Pattern recognition

Assignment

Name :- Abdelrahman Ahmed Fathy

ID :- 20221441784 years: 3rd **Ai**

Naive Bayes Classifier

A Naive Bayes Classifier is a simple probabilistic machine learning model that calculates the probability of a given input belonging to a certain class based on the probabilities of its features. It's called "naive" because it makes the assumption that the features are independent of each other, which simplifies the calculation.

Git hub link

https://github.com/Abdelrahman20244/patterns-assignment-20221441784

1st step

We import and define data from the url and try to handel theNA_values

```
url = "https://archive.ics.uci.edu/ml/machine-learning-
databases/adult/adult.data"
```

```
1]:
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import confusion_matrix


# Define the URL from which data will be imported
url = "https://archive.ics.uci.edu/ml/machine-learning-databases/adult/adult.data"
# Define column names for the dataset
columns = [
    "age", "workclass", "fnlwgt", "education", "education-num", "marital-status",
    "occupation", "relationship", "race", "sex", "capital-gain", "capital-loss",
    "hours-per-week", "native-country", "income"
]
# Read data from the specified URL into a pandas DataFrame
# na_values=' ?' specifies that any occurrence of ' ?' in the data should be treated as NaN
# skipinitialspace=True skips any initial spaces after delimiter in each line
data = pd.read_csv(url, names=columns, na_values=' ?', skipinitialspace=True)
```

2nd data preprocessing

for better classification we need to make data preprocessing like

- drop rows of missing values
- we make a map to convert income into binary labels for classification
- split data into feature(x) and target variable (y)

```
# Drop rows with missing values
data = data.dropna()
# Encode categorical variables
data = pd.get_dummies(data, columns=["workclass", "education", "marital-status", "occupation", "relationship", "race", "sex",
# Map income to binary values (0 for <=50K, 1 for >50K)
#This step converts the income column into binary labels for classification
data['income'] = data['income'].map({'<=50K': 0, '>50K': 1})

# Split data into features (X) and target variable (y)
X = data.drop('income', axis=1)# Features (independent variables)
y = data['income']  # Target variable (dependent variable)
```

3rd Naive Bayes Classifier

Split data into training and test data sets for better classification

- We split 20% for testing and 80% for training

Initialize naïve bayes classifier [GaussianNB] the train using tainin data

```
# Split the data into training and testing sets
# 20% for testing,80% for training
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Initialize a Naive Bayes classifier (Gaussian Naive Bayes in this case)
nb_classifier = GaussianNB()

# Train the Naive Bayes classifier using the training data
nb_classifier.fit(X_train, y_train)
```

```
GaussianNB()
```

4th Computation OF Sensitivity and Specificity

To find sensitivity and Specificity we need to the get the confusion matrix to get the performance of the classifier a

It tell the possible correct and incorrect predictions made by the classifier

`tn = true negative, fp = false positive, fn = false negative, tp = true positive`

```
# Use Naive Bayes classifier to make predictions on the test data
y_pred = nb_classifier.predict(X_test)
# Compute confusion matrix to evaluate the performance of the classifier
# It summarizes the number of correct and incorrect predictions made by the classifier
# tn = true negative, fp = false positive, fn = false negative, tp = true positive
tn, fp, fn, tp = confusion_matrix(y_test, y_pred).ravel()
sensitivity = tp / (tp + fn)
specificity = tn / (tn + fp)
print("Sensitivity:", sensitivity)
print("Specificity:", specificity)
```

```
Sensitivity: 0.32017823042647997
Specificity: 0.9514366653176851
```

Sensitivity (True Positive Rate)

- 0.32017823042647997 (approximately 32.02%)

This means

The classifier correctly identified about 32.02% of individuals who actually earn over 50K a year (positive instances).

Specificity (True Negative Rate)

- 0.9514366653176851 (approximately 95.14%)

This means

The classifier correctly identified about 95.14% of individuals who actually earn less than or equal to 50K a year (negative instances).

5th Posterior Probability

The posterior probability, in the context of Bayesian statistics, refers to the updated probability of an event occurring after considering new evidence or information. It's calculated using Bayes' theorem, which combines prior knowledge about the event with the likelihood of observing the evidence given the event. The posterior probability reflects our updated belief about the event given the evidence we have observed.

- Extract the probability of the positive class (income >50K) from the posterior probabilities

```
# Use Naive Bayes classifier to predict probabilities for each class in the test data
posterior_probs = nb_classifier.predict_proba(X_test)
# Extract the probability of the positive class (income >50K) from the posterior probabilities
positive_class_probs = posterior_probs[:, 1]
print("Posterior Probability of making over 50K a year:", positive_class_probs)
```

```
Posterior Probability of making over 50K a year: [4.31088775e-03 1.37859620e-02 1.71229441e-02 ... 1.00000000e+00
 6.39925511e-03 6.53278628e-04]
```

- Probabilities closer to 1 indicate a higher likelihood that the individual earns over 50K a year.

- Probabilities closer to 0 indicate a lower likelihood that the individual earns over 50K a year. Instances with higher probabilities are more confidently predicted to belong to the positive class (income >50K), while instances with lower probabilities may have higher uncertainty associated with their predictions.

Name :- Abdelrahman Ahmed Fathy

ID :- 20221441784 years: 3rd **Ai**