Experiment 04 - Classification using python on the dataset.

Roll No.	19
Name	Manav Jawrani
Class	D15A
Subject	Business Intelligence Lab
LO Mapped	LO2: Organize and prepare the data needed for data mining algorithms in terms of attributes and class inputs, training, validating, and testing files. LO4: Implement various data mining algorithms from scratch using languages like Python/ Java etc.
Grade	

~~A~~ 9. DATE:

Aim - Classification using python on the dataset.

Theory -

- Classification algorithm -

1. Decision tree -

A decision tree is a supervised learning algorithm that builds a tree-like model to predict the value of a target variable by learning decision rules inferred from the data features.

Steps to form decision tree in python :

1. Import necessary libraries.
2. Load and preprocess dataset.
3. Split the dataset into training and test sets.
4. Create a fit of 'decision Tree classifier' model.
5. Use a model to make predictions on the testing data.
6. Evaluate the accuracy of the model using 'accuracy-score'
7. Generate a classification report using 'classification-report'.

2. Naive Bayes -

Naive Bayes is a probabilistic algorithm used for classification and is based on Bayes theorem. It assumes

DATE:

that the features are independent of each other and calculates the probability of each class based on the frequency of each feature in training set.

Steps to perform in python:

1. Import necessary libraries
2. Load and preprocess the dataset.
3. Split the dataset in training and test data.
4. Create a 'GaussianNB' model.
5. Fit the model to the training data using the 'fit' method.
6. Use the 'predict' method to make predictions on the testing data.
7. Evaluate the accuracy of model using 'accuracy_score'
8. Generate a classification report using 'classification_report'.

Date
Page

• Working of algorithms -

1. Decision Tree:

a. Compute the Gini index or entropy of the original dataset. The Gini index is a measure of impurity of a dataset. For a dataset D, the Gini index G(D) is calculated as:

$$G(D) = 1 - \Sigma_i (p_i)^2$$

where p_i is the proportion of samples in D that belong to class i. Alternatively we can measure the uncertainity of a dataset.

$$H(D) = -\Sigma_i p_i \log_2 (p_i)$$

b. Calculate the information gain of each feature: The information gain is a measure of the effectiveness of a feature in classifying the data. For a feature A and data set D, the information gain $IG(D, A)$ is calculated as:

$$IG(D, A) = H(D) - \Sigma_v \left(\frac{|D_v|}{|D|} \right) H(D_v)$$

where v is the possible values of A. D_v is the subset of D, where $A = v_i$ and $|D_v|$ is the no. of samples in D_v.

c. Select the feature with highest information gain: The feature with highest information gain is selected as root node.

d. Split the dataset based on the feature selected feature.

Teacher's Sign.: _____

2. Naive Bayes –

a. Prepare the training data

b. Calculate class probabilities

c. Calculate conditional probabilities : calculate the likelihood of each feature. For a feature f and class C, the conditional probability P(F|C) is calculated as:

$$P(F|C) = \frac{\text{(no. of instances with feature f and class C)}}{\text{(total no. of instances with class C)}}$$

d. Apply bayes theorem - calculate the posterior probability of each class given the observed features. For a class C and set of features X

$$\therefore P(C|X) = P(C) \times \Pi_i P(X_i|C)$$

where x_i is the ith feature in X and Π_i denotes the product over all i features.

e. Finally, select the class with highest posterior probability as predicted class for given features.

Teacher's Sign.: _____

Examples -

Q. Decision tree -

Tid	Income	Age	Own House
1	very high	Young	yes.
2	High.	medium	yes
3	Low	Young	Rented
4	High	medium.	yes
5	very High	medium.	yes
6	medium	young	yes
7	High.	old	yes
8	medium.	medium.	Rented
9	Low	medium.	Rented
10	Low	old	Rented
11	High	young	yes
12	medium.	old	Rented

Soln: -

class P : Own house = 'yes' , class N: own house = 'Rented'

Total no. of records = 12

No. of records with 'yes' = 7 and with 'No' = 5

Information gain = $I(P,N) = -\log 2 - \log 2$

$$I(P,N) = I(7,5) = -\frac{7}{12}\log_2\left(\frac{7}{12}\right) - \frac{5}{12}\log\left(\frac{5}{12}\right)$$

$$= 0.979$$

Compute the entropy for income -

For income = very high,

Pi = with 'yes' class = 2 and Ni, with 'no' class = 0

$$I(Pi, Ni) = I(2,0) = 0.$$

Teacher's Sign.: _____

Similarly for different ranges.

Income	Pi	ni	I (Pi, ni).
very High	2	0	0.
High.	4	0.	0
medium	1	2	0.918
Low	0	3.	0

$\therefore E(A) = I(P_i, n_i)$

$E(\text{Income}) = \frac{2}{12} \times I(2,0) + \frac{4}{12} \times I(4,0) + \frac{3}{12} \times I(0,3) = 0.229.$

Hence,

Gain (S, Income) = I(P,n) − E(income)

= 0.979 − 0.229 = 0.75

o. Compute the entropy for Age: [Young, medium, old]

Age	Pi	ni	I (Pi, ni)
Young	3	1	0.811
medium	3	2	0.971
old	1	2	0.918

Calculating entropy, E(A) = I(Pi, ni).

$E(Age) = \frac{4}{12} \times I(3,1) + \frac{5}{12} \times I(3,2) + \frac{3}{12} I(1,2)$

$\therefore E(Age) = 0.904.$

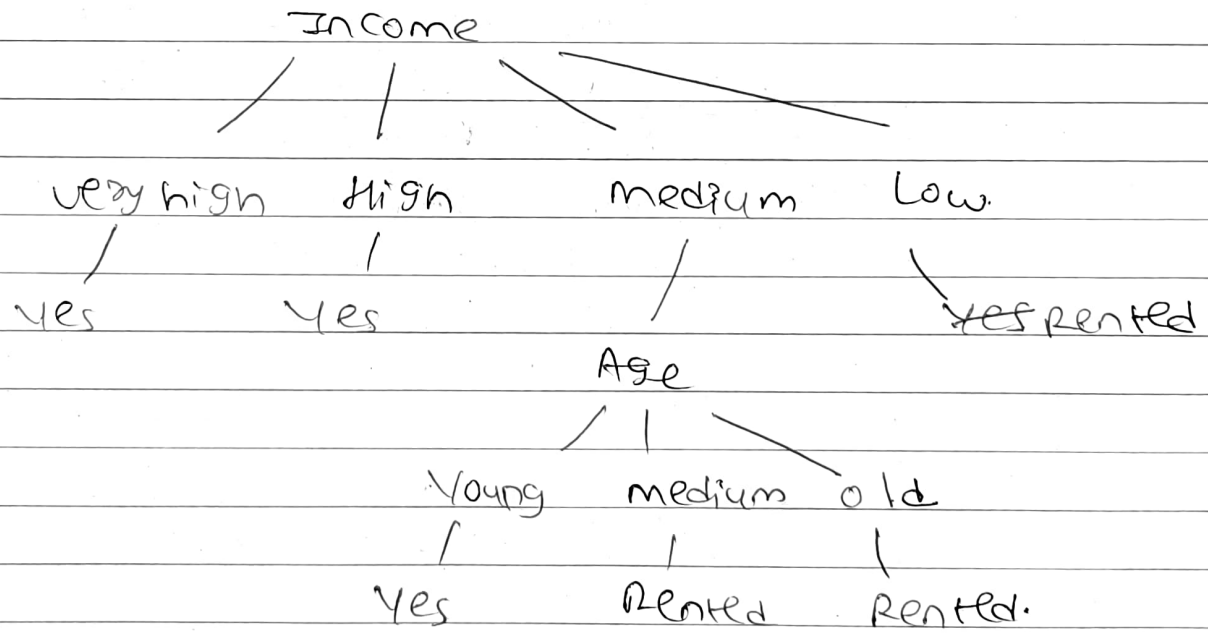
Teacher's Sign.: _____

Hence,

$$\text{Gain}(S, \text{age}) = I(p,n) - E(\text{age})$$
$$= 0.979 - 0.904$$
$$= 0.075$$

Since, Income attribute has highest gain, therefore it is root node.

Since we have used Income at root, now we have to decide on the age attribute. So final tree.



Teacher's Sign.: _____

b. Naive Bayes -

Case No.	Color	Type	Origin	Stolen?
1	Red	Sports	Domestic	Yes
2	Red	Sports	Domestic	No
3	Red	Sports	Domestic	Yes
4	Yellow	Sports	Domestic	No
5	Yellow	Sports.	Imported	Yes
6	Yellow	SUV	Imported	No
7	Yellow	SUV	Imported	Yes
8	Yellow	SUV.	Domestic	No
9	Red	SUV.	Imported	No
10	Red	Sports	Imported	Yes.

soln.

We want to classify a <Red, Domestic, SUV>, i.e. unseen sample. Note there is no such example

$\therefore P(Yes) = 5/10$

$P(No) = 5/10$

Teacher's Sign.: _____

Now,

Color -

P(Red | Yes) = 3/5

P(Red | No) = 2/5

P(Yellow | Yes) = 2/5

P(Yellow | No) = 3/5

Type -

P(SUV | Yes) = 1/5

P(SUV | No) = 3/5

P(Sports | Yes) = 4/5

P(Sports | No) = 2/5

Origin -

P(Domestic | Yes) = 2/5 P(Domestic | No) = 3/5

P(Imported | Yes) = 3/5 P(Imported | No) = 2/5

Teacher's Sign.: _____

An unseen example $X = \langle Red, Domestic, SUV \rangle$

$P(X|Yes) \cdot P(Yes) = P(Red|Yes) \times P(Domestic|Yes) \times P(SUV|Yes) \cdot P(Yes)$

$= 3/5 \times 2/5 \times 1/5 \times 5/10$

$= 0.024$

$P(X|No) \cdot P(No) = P(Red|No) \times P(Domestic|No) \times P(SUV|No) \times P(No)$

$= 2/5 \times 3/5 \times 3/5 \times 5/10$

$= 0.072$

Since $0.072 > 0.024$, our sample gets classified as 'No'.

Teacher's Sign.: _____

DATE:

Python library functions used-

1. 'accuracy_score' from 'sklearn.metrics' : calculates the accuracy of the machine learning model.
2. 'classification_report' from 'sklearn.metrics': Generates a report of classification metrics for a model.
3. 'confusion matrix Display' from 'sklearn.metrics' : Displays confusion ~~metrics~~ matrix in a visually informative way.
4. 'Decision Tree Classifier' from 'sklearn.tree' : Builds decision tree models.
5. 'f1_score' from 'sklearn.metrics' : calculates the F1 score of a model.
6. 'GaussianNB' from 'sklearn.naive_bayes' : Builds Gaussian Naive Bayes models.
7. 'matplotlib.pyplot' as 'plt' : Plotting library for Python.
8. 'numpy' as 'np' : Provides support for large, multi-dimensional arrays and matrices.
9. 'pandas' as 'pd' : Data manipulation and analysis library for python.
10. 'seaborn' as 'sns' : Python data visualization library based on mat plot lib.
11. 'train_test_split' from 'sklearn.model_selection' : Splits data into training and test set.

DATE:

Observation:

Final accuracies we found are:

1. Decision tree - ~~93.34%~~ 92.39%
2. Naive Bayes - 93.34%

Conclusion:

1. From the above accuracies we can conclude that both the classifiers has given us a high level of accuracy.
2. Also, we can see that the Naive Bayes classifier has a ~~higher~~ accuracy than Decision tree.
3. Thus, we can say that Naive Bayes model is more suitable for our dataset.
4. From our previous experiments also we found that Naive Bayes is more suitable for our dataset which we performed on rapid miner.

Implementation:

All details of dataset

```
[57] from google.colab import files
     uploaded = files.upload()
```

Choose Files new_dataset.csv

- **new_dataset.csv**(text/csv) - 327451 bytes, last modified: 2/4/2023 - 100% done

```
Saving new_dataset.csv to new_dataset (2).csv
```

```
[58] # All the necessary imports for classification
     import matplotlib.pyplot as plt
     import numpy as np
     import pandas as pd
     import seaborn as sns
     from sklearn import metrics
     from sklearn.metrics import (accuracy_score, classification_report,
     confusion_matrix, f1_score)
     from sklearn.metrics import ConfusionMatrixDisplay
     from sklearn.model_selection import train_test_split
     from sklearn.naive_bayes import GaussianNB
     from sklearn.tree import DecisionTreeClassifier
```

```
[31] dataset = pd.read_csv("new_dataset.csv")
```

```
dataset.describe()
```

	id	age	hypertension	heart_disease	avg_glucose_level	bmi	stroke
count	5110.000000	5110.000000	5110.000000	5110.000000	5110.000000	4909.000000	5110.000000
mean	36517.829354	43.226614	0.097456	0.054012	106.147677	28.893237	0.048728
std	21161.721625	22.612647	0.296607	0.226063	45.283560	7.854067	0.215320
min	67.000000	0.080000	0.000000	0.000000	55.120000	10.300000	0.000000
25%	17741.250000	25.000000	0.000000	0.000000	77.245000	23.500000	0.000000
50%	36932.000000	45.000000	0.000000	0.000000	91.885000	28.100000	0.000000
75%	54682.000000	61.000000	0.000000	0.000000	114.090000	33.100000	0.000000
max	72940.000000	82.000000	1.000000	1.000000	271.740000	97.600000	1.000000

```
[5] dataset = dataset.dropna(subset=['bmi'])
```

```
dataset.describe()
```

	id	age	hypertension	heart_disease	avg_glucose_level	bmi	stroke
count	4909.000000	4909.000000	4909.000000	4909.000000	4909.000000	4909.000000	4909.000000
mean	37064.313506	42.865374	0.091872	0.049501	105.305150	28.893237	0.042575
std	20995.098457	22.555115	0.288875	0.216934	44.424341	7.854067	0.201917
min	77.000000	0.080000	0.000000	0.000000	55.120000	10.300000	0.000000
25%	18605.000000	25.000000	0.000000	0.000000	77.070000	23.500000	0.000000
50%	37608.000000	44.000000	0.000000	0.000000	91.680000	28.100000	0.000000
75%	55220.000000	60.000000	0.000000	0.000000	113.570000	33.100000	0.000000
max	72940.000000	82.000000	1.000000	1.000000	271.740000	97.600000	1.000000

Splitting into training and test data

```
[17] #split dataset in features and target variable
     feature_cols = ['id', 'age', 'hypertension', 'heart_disease', 'avg_glucose_level', 'bmi']
     X = dataset[feature_cols] # Features
     y = dataset.stroke # Target variable
```

```
[18] # Split dataset into training set and test set
     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1) # 70% training and 30% test
```

Decision Tree

```
[59] #Train the Classification model on the Training set
     clf = DecisionTreeClassifier()
     clf = clf.fit(X_train,y_train)
     y_pred = clf.predict(X_test)
```

```
[21] # Defining the decision tree algorithm
     dtree=DecisionTreeClassifier()
     dtree.fit(X_train,y_train)
     print('Decision Tree Classifier Created')
```

```
Decision Tree Classifier Created
```

```
[64] # Model Accuracy, how often is the classifier correct?
     print("Accuracy:",metrics.accuracy_score(y_test, y_pred)*100)
     cm=confusion_matrix(y_test, y_pred)
     print("Confusion Matrix:\n",cm)
```

```
Accuracy: 92.39646978954514
Confusion Matrix:
 [[1350   59]
 [  53   11]]
```

Naive Bayes

```
[65] #Model building and training
     from sklearn.naive_bayes import GaussianNB

     model = GaussianNB()

     model.fit(X_train, y_train);
```

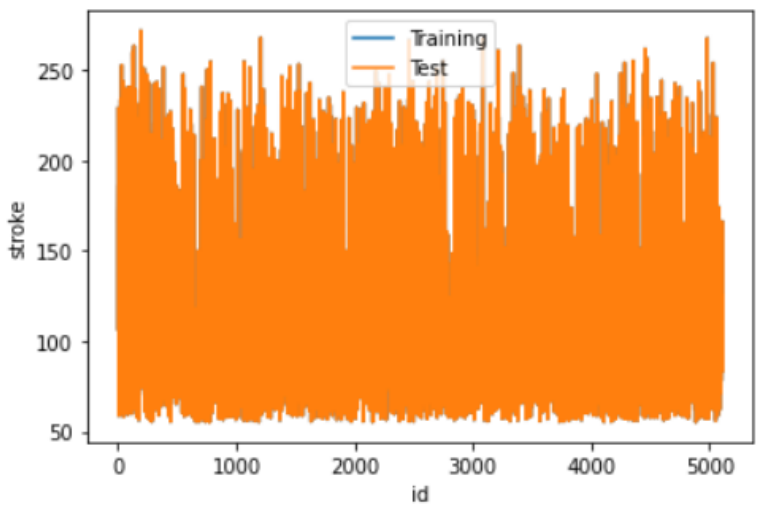
```
#Model Evaluation
from sklearn.metrics import (
    accuracy_score,
    confusion_matrix,
    ConfusionMatrixDisplay,
    f1_score,
    classification_report,
)

y_pred = model.predict(X_test)

accuray = accuracy_score(y_pred, y_test)*100
f1 = f1_score(y_pred, y_test, average="weighted")

print("Accuracy:", accuray)
print("F1 Score:", f1)

labels = [0,1]
cm = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:\n",cm)
```

```
Accuracy: 93.346911065852
F1 Score: 0.9355926372751519
Confusion Matrix:
 [[1364   45]
 [  53   11]]
```

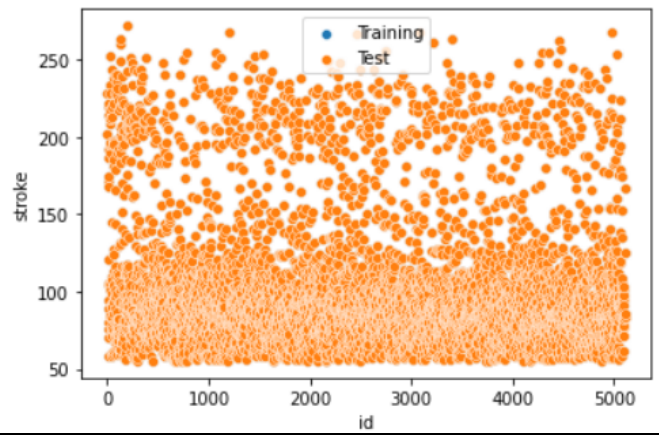
Visualization

```
[41] import matplotlib.pyplot as plt

     # Plot a line graph of the ratings in the training dataset
     plt.plot(dataset.index, dataset['avg_glucose_level'], label='Training')

     # Plot a line graph of the ratings in the test dataset
     plt.plot(dataset.index, dataset['avg_glucose_level'], label='Test')

     # Add a legend and labels to the plot
     plt.legend()
     plt.xlabel('id')
     plt.ylabel('stroke')

     # Show the plot
     plt.show()
```

```
# Show the plot
plt.show()
```



```
# Plot a scatterplot of the avg_glucose_level in the training dataset
sns.scatterplot(x=dataset.index, y='avg_glucose_level', data=dataset, label='Training')

# Plot a scatterplot of the avg_glucose_level in the test dataset
sns.scatterplot(x=dataset.index, y='avg_glucose_level', data=dataset, label='Test')

# Add a legend and labels to the plot
plt.legend()
plt.xlabel('id')
plt.ylabel('stroke')

# Show the plot
plt.show()
```



0s

```
# Plot a histplot of the avg_glucose_level in the training dataset
sns.histplot(x=dataset.index, y='work_type', data=dataset, label='Training')

# Plot a histplot of the avg_glucose_level in the test dataset
sns.histplot(x=dataset.index, y='work_type', data=dataset, label='Test')

# Add a legend and labels to the plot
plt.legend()
plt.xlabel('id')
plt.ylabel('stroke')

# Show the plot
plt.show()
```

```
WARNING:matplotlib.legend:No artists with labels found to put in legend.  Note t
```