Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης
Τομέας Ηλεκτρονικής και Υπολογιστών

9ο Εξάμηνο
Φθινόπωρο 2021

Naive Bayes και Classification Evaluation

1 Εισαγωγή

1.1 Εισαγωγή στο Naive Bayes

Ο Naive Bayes είναι ένας αλγόριθμος μηχανικής μάθησης που βασίζεται στο θεώρημα του Bayes καθώς και στην υπόθεση της ανεξαρτησίας μεταξύ των χαρακτηριστικών. Δεδομένου ενός συνόλου δεδομένων με χαρακτηριστικά $x_1, x_2, \ldots, x_n$ και χαρακτηριστικό-κλάση c, το θεώρημα του Bayes επιτρέπει τον υπολογισμό της πιθανότητας ένα δείγμα $x = \{x_1, x_2, \ldots, x_n\}$ να ανήκει στο c ως το συνδυασμό των πιθανοτήτων $P(x_1|c), P(x_2|c), \ldots, P(x_n|c)$:

$$P(c|x) = \frac{P(x_1|c) \cdot P(x_2|c) \cdot \ldots \cdot P(x_n|c) \cdot P(c)}{P(x_1) \cdot P(x_2) \cdot \ldots \cdot P(x_n)}$$

όπου παρατηρούμε ότι η πιθανότητα του συνόλου των χαρακτηριστικών δίνεται από το γινόμενό τους, καθώς οι τιμές των χαρακτηριστικών είναι ανεξάρτητες μεταξύ τους.

1.2 Naive Bayes στην Python

Για την κατασκευή του πιθανοτικού μοντέλου Naive Bayes στην Python, μπορούμε να χρησιμοποιήσουμε τη βιβλιοθήκη sklearn:

```
>>> from sklearn.naive_bayes import GaussianNB, CategoricalNB
```

Για να κατασκευάσουμε ένα μοντέλο τρέχουμε την εντολή:

```
>>> clf = GaussianNB() # Ή clf = CategoricalNB(alpha=alpha)
```

όπου με την παράμετρο alpha επιλέγουμε τη χρήση laplace smoothing.

Για να εκπαιδεύσουμε το μοντέλο στα δεδομένα εκπαίδευσης, χρησιμοποιούμε την εντολή fit:

```
>>> clf = clf.fit(X, y)
```

Έχοντας ένα μοντέλο, για να προβλέψουμε μια νέα τιμή τρέχουμε την εντολή

```
>>> print(clf.predict(x))
```

Και για να πάρουμε τις εκ των υστέρων πιθανότητες:

```
>>> print(clf.predict_proba(x))
```

1.2 Classification Evaluation στην Python

Για να αξιολογήσουμε ένα μοντέλο ταξινόμησης χρησιμοποιούμε τις παρακάτω βιβλιοθήκες:

```
>>> from sklearn.metrics import confusion_matrix, accuracy_score, precision_score, recall_score, f1_score, roc_curve, auc
```

Έχοντας το αποτέλεσμα ενός αλγορίθμου (`pred`) και τις πραγματικές τιμές μπορούμε να δούμε το confusion matrix και να υπολογίσουμε χρήσιμες μετρικές με τις παρακάτω εντολές:

```
>>> confusion_matrix(ytest, pred)
>>> accuracy_score(ytest, pred)
>>> precision_score(ytest, pred, pos_label)
>>> recall_score(ytest, pred, pos_label)
>>> f1_score(ytest, pred, pos_label)
>>> fpr, tpr, thresholds = roc_curve(ytest, pred_prob, pos_label)
>>> auc(fpr, tpr)
```

Μπορούμε να σχεδιάσουμε την καμπύλη με τις παρακάτω εντολές:

```
>>> import matplotlib.pyplot as plt
>>> plt.title('Receiver Operating Characteristic')
>>> plt.plot(fpr, tpr, 'b', label = 'AUC = %0.2f' % auc(fpr, tpr))
>>> plt.legend(loc = 'lower right')
>>> plt.plot([0, 1], [0, 1],'r--')
>>> plt.xlim([0, 1])
>>> plt.ylim([0, 1])
>>> plt.ylabel('True Positive Rate')
```

```
>>> plt.xlabel('False Positive Rate')
>>> plt.show()
```

2 Κατασκευή Μοντέλου Naive Bayes και Κατάταξη Τιμών

Για την κατασκευή ενός μοντέλου Naive Bayes θα χρησιμοποιήσουμε ως εφαρμογή τα δεδομένα εκπαίδευσης του παρακάτω πίνακα για ένα πρόβλημα δυαδικής ταξινόμησης.

Weather	Day	HighTraffic
Hot	Vacation	No
Cold	Work	Yes
Normal	Work	No
Cold	Weekend	Yes
Normal	Weekend	Yes
Cold	Work	No
Hot	Work	No
Hot	Vacation	Yes

Θα απαντήσουμε στα παρακάτω ερωτήματα:

α) Χρησιμοποιώντας τον Naive Bayes Classifier, σε ποια κλάση θα κατατάσσατε μία νέα παρατήρηση με τιμές (Weather, Day) = (Hot, Vacation);

β) Χρησιμοποιώντας τον Naive Bayes Classifier, σε ποια κλάση θα κατατάσσατε μία νέα παρατήρηση με τιμές (Weather, Day) = (Hot, Weekend);

γ) Επαναλάβετε το ερώτημα (β) χρησιμοποιώντας Laplace smoothing.

δ) Κατασκευάστε το πλήρες μοντέλο στην Python και επαναλάβετε το ερώτημα (α).

ε) Κατασκευάστε το πλήρες μοντέλο στην Python χρησιμοποιώντας Laplace Smoothing και επαναλάβετε το ερώτημα (β).

2.1 Εισαγωγή Δεδομένων και Βιβλιοθηκών

Αρχικά διαβάζουμε τα δεδομένα και φορτώνουμε τις απαραίτητες βιβλιοθήκες:

```
>>> import pandas as pd
>>> from sklearn.preprocessing import OneHotEncoder
>>> from sklearn.naive_bayes import CategoricalNB
>>> traffic = pd.read_csv(“./traffic.txt”)
```

2.2 Υπολογισμός τιμής πιθανότητας

Για το ερώτημα (α) υπολογίζουμε τις πιθανότητες με τους παρακάτω τύπους:

$$P(Yes|Hot, Vacation) = \frac{P(Hot|Yes) \cdot P(Vacation|Yes) \cdot P(Yes)}{P(Hot) \cdot P(Vacation)} = \frac{\frac{1}{4} \cdot \frac{1}{4} \cdot \frac{1}{2}}{\frac{3}{8} \cdot \frac{2}{8}} = \frac{1}{3}$$

$$P(No|Hot, Vacation) = \frac{P(Hot|No) \cdot P(Vacation|No) \cdot P(No)}{P(Hot) \cdot P(Vacation)} = \frac{2/4 \cdot 1/4 \cdot 1/2}{3/8 \cdot 2/8} = 2/3$$

Αφού *P(No|Hot,Vacation) > P(Yes|Hot,Vacation)*, ο αλγόριθμος θα κατατάξει την τιμή ως No.

Αντίστοιχα, για το ερώτημα (β):

$$P(Yes|Hot, Weekend) = \frac{P(Hot|Yes) \cdot P(Weekend|Yes) \cdot P(Yes)}{P(Hot) \cdot P(Weekend)} = \frac{1/4 \cdot 2/4 \cdot 1/2}{3/8 \cdot 2/8} = 2/3$$

$$P(No|Hot, Weekend) = \frac{P(Hot|No) \cdot P(Weekend|No) \cdot P(No)}{P(Hot) \cdot P(Weekend)} = \frac{2/4 \cdot 0/4 \cdot 1/2}{3/8 \cdot 2/8} = 0$$

Αφού *P(No|Hot,Weekend) < P(Yes|Hot,Weekend)*, ο αλγόριθμος θα κατατάξει την τιμή ως Yes.

Για το ερώτημα (γ) θα υπολογίσουμε εκ νέου τις πιθανότητες προσθέτοντας όμως τη μονάδα ως Laplaciανό συντελεστή, οπότε έχουμε τις πιθανότητες $P(Hot|Yes) = (1+1)/(4+3) = 2/7$, $P(Weekend|Yes) = (2+1)/(4+3) = 3/7$, $P(Hot|No) = (2+1)/(4+3) = 3/7$ και την πιθανότητα $P(Weekend|No) = (0+1)/(4+3) = 1/7$, άρα τελικά έχουμε τα παρακάτω:

$$P(Yes|Hot, Weekend) = \frac{P(Hot|Yes) \cdot P(Weekend|Yes) \cdot P(Yes)}{P(Hot) \cdot P(Weekend)} = \frac{2/7 \cdot 3/7 \cdot 1/2}{3/8 \cdot 2/8} = 32/49$$

$$P(No|Hot, Weekend) = \frac{P(Hot|No) \cdot P(Weekend|No) \cdot P(No)}{P(Hot) \cdot P(Weekend)} = \frac{3/7 \cdot 1/7 \cdot 1/2}{3/8 \cdot 2/8} = 16/49$$

Άρα, αφού *P(No|Hot,Weekend) < P(Yes|Hot,Weekend)*, ο αλγόριθμος θα κατατάξει την τιμή ως Yes.

2.3 Κατασκευή Μοντέλου με την Python

2.3.1 Κατασκευή Απλού Μοντέλου

Αρχικά, διαχωρίζουμε τη στήλη-κλάση από τα δεδομένα:

```
>>> X = traffic.loc[:, [“Weather”, “Day”]]
>>> y = traffic.loc[:, “HighTraffic”]
```

και μετατρέπουμε τα δεδομένα σε One-Hot Encoding:

```
>>> encoder = OneHotEncoder(handle_unknown=”ignore”, sparse=False)
>>> encoder = encoder.fit(X)
>>> X = encoder.transform(X)
```

Για το ερώτημα (δ) εφαρμόζουμε το Naive Bayes για το dataset:

```
>>> clf = CategoricalNB(alpha = 0)
>>> clf.fit(X, y)
```

Υπολογίζουμε εκ νέου το ερώτημα (α):

```
>>> new_data = pd.DataFrame({"Weather": ["Hot"], "Day":
["Vacation"]})
>>> transformed_new_data = encoder.transform(new_data)
>>> print(clf.predict(transformed_new_data))
```

Μπορούμε επιπλέον να πάρουμε τις πιθανότητες:

```
>>> print(clf.predict_proba(transformed_new_data))
```

2.3.1 Κατασκευή Μοντέλου με Laplace Smoothing

Για το ερώτημα (ε) εφαρμόζουμε το Naive Bayes με Laplace Smoothing για το dataset:

```
>>> clf = CategoricalNB(alpha=1)
>>> clf.fit(X, y)
```

Υπολογίζουμε εκ νέου το ερώτημα (β):

```
>>> new_data = pd.DataFrame({"Weather": ["Hot"], "Day": ["Weekend"]})
>>> transformed_new_data = encoder.transform(new_data)
>>> print(clf.predict(transformed_new_data))
>>> print(clf.predict_proba(transformed_new_data))
```

3 Εφαρμογή με Μετρικές Αξιολόγησης και Καμπύλη ROC

Θα εφαρμόσουμε τον ταξινομητή Naive Bayes στο dataset iris. Αρχικά, εισάγουμε το dataset, επιλέγοντας ωστόσο μόνο τις 2 πρώτες στήλες και αλλάζοντας τις τελευταίες 50 τιμές του Species ώστε να κάνουμε το πρόβλημα binary classification:

```
>>> from sklearn import datasets
>>> import numpy as np
>>> iris = datasets.load_iris()
>>> data = iris.data[:, [0, 1]]
>>> target = iris.target
>>> target[100:125] = 1
>>> target[125:150] = 0
```

Κατόπιν, κάνουμε split το dataset σε training και testing data:

```
>>> xtrain = np.concatenate((data[0:40], data[50:90], data[100:140]))
>>> ytrain = np.concatenate((target[0:40], target[50:90],
target[100:140]))
>>> xtest = np.concatenate((data[40:50], data[90:100],
data[140:150]))
>>> ytest = np.concatenate((target[40:50], target[90:100],
target[140:150]))
```

Στη συνέχεια, θα απαντήσουμε στα παρακάτω ερωτήματα:

α) Κατασκευάστε ένα μοντέλο Naive Bayes χρησιμοποιώντας τα δεδομένα εκπαίδευσης και εφαρμόστε το μοντέλο στα test data.

β) Για τα test data υπολογίστε precision, recall και f-measure για την κλάση 1.

γ) Κατασκευάστε την καμπύλη ROC για το μοντέλο.

3.1 Κατασκευή και Εφαρμογή Μοντέλου Naive Bayes

Κάνουμε training το μοντέλο με τις παρακάτω εντολές (ερώτημα (α)):

```
>>> from sklearn.naive_bayes import GaussianNB
>>> clf = GaussianNB()
>>> clf.fit(xtrain, ytrain)
```

Μπορούμε στη συνέχεια να εφαρμόσουμε το μοντέλο στο test set:

```
>>> pred = clf.predict(xtest)
>>> predprob = clf.predict_proba(xtest)
```

3.2 Υπολογισμός Μετρικών Αξιολόγησης και Σχεδίαση ROC Curve

Μπορούμε να υπολογίσουμε χρήσιμες μετρικές με τις παρακάτω εντολές (ερώτημα (β)):

```
>>> from sklearn.metrics import accuracy_score, precision_score,
recall_score, f1_score, roc_curve, auc
>>> print("Accuracy: ", accuracy_score(ytest, pred))
>>> print("Precision: ", precision_score(ytest, pred, pos_label=1))
>>> print("Recall: ", recall_score(ytest, pred, pos_label=1))
>>> print("F1 Score: ", f1_score(ytest, pred, pos_label=1))
```

Για την καμπύλη ROC (ερώτημα (γ)) θα χρειαστεί αρχικά να υπολογίσουμε τα TPR και FRP με τις παρακάτω εντολές:

```
>>> fpr, tpr, thresholds = roc_curve(ytest, predprob[:, 1])
```

Μπορούμε να βρούμε την περιοχή κάτω από την καμπύλη με την εντολή:

```
>>> print("AUC: ", auc(fpr, tpr))
```

Τέλος, μπορούμε να σχεδιάσουμε την καμπύλη με τις παρακάτω εντολές:

```
>>> import matplotlib.pyplot as plt
>>> plt.title('Receiver Operating Characteristic')
```

```
>>> plt.plot(fpr, tpr, 'b', label = 'AUC = %0.2f' % auc(fpr, tpr))
>>> plt.legend(loc = 'lower right')
>>> plt.plot([0, 1], [0, 1],'r--')
>>> plt.xlim([0, 1])
>>> plt.ylim([0, 1])
>>> plt.ylabel('True Positive Rate')
>>> plt.xlabel('False Positive Rate')
>>> plt.show()
```