

Introduction:

Linear Regression

Linear Regression is a machine learning algorithm based on **supervised learning**. It performs a **regression task**. Regression models a target prediction value based on independent variables. It is mostly used for finding out the relationship between variables and forecasting. Different regression models differ based on – the kind of relationship between dependent and independent variables they are considering, and the number of independent variables getting used.



Linear regression algorithm shows a linear relationship between a dependent (y) and one or more independent (y) variables, hence called linear regression. Since linear regression shows the linear relationship, which means it finds how the value of the dependent variable is changing according to the value of the independent variable.

The linear regression model provides a sloped straight line representing the relationship between the variables.

Mathematically, we can represent a linear regression as:

y= mx + b

Here,

Y= Dependent Variable (Target Variable)

X= Independent Variable (predictor Variable)

b= intercept of the line (Gives an additional degree of freedom)

m= Linear regression coefficient (scale factor to each input value).

The values for x and y variables are training datasets for Linear Regression model representation.

Finding the best fit line:

When working with linear regression, our main goal is to find the best fit line that means the error between predicted values and actual values should be minimized. The best fit line will have the least error.

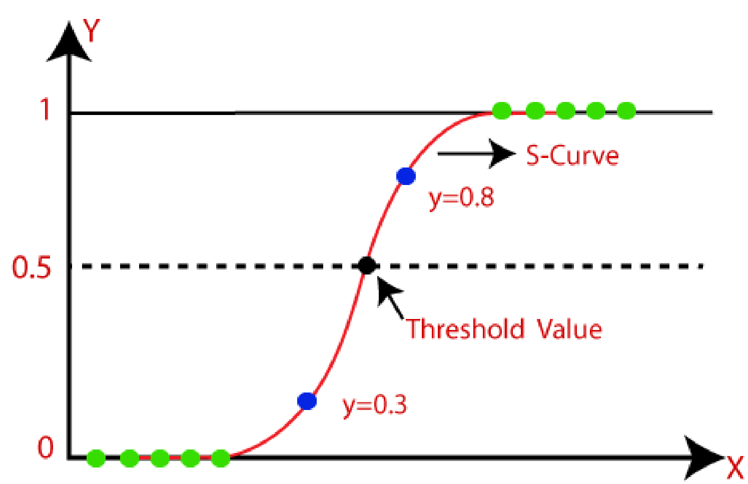
The different values for weights or the coefficient of lines (a_0, a_1) gives a different line of regression, so we need to calculate the best values for a_0 and a_1 to find the best fit line, so to calculate this we use cost function.

Logistic Regression in Machine Learning

- Logistic regression is one of the most popular Machine Learning algorithms, which comes under the Supervised Learning technique. It is used for predicting the categorical dependent variable using a given set of independent variables.
- Logistic regression predicts the output of a categorical dependent variable. Therefore the outcome must be a categorical or discrete value. It can be either Yes or

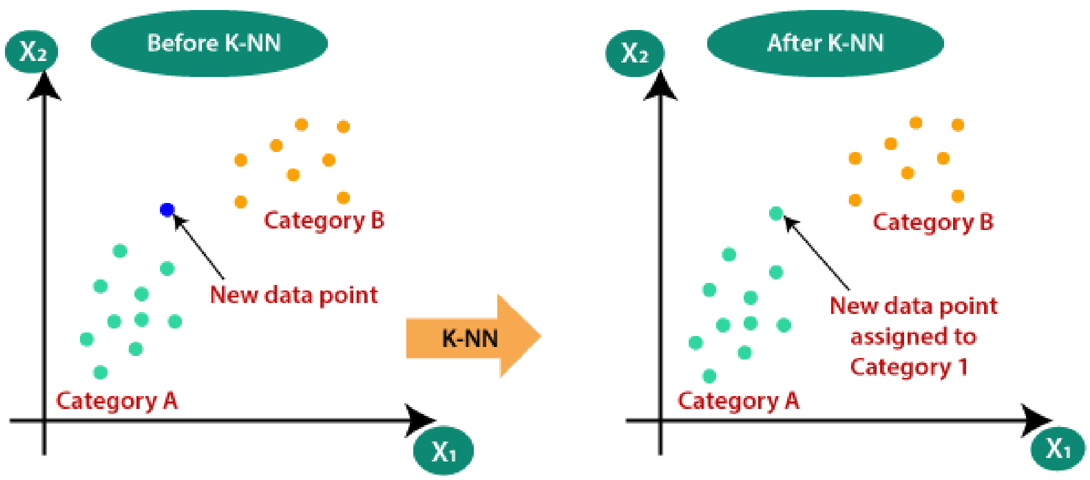
No, 0 or 1, true or False, etc. but instead of giving the exact value as 0 and 1, **it gives the probabilistic values which lie between 0 and 1**.

- Logistic Regression is much similar to the Linear Regression except that how they are used. Linear Regression is used for solving Regression problems, whereas **Logistic regression is used for solving the classification problems**.
- In Logistic regression, instead of fitting a regression line, we fit an "S" shaped logistic function, which predicts two maximum values (0 or 1).
- The curve from the logistic function indicates the likelihood of something such as whether the cells are cancerous or not, a mouse is obese or not based on its weight, etc.
- Logistic Regression is a significant machine learning algorithm because it has the ability to provide probabilities and classify new data using continuous and discrete datasets.
- Logistic Regression can be used to classify the observations using different types of data and can easily determine the most effective variables used for the classification. The below image is showing the logistic function:



K-Nearest Neighbor(KNN) Algorithm for Machine Learning

Suppose there are two categories, i.e., Category A and Category B, and we have a new data point x1, so this data point will lie in which of these categories. To solve this type of problem, we need a K-NN algorithm. With the help of K-NN, we can easily identify the category or class of a particular dataset. Consider the below diagram:



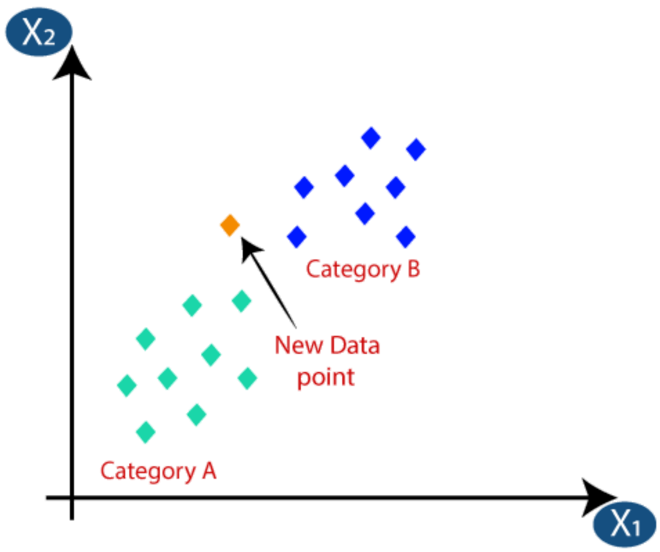
How does K-NN work?

The K-NN working can be explained on the basis of the below algorithm:

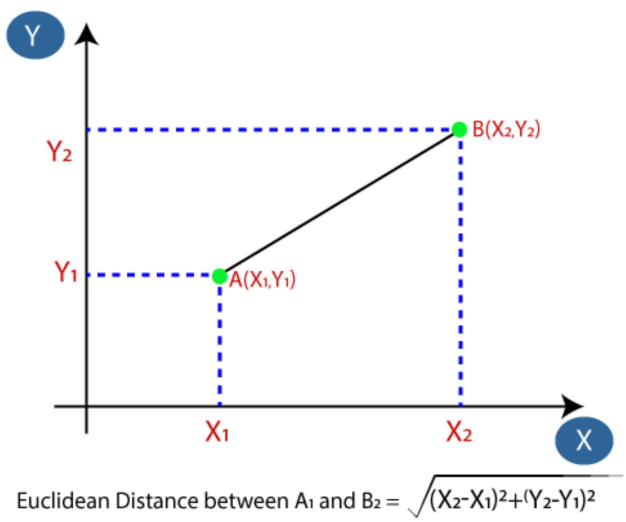
- **Step-1:** Select the number K of the neighbors
- **Step-2:** Calculate the Euclidean distance of **K number of neighbors**
- **Step-3:** Take the K nearest neighbors as per the calculated Euclidean distance.
- **Step-4:** Among these k neighbors, count the number of the data points in each category.
- **Step-5:** Assign the new data points to that category for which the number of the neighbor is maximum.

- **Step-6:** Our model is ready.

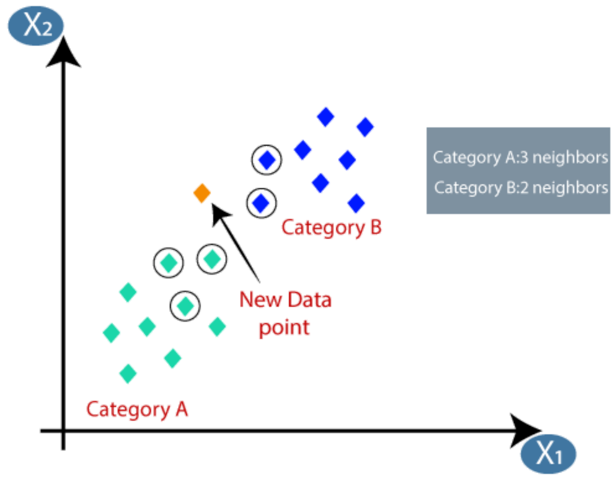
Suppose we have a new data point and we need to put it in the required category. Consider the below image:



- Firstly, we will choose the number of neighbors, so we will choose the k=5.
- Next, we will calculate the **Euclidean distance** between the data points. The Euclidean distance is the distance between two points, which we have already studied in geometry. It can be calculated as:



- By calculating the Euclidean distance we got the nearest neighbors, as three nearest neighbors in category A and two nearest neighbors in category B. Consider the below image:



- As we can see the 3 nearest neighbors are from category A, hence this new data point must belong to category A.

Exercise -

1. **Predict Canada's per capita income in 2020. There is an exercise folder here on github at the same level as this notebook, download that and you will find the canada_per_capita_income.csv file. Using this build a regression model and predict the per capita income of canadian citizens in year 2020**

Program:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression

# Read the data
url = 'https://raw.githubusercontent.com/codebasics/py/master/ML/1_linear_reg/Exercise/canada_per_capita_income.csv'
data = pd.read_csv(url)
```

```
# Prepare the data
X = data["year"].values.reshape(-1, 1)  # Features: Year
y = data["per capita income (US$)"]     # Target variable: Per capita income

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

# Build the regression model
model = LinearRegression()
model.fit(X, y)

# Make predictions for the year 2020
year_2020 = [[2020]]
predicted_income_2020 = model.predict(year_2020)
print("Predicted per capita income for Canada in 2020:", predicted_income_2020[0])
```

Output:

```
Predicted per capita income for Canada in 2020: 41288.69409441762
```

2. **Download employee retention dataset from here: https://www.kaggle.com/giripujar/hr-analytics.**

   Now do some exploratory data analysis to figure out which variables have direct and clear impact on employee retention (i.e. whether they leave the company or continue to work)

   Plot bar charts showing impact of employee salaries on retention

   Plot bar charts showing correlation between department and employee retention

   Now build logistic regression model using variables that were narrowed down in step 1

   Measure the accuracy of the model

   Program:

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report

# Step 1: Data Loading and Exploration
# Load the dataset
url =
'https://raw.githubusercontent.com/OnkarMalawade/MCA-SEM-II-AIML/main/HR_comma_sep.csv'
hr_data = pd.read_csv(url)

# Explore the dataset
print(hr_data.head())
print(hr_data.info())
print(hr_data.describe())
```

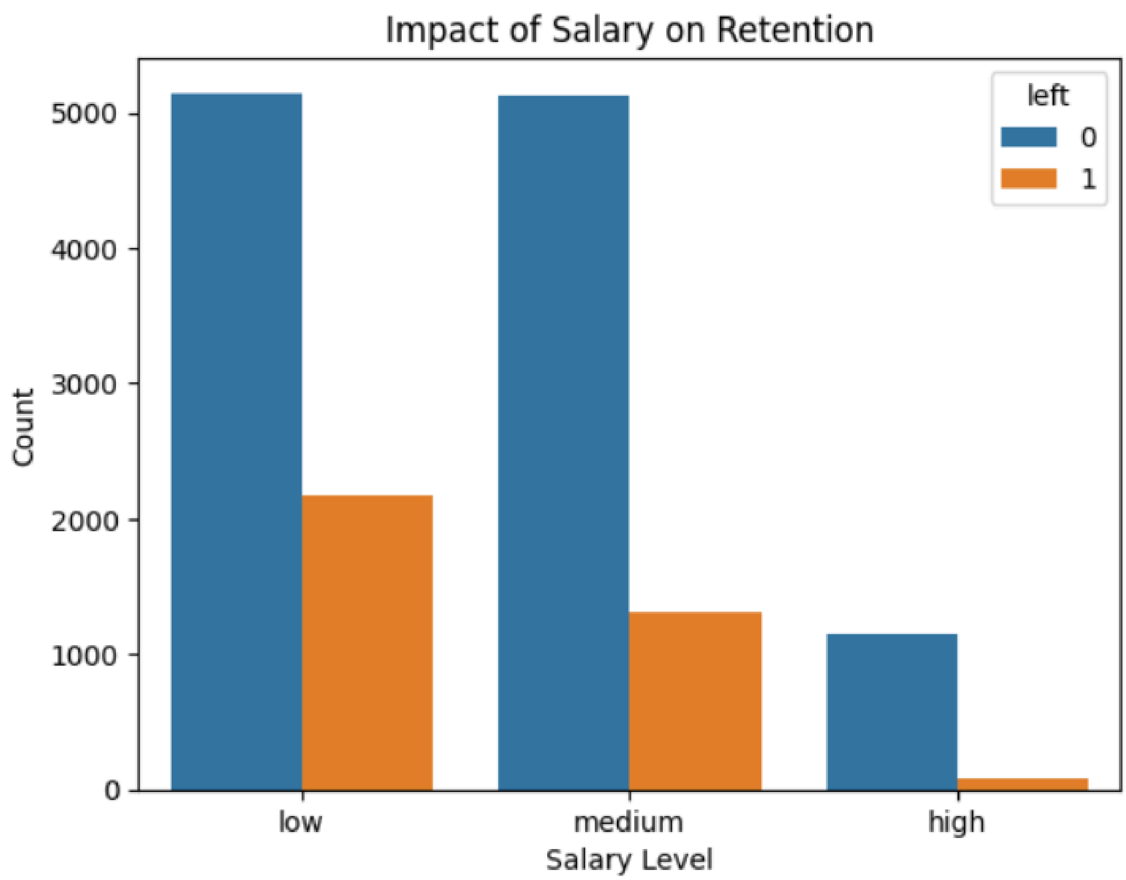
```
# Step 2: Identify Impactful Variables
# Conduct exploratory data analysis

# Step 3: Plot Bar Charts for Salary Impact
sns.countplot(x='salary', hue='left', data=hr_data)
plt.title('Impact of Salary on Retention')
plt.xlabel('Salary Level')
plt.ylabel('Count')
plt.show()

# Step 4: Plot Bar Charts for Department Impact
sns.countplot(x='Department', hue='left', data=hr_data)
plt.title('Impact of Department on Retention')
plt.xlabel('Department')
plt.ylabel('Count')
plt.xticks(rotation=45)
plt.show()

# Step 5: Build Logistic Regression Model
# Define predictors and target variable
X = hr_data[['satisfaction_level', 'last_evaluation', 'number_project',
'average_montly_hours', 'time_spend_company', 'Work_accident',
'promotion_last_5years', 'salary', 'Department']]
y = hr_data['left']

# Convert categorical variables to dummy variables
X = pd.get_dummies(X, drop_first=True)
```

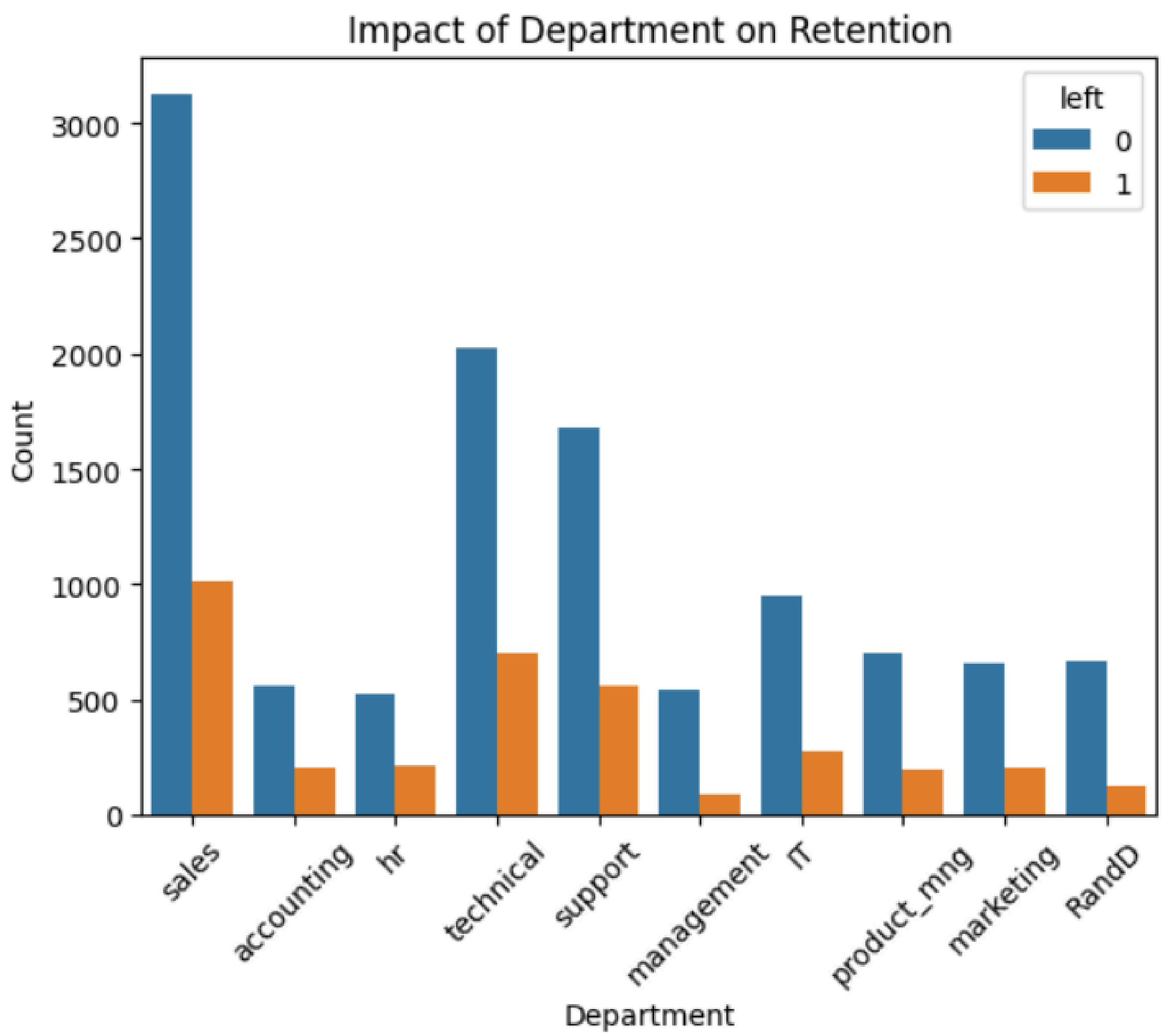
```
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Build the logistic regression model
model = LogisticRegression()
model.fit(X_train, y_train)

# Step 6: Model Evaluation
# Make predictions
y_pred = model.predict(X_test)

# Calculate accuracy
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)

# Print classification report
print(classification_report(y_test, y_pred))
```

Output:

```
   satisfaction_level  last_evaluation  number_project  average_montly_hours  \
0                0.38             0.53               2                   157
1                0.80             0.86               5                   262
2                0.11             0.88               7                   272
3                0.72             0.87               5                   223
4                0.37             0.52               2                   159

   time_spend_company  Work_accident  left  promotion_last_5years Department  \
0                   3              0     1                      0      sales
1                   6              0     1                      0      sales
2                   4              0     1                      0      sales
3                   5              0     1                      0      sales
4                   3              0     1                      0      sales

   salary
0     low
1  medium
2  medium
3     low
4     low
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14999 entries, 0 to 14998
Data columns (total 10 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   satisfaction_level     14999 non-null  float64
 1   last_evaluation        14999 non-null  float64
 2   number_project         14999 non-null  int64
 3   average_montly_hours   14999 non-null  int64
 4   time_spend_company     14999 non-null  int64
 5   Work_accident          14999 non-null  int64
 6   left                   14999 non-null  int64
 7   promotion_last_5years  14999 non-null  int64
 8   Department             14999 non-null  object
 9   salary                 14999 non-null  object
dtypes: float64(2), int64(6), object(2)
memory usage: 1.1+ MB
None
```

```
       satisfaction_level  last_evaluation  number_project  \
count        14999.000000     14999.000000    14999.000000
mean             0.612834         0.716102        3.803054
std              0.248631         0.171169        1.232592
min              0.090000         0.360000        2.000000
25%              0.440000         0.560000        3.000000
50%              0.640000         0.720000        4.000000
75%              0.820000         0.870000        5.000000
max              1.000000         1.000000        7.000000

       average_montly_hours  time_spend_company  Work_accident          left  \
count          14999.000000        14999.000000   14999.000000  14999.000000
mean             201.050337            3.498233       0.144610      0.238083
std               49.943099            1.460136       0.351719      0.425924
min               96.000000            2.000000       0.000000      0.000000
25%              156.000000            3.000000       0.000000      0.000000
50%              200.000000            3.000000       0.000000      0.000000
75%              245.000000            4.000000       0.000000      0.000000
max              310.000000           10.000000       1.000000      1.000000

       promotion_last_5years
count           14999.000000
mean                0.021268
std                 0.144281
min                 0.000000
25%                 0.000000
50%                 0.000000
75%                 0.000000
max                 1.000000
```


Impact of Salary on Retention
left
0
1
5000
4000
3000
2000
1000
0
Count
low
medium
high
Salary Level




```
Accuracy: 0.7966666666666666
              precision    recall  f1-score   support

           0       0.83      0.92      0.87      2294
           1       0.61      0.39      0.47       706

    accuracy                           0.80      3000
   macro avg       0.72      0.66      0.67      3000
weighted avg       0.78      0.80      0.78      3000
```

3. **Using K nearest neighbors classification predict type of flower given 'sepal_length', 'sepal_width', 'petal_length', 'petal_width' = 4.8,3.0,1.5,0.3**

Program:

```
In [3]:
from sklearn import linear_model
from matplotlib import pyplot as plt
import pandas as pd
import numpy as np
```

```
In [6]:
import pandas as pd
url = 'https://gist.githubusercontent.com/curran/a08a1080b88344b0c8a7/raw/0e7a9b0a5d22642a06d3d5b9bcbad9890c8ee534/iris.csv
iris = pd.read_csv(url)
iris
```

```
In [ ]:
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(iris.drop('species',axis=1),iris['species'],train_size=0.9)
X_test
```

Out[ ]:

	sepal_length	sepal_width	petal_length	petal_width
31	5.4	3.4	1.5	0.4
92	5.8	2.6	4.0	1.2
50	7.0	3.2	4.7	1.4
84	5.4	3.0	4.5	1.5
29	4.7	3.2	1.6	0.2
90	5.5	2.6	4.4	1.2
118	7.7	2.6	6.9	2.3
77	6.7	3.0	5.0	1.7
28	5.2	3.4	1.4	0.2
136	6.3	3.4	5.6	2.4
14	5.8	4.0	1.2	0.2
80	5.5	2.4	3.8	1.1
45	4.8	3.0	1.4	0.3
81	5.5	2.4	3.7	1.0
44	5.1	3.8	1.9	0.4

In [ ]:
```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier()
knn.fit(X_train,y_train)
```

Out[ ]:
```
KNeighborsClassifier()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [12]:
```
knn.predict([[4.8,3.0,1.5,0.3]])
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but KNeighbors
Classifier was fitted with feature names
  warnings.warn(
```

Out[12]:
```
array(['setosa'], dtype=object)
```