classmate
Date 11/04/2023.
Page

SUPERVISED LEARNING ALGORITHMS.

1. LogisticRegression : Classify a person enquiring online is a Man or Woman for gym center.

#Step-1 : import the packages i.e Select ML Algorithm to Apply.

```
from sklearn import linear_model
```

#Step2 : Load the Training Data

```
X = [ [165, 19], [175, 32], [136, 35], [174, 65],
      [141, 28], [176, 15], [131, 32], [166, 6] .... ]

Y = [ 'Man', 'Women', 'Women', 'Man', 'Women',
      'Man', 'Women', 'Man', 'Women' .... ]

data_features_name = ['height', 'age']
```

Step 3 : Create a Model

```
LR_model = linear_model.LogisticRegression()
```

#Step-4 : Train the Model – to identify a person is male or female

```
LR_model.fit(X, Y)
```

#Step-5 : Predict the outcome of the new observation.

What is the gender of a person with height 156 cms & age 53 years?

```
prediction = LR_model.predict([[169, 19]])
print(prediction) # ['Man']

print('Accuracy on the training subset:',
      format(LR_model.score(X, Y)))
```

2 Linear Regression: Forecasting example using Regression model
Forecast num of cars sold in Millions.

# Step1: Import the package i.e Select a ML algorithm to apply
from sklearn.linear_model import LinearRegression.

Step2: Load the Training or Historic Data.

X = [[2001, 5.2], [2002, 5.1], [2003, 5.1],
[2004, 4.9], [2005, 5.0], [2006, 5.1] ...]

Y = [2.5, 2.52, 2.54, 2.48, 2.52, 2.54 ....]

data = [year, GDP]

# step3: Create a Model
LR_model = LinearRegression()

# step 4: Train or Fit the model to data
LR_model.fit(X, Y)

# step 5: predict the outcome for the new observation
prediction = LR_model.predict([[2021, 6.1]])
print(prediction) → [3.28661901]

prediction_2022 = LR_model.predict([[2022, 6.4]])
print(prediction_2022)
[3.39155433]

3. Support Vector Machine

```
from sklearn.svm import SVC

X = [[165, 19], [175, 32], [136, 35], [170, 12],
     [140, 6], [180, 19], [126, 25] ... ]

Y = ['Man', 'Women', 'Women', 'Man', 'Women', 'Man',
     'Man']

data_features_names = ['height', 'age']

SVC_model = SVC( gamma = 'auto')

SVC_model.fit(X, Y)

print ( SVC_model.predict ( [[156, 53]]);
['Woman']
```

4 Gaussian Naive Bayes Model

```
from sklearn.naive_bayes import GaussianNB
import numpy as np

X = np.array([ [-3,7],[1,5],[1,2],[-2,0],[2,3],
               [-4,0],[-1,1],[-2,2],[2,7]..])
Y = np.array([3,3,3,3,4,3,3,4,3,4])

model = GaussianNB()

model.fit(X,Y)

prediction = model.predict[[[1,2]])
print(prediction)
[3]
```

KNN:

```
import sklearn.neighbors import NearestNeighbors
import numpy as np
import matplotlib.pyplot as plt

A = np.array([[3.1, 2.3], [2.3, 4.2], [3.9, 3.5],
              [3.7, 6.4], [4.8, 1.9], [8.3, 3.1]...])

plt.figure()
plt.title('Input Data')
plt.scatter(A[:,0], A[:,1], marker='x', s=50,
            color='red')

test_data = [5.2, 2.9]

knn_model = NearestNeighbors(n_neighbors=3,
                             algorithm='auto')
knn_model.fit(A)

distances, indices = knn_model.kneighbors([test_data])

print("K Nearest Neighbors are:")
for rank, index in enumerate(indices[0][:3], start=1):
    print(str(rank) + " is", A[index])
```

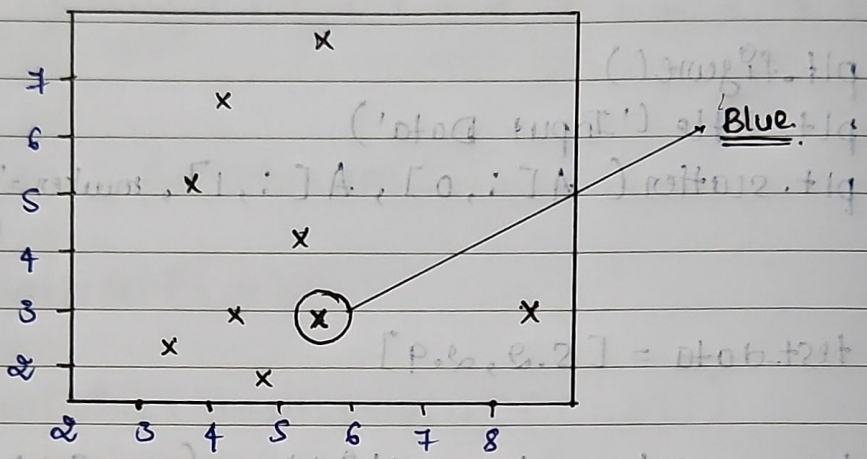
⇒

K Nearest Neighbors are:
1 is [4.4, 2.9]
2 is [4.8, 1.9]
3 is [3.9, 3.5]

P.T.O →

```
plt.figure()
plt.title("Nearest Neighbors")

plt.scatter(A[:,0], A[:,1], marker='x', s=100, color='red')
plt.scatter(test_data[0], test_data[1], marker='x',
            s=100, color='blue')
plt.show()
```



UN-SUPERVISED LEARNING ALGORITHM.

1 Market Basket Analysis using Apriori Algorithm: Association

```
! pip install efficient-apriori

from efficient-apriori import apriori

#Market Data
transactions = [
                ('butter', 'milk', 'bread'),
                ('butter', 'milk', 'apple'),
                ('bread', 'milk', 'banana'),
                ('milk', 'bread', 'butter')
               ]

itemsets, rules = apriori(transactions, min_support=0.5,
                          min_confidence=1)

print(rules)
print(itemsets)
```

2 K-Means Algorithm : Clustering

```
from sklearn.cluster import KMeans

X = [ [165, 19], [175, 32], [136, 35], [174, 65],
      [141, 28], [176, 15], [131, 32], [166, 6] ... ])
data_features = ["Height", "Age"]

model = KMeans (n_clusters = 3)

model.fit (X)

cluster_labels = model.predict(X)
print (cluster_labels)
  [2, 2, 1, 0, 1, 2, 1, 2, 1 ... ]

x1 = []   # height
x2 = []   # age

for item in X:
    x1.append (item[0]);
    x2.append (item [1]);

print(x1)
print(x2)

import matplotlib.pyplot as plt

plt.scatter (x1, x2, c = model.labels_)
```