1.Linear regression

Aim:

To Train the data using Linear regression and visualize the data

Algorithm:

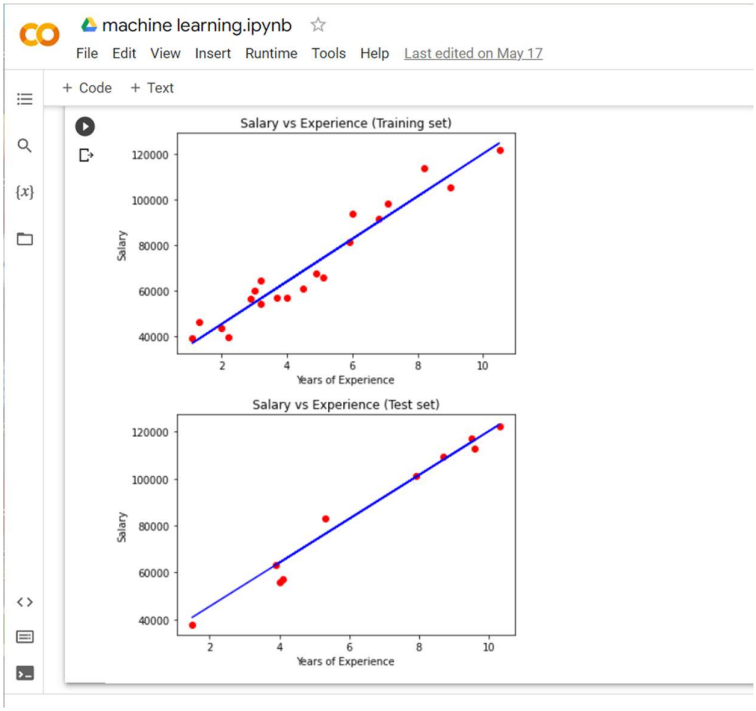
1. Import the libraries what we need i.e. (**Numpy,pandas,matplotlib.pyplot**)
2. Importing the dataset as csv file
3. splitting the dataset into training and test sets using (**sklearn.model_selection)** module
4. From (**sklearn.model_selection)** we import (**LinearRegression Package**) to fit training set
5. Predicting the test set
6. Visualizing the training set results
7. Visualizing the testing set results

Code:

```
# Simple Linear Regression

#import the Libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

#importing the dataset
dataset = pd.read_csv('Salary_Data.csv')
X = dataset.iloc[:,:-1].values
y = dataset.iloc[:,1].values

#splitting the dataset into training and test set
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 1
/3, random_state=0)

#Fitting simple linear regression to the training set
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X_train, y_train)

# Predicting the test set
y_pred = regressor.predict(X_test)

# Visualizing the training set results
plt.scatter(X_train, y_train, color='red')
plt.plot(X_train, regressor.predict(X_train), color='blue')
plt.title('Salary vs Experience (Training set)')
```

```
plt.xlabel('Years of Experience')
plt.ylabel('Salary')
plt.show()

# Visualing the testing set results
plt.scatter(X_test, y_test, color='red')
plt.plot(X_test, regressor.predict(X_test), color='blue')
plt.title('Salary vs Experience (Test set)')
plt.xlabel('Years of Experience')
plt.ylabel('Salary')
plt.show()
```

Output:



Result:
Training the data using **Linear regression** Was implemented successfully and output is verified...

2.Logistic regression

Aim:

To Train the data using Logistic regression and visualize the data

Algorithm:

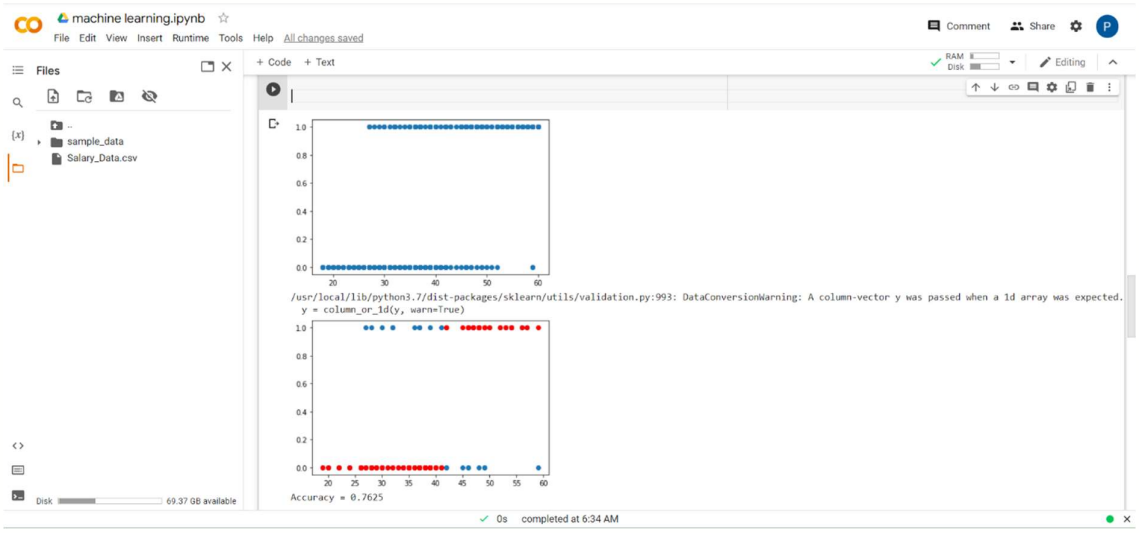
1. Import the libraries what we need i.e. (**Numpy,pandas,matplotlib.pyplot,math**)
2. Importing the dataset as csv file
3. Visualizing the dataset
4. splitting the dataset into training and test sets using (**sklearn.model_selection)** module
5. From (**sklearn.model_selection)** we import (**Logisticregression Package**) to fit training set
6. Create an instance and fit the model
7. Predicting the test set
8. Finding Accuracy

Code:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from math import exp
data = pd.read_csv("https://raw.githubusercontent.com/shivang98/Social-
Network-ads-Boost/master/Social_Network_Ads.csv")
data.head()
# Visualizing the dataset
plt.scatter(data['Age'], data['Purchased'])
plt.show()
# Divide the data to training set and test set
X_train, X_test, y_train, y_test = train_test_split(data['Age'], data['
Purchased'], test_size=0.20)
# Making predictions using scikit learn
from sklearn.linear_model import LogisticRegression

# Create an instance and fit the model
lr_model = LogisticRegression()
lr_model.fit(X_train.values.reshape(-1, 1), y_train.values.reshape(-
1, 1))

# Making predictions
y_pred_sk = lr_model.predict(X_test.values.reshape(-1, 1))
plt.clf()
plt.scatter(X_test, y_test)
plt.scatter(X_test, y_pred_sk, c="red")
plt.show()
```

```
# Accuracy
print(f"Accuracy = {lr_model.score(X_test.values.reshape(-
1, 1), y_test.values.reshape(-1, 1))}")
```

Output :



Result:

Training the data using **Logistic regression** Was implemented successfully and output is verified...

3.Decision Tree (ID3 algorithm)

Aim:

To Train the data using Decision Tree (ID3 algorithm)

Algorithm:

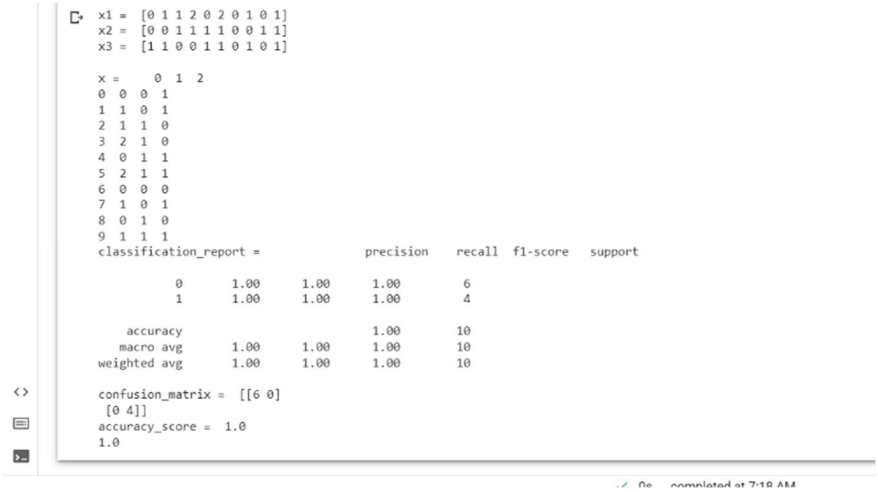
1. Import the libraries what we need i.e. (**Numpy,pandas,matplotlib.pyplot,math**)
2. Creating the Data
3. Data pre-processing is required for encoding the input string data
4. Creating a model and predicting the values
5. Finding Classification report of the created model
6. Finding Confusion matrix of the give model
7. Finding accuracy of the given model

Code:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import preprocessing
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report,confusion_matrix

# creating the data
Running = ['Fast','Medium','Medium','Slow','Fast','Slow','Fast','Medium
','Fast','Medium']
Diving = ['Long','Long','Short','Short','Short','Short','Long','Long','
Short','Short']
Fighting = ['Furious','Furious','Calm','Calm','Furious','Furious','Calm
','Furious','Calm','Furious']
Action_Hero = ['Yes','Yes','No','No','Yes','No','No','Yes','No','No',]

#Data preprocessing
labeling = preprocessing.LabelEncoder()
x1 = labeling.fit_transform(Running)
x2 = labeling.fit_transform(Diving)
x3 = labeling.fit_transform(Fighting)
x = pd.DataFrame(list(zip(x1,x2,x3)))
print('x1 = ',x1)
print('x2 = ',x2)
print('x3 = ',x3)
print()
print('x = ',x)
```

```
y = labeling.fit_transform(Action_Hero)
y

#creating a model and predicting model
model = DecisionTreeClassifier()
model.fit(x,y)
y_pred = model.predict(x)
y_pred

#classification report of the created model
print('classification_report = '+classification_report(y,y_pred))

#confusion matrix of the given model
print('confusion_matrix = ',confusion_matrix(y,y_pred))

#Accuracy score
print('accuracy_score = ',accuracy_score(y,y_pred))

#model score
model.score(x,y)
```

Output:



Result:

Training the data using **Decision Tree (ID3 algorithm)** Was implemented successfully and output is verified...

4.K-Means Clustering

Aim:

To Train the data using **K-Means Clustering** and visualize the data…

Algorithm:

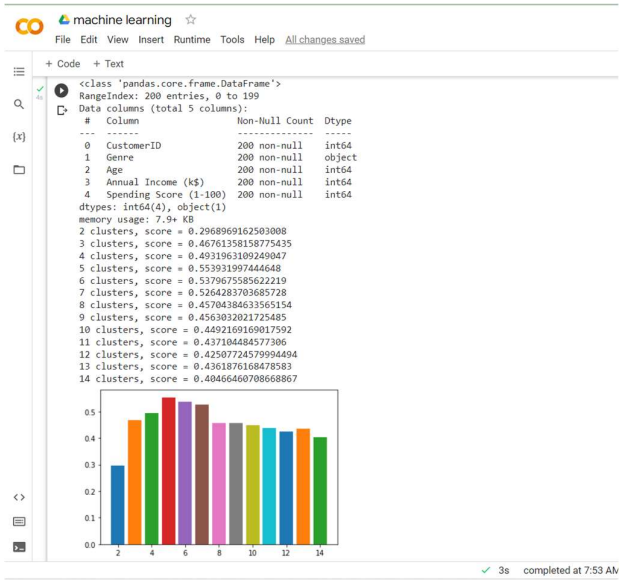
1. Import the libraries what we need i.e. (**Numpy,pandas,matplotlib.pyplot,math**)
2. Reading the data as csv file
3. Allocating required values for features
4. By using silhouette score we can find the model accuracy of particular value
5. Creating a model and predicting the values
6. Identifying the number of values allocated for respective cluster
7. Scatter plotting of input data
8. Finding cluster centres
9. Scatter plotting of clustered data

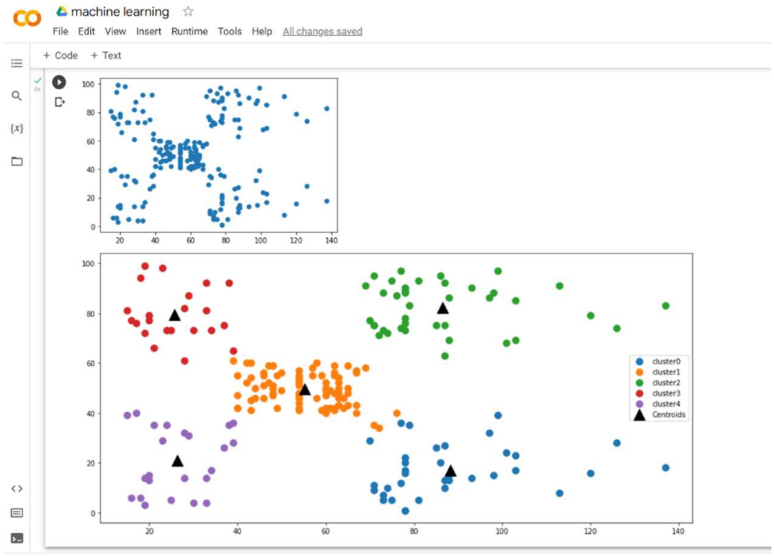
Code:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score

#reading a csv file
df = pd.read_csv('https://raw.githubusercontent.com/arib168/data/main/M
all_Customers.csv')
df
df.info()

#allocating required values
x = df.iloc[:,[3,4]].values
x
np.sqrt(200)

#using silhouette score for finding accuracy of particular k value
k = range(2,15)

for i in k:
  demo_model = KMeans(n_clusters=i,random_state=0)
  demo_model.fit(x)
  y = demo_model.predict(x)
  print(f'{i} clusters, score = {silhouette_score(x,y)}')
  plt.bar(i,silhouette_score(x,y))
plt.show()
```

```
#creating a model

k = 5
model = KMeans(n_clusters=5,random_state=0)
model.fit(x)
y = model.predict(x)
y

#silhouette score of the model
silhouette_score(x,y)

#identifying the no of values allocated for respective cluster
np.unique(y,return_counts=True)

#scatter plot
plt.scatter(df['Annual Income (k$)'],df['Spending Score (1-100)'])
plt.show()

#cluster centers
model.cluster_centers_

#scatter plot
plt.figure(figsize=(15,7))
for i in range(k):
  plt.scatter(x[y==i,0],x[y==i,1],s=90,label = f'cluster{i}')
plt.scatter(model.cluster_centers_[:,0],model.cluster_centers_[:,1],lab
el = 'Centroids',s = 250,marker='^',c='black')
plt.legend()
plt.show()
```

Output:





Result:

Training the data using **K-Means Clustering** Was implemented successfully and output is verified...

5.Bayesian classification

Aim:

To Train the data using **Bayesian classification**

Algorithm:

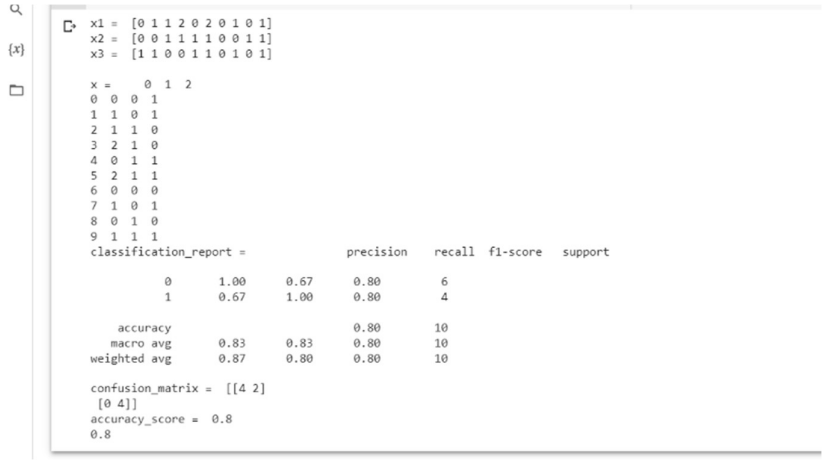
1. Import the libraries what we need i.e. (**Numpy,pandas,matplotlib.pyplot,math**)
2. Creating the data
3. Data pre-processing is required for encoding the input string data
4. Finding classification report of the model
5. Finding confusion matrix of the model
6. Finding the model, the score

Code:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import preprocessing
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report,confusion_matrix

# creating the data
Running = ['Fast','Medium','Medium','Slow','Fast','Slow','Fast','Medium
','Fast','Medium']
Diving = ['Long','Long','Short','Short','Short','Short','Long','Long','
Short','Short']
Fighting = ['Furious','Furious','Calm','Calm','Furious','Furious','Calm
','Furious','Calm','Furious']
Action_Hero = ['Yes','Yes','No','No','Yes','No','No','Yes','No','No',]

#Data preprocessing
labeling = preprocessing.LabelEncoder()
x1 = labeling.fit_transform(Running)
x2 = labeling.fit_transform(Diving)
x3 = labeling.fit_transform(Fighting)
x = pd.DataFrame(list(zip(x1,x2,x3)))
print('x1 = ',x1)
print('x2 = ',x2)
print('x3 = ',x3)
print()
print('x = ',x)

y = labeling.fit_transform(Action_Hero)
y
```

```
#creating a model and predicting model
model = GaussianNB()
model.fit(x,y)
y_pred = model.predict(x)
y_pred

#classification report of the created model
print('classification_report = '+classification_report(y,y_pred))

#confusion matrix of the given model
print('confusion_matrix = ',confusion_matrix(y,y_pred))

#Accuracy score
print('accuracy_score = ',accuracy_score(y,y_pred))

#model score
model.score(x,y)
```

Output:



```
x1 =  [0 1 1 2 0 2 0 1 0 1]
x2 =  [0 0 1 1 1 1 0 0 1 1]
x3 =  [1 1 0 0 1 1 0 1 0 1]

x =    0  1  2
0  0  0  1
1  1  0  1
2  1  1  0
3  2  1  0
4  0  1  1
5  2  1  1
6  0  0  0
7  1  0  1
8  0  1  0
9  1  1  1
classification_report =              precision    recall  f1-score   support

           0       1.00      0.67      0.80         6
           1       0.67      1.00      0.80         4

    accuracy                           0.80        10
   macro avg       0.83      0.83      0.80        10
weighted avg       0.87      0.80      0.80        10

confusion_matrix =  [[4 2]
 [0 4]]
accuracy_score =  0.8
0.8
```

Result:

Training the data using **Bayesian classification** Was implemented successfully and output is verified...