T.C.

BALIKESİR ÜNİVERSİTESİ

MÜHENDİSLİK FAKÜLTESİ

BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

PYTHON İLE NAİVE BAYES SINIFLANDIRICI

ÖRÜNTÜ TANIMA DERSİ DÖNEM PROJESİ

201713709032 Yunus Emre Arslan

Danışman:Dr. Öğr. Üyesi Fatih AYDIN

İçerikler

1. Naïve Bayes algoritması hakkında bilgi

Makine öğreniminde, Naïve Bayes sınıflandırması, sınıflandırma görevi için basit ve güçlü bir algoritmadır. Naïve Bayes sınıflandırması, özellikler arasında güçlü bağımsızlık varsayımı ile Bayes'in teoremini uygulamaya dayanmaktadır. Naïve Bayes sınıflandırması, Doğal Dil İşleme gibi metinsel veri analizi için kullandığımızda iyi sonuçlar verir.

Naïve Bayes modelleri, basit Bayes veya bağımsız Bayes olarak da bilinir. Tüm bu isimler, sınıflandırıcının karar kuralında Bayes'in teoreminin uygulanmasına atıfta bulunur. Naïve Bayes sınıflandırıcısı, Bayes'in teoremini pratikte uygular. Bu sınıflandırıcı, Bayes teoreminin gücünü makine öğrenimine taşır.

2. Naïve Bayes algoritması sezgisi

Naïve Bayes Sınıflandırıcı, verilen kayıt veya veri noktasının belirli bir sınıfa ait olma olasılığı gibi her sınıf için üyelik olasılıklarını tahmin etmek için Bayes teoremini kullanır. En yüksek olasılığa sahip sınıf, en olası sınıf olarak kabul edilir. Bu aynı zamanda Maksimum A Posteriori (MAP) olarak da bilinir.

A ve B 2 olaylı bir hipotez için MAP şu şekildedir:

```
MAP (A)

= max (P (A | B))

= max (P (B | A) * P (A))/P (B)

= max (P (B | A) * P (A))
```

Burada, P (B) kanıt olasılıktır. Sonucu normalleştirmek için kullanılır. Aynı kalır, yani kaldırılması sonucu etkilemez.

Naïve Bayes Sınıflandırıcı, tüm özelliklerin birbiriyle ilgisi olmadığını varsayar. Bir özelliğin varlığı veya yokluğu, başka herhangi bir özelliğin varlığını veya yokluğunu etkilemez.

Gerçek dünya veri kümelerinde, özellikler hakkında çok sayıda kanıt verilen bir hipotezi test ediyoruz. Yani hesaplamalar oldukça karmaşık hale geliyor. Çalışmayı basitleştirmek için, özellikten bağımsızlık yaklaşımı, birden fazla kanıtı ayırmak ve her birini bağımsız olarak ele almak için kullanılır.

3. Naïve Bayes algoritması türleri

3 tür Naive Bayes algoritması vardır.

- Gaussian Naïve Bayes
- Multinomial Naïve Bayes
- Bernoulli Naïve Bayes

Gaussian Naïve Bayes algoritması

Sürekli öznitelik değerlerimiz olduğunda, her bir sınıfla ilişkili değerlerin Gauss veya Normal dağılıma göre dağıldığını varsayıyoruz. Örneğin, eğitim verilerinin sürekli bir x özelliği içerdiğini varsayalım. Önce verileri sınıfa göre bölümlere ayırıyoruz ve ardından her sınıftaki x'in ortalamasını ve varyansını hesaplıyoruz. Mi değerlerin ortalaması olsun ve σi i'inci sınıfla ilişkili değerlerin varyansı olsun. Bir xi gözlem değerimiz olduğunu varsayalım. Daha sonra, bir sınıf verilen xi'nin olasılık dağılımı aşağıdaki denklemle hesaplanabilir.

Naïve Bayes algoritması nerelerde kullanılır?

- Spam filtering
- Text classification
- Sentiment analysis
- Recommender systems

Kütüphanlerin eklenmesi

```
import numpy as np # Lineer cebir
import pandas as pd # Veri işlemleri, CSV file I/O (e.g. pd.read_csv)
import matplotlib.pyplot as plt # Veri görselleştirme
import seaborn as sns # İstatistiksel veri görselleştirme
%matplotlib inline

# Input data files are available in the "../input/" directory.
# For example, running this (by clicking run or pressing Shift+Enter) will
list all files under the input directory

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
```

```

## Veri setinin eklenmesi python data = '/kaggle/input/adult-dataset/adult.csv'
df = pd.read_csv(data, header=None, sep=',\s')

# veri kümesinin boyutlarını görüntüle
df.shape
```

Sütunları yeniden adlandıralım

```
col_names = ['age', 'workclass', 'fnlwgt', 'education', 'education_num',
'marital_status', 'occupation', 'relationship',
             'race', 'sex', 'capital_gain', 'capital_loss', 'hours_per_week',
'native_country', 'income']

df.columns = col_names

df.columns
```

Veri setinin özetini görüntüleyelim

```
# Veri seti özellikleri

df.info()
```

Değişken türleri

Bu bölümde, veri setini kategorik ve sayısal değişkenlere ayırıyorum. Veri setinde kategorik ve sayısal değişkenlerin bir karışımı vardır. Kategorik değişkenlerin veri türü nesnesi vardır. Sayısal değişkenler int64 veri türüne sahiptir.

Kategorik değişkenleri keşfedin

```
categorical = [var for var in df.columns if df[var].dtype=='O']

print('There are {} categorical variables\n'.format(len(categorical)))

print('The categorical variables are :\n\n', categorical)
```

Kategorik değişkenleri görüntüleyin

```
df[categorical].head()
```

Kategorik değişkenlerin özeti

- 9 kategorik değişken vardır.
- Kategorik değişkenler çalışma sınıfı, eğitim, medeni_durum, meslek, ilişki, ırk, cinsiyet, yerel_ ülke ve gelir ile verilmektedir.
- gelir hedef değişkendir.

Kategorik değişkenlerdeki eksik değerleri tekrar kontrol edin

```
df['categorical'].isnull().sum()
```

Şimdi, çalışma sınıfı, meslek ve native_country değişkeninin eksik değerler içerdiğini görebiliriz.

Sayısal Değişkenleri Keşfedin

```
# Nümerik verileri bulun

numerical = [var for var in df.columns if df[var].dtype!='O']

print('There are {} numerical variables\n'.format(len(numerical)))

print('The numerical variables are :', numerical)
```

Nümerik verilerin özeti

- 6 sayısal değişken vardır.
- Bunlar yaş, fnlwgt, eğitim_sayısı, sermaye_geri, sermaye_kaybı ve saat_başına_hafta ile verilir.
- Tüm sayısal değişkenler ayrı veri tipindedir.

Sayısal değişkenlerde eksik değerler

```
# sayısal değişkenlerdeki eksik değerleri kontrol edin
df['numerical'].isnull().sum()

6 sayısal değişkenin tamamının eksik değerler içermediğini görebiliriz.
```

Özellik vektörlerini ve hedef değişkeni bildirin

```
X = df.drop(['income'], axis=1)

y = df['income']
```

Veri setini eğitim ve test seti olarak bölün

```
# X ve y'yi eğitim ve test setlerine ayırın
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3,
random_state = 0)

# X_train ve X_test'in şeklini kontrol edin

X_train.shape, X_test.shape
```

Nümerik değişkenleri görüntüleme

```
numerical = [col for col in X_train.columns if X_train[col].dtypes != 'O']
numerical
```

Engineering missing values in categorical variables

```
# eğitim setindeki kategorik değişkenlerdeki eksik değerlerin yüzdesini
yazdır

X_train[categorical].isnull().mean()

# Eksik verilerle kategorik değişkenleri yazdırın

for col in categorical:
    if X_train[col].isnull().mean()>0:
        print(col, (X_train[col].isnull().mean()))

# Eksik kategorik değişkenleri en sık değere dayandırın

for df2 in [X_train, X_test]:
    df2['workclass'].fillna(X_train['workclass'].mode()[0], inplace=True)
    df2['occupation'].fillna(X_train['occupation'].mode()[0], inplace=True)
    df2['native_country'].fillna(X_train['native_country'].mode()[0],
inplace=True)

# X_train'de kategorik değişkenlerdeki eksik değerleri kontrol edin

X_train[categorical].isnull().sum()
```

Son bir kontrol olarak, X_train ve X_test'teki eksik değerleri kontrol edilebilir.

```
# X_train'deki eksik değerleri kontrol edin
X_train.isnull().sum()

# X_test'deki eksik değerleri kontrol edin
X_test.isnull().sum()
```

X_train ve X_test'te eksik değer olmadığını görebiliriz.

```
# import category encoders

import category_encoders as ce

# kalan değişkenleri tek sıcak kodlama ile kodlayın

encoder = ce.OneHotEncoder(cols=['workclass', 'education', 'marital_status',
'occupation', 'relationship',
                                 'race', 'sex', 'native_country'])

X_train = encoder.fit_transform(X_train)

X_test = encoder.transform(X_test)


X_train.shape


X_test.shape
```

Özellik ölçekleme

```
cols = X_train.columns

from sklearn.preprocessing import RobustScaler

scaler = RobustScaler()

X_train = scaler.fit_transform(X_train)

X_test = scaler.transform(X_test)

X_train = pd.DataFrame(X_train, columns=[cols])

X_test = pd.DataFrame(X_test, columns=[cols])
```

Artık Gaussian Naive Bayes sınıflandırıcısına beslenmeye hazır X_train veri kümesine sahibiz.

Model eğitimi

```
# eğitim setinde bir Gaussian Naive Bayes sınıflandırıcısı eğitin
from sklearn.naive_bayes import GaussianNB
```

```
# modeli örneklemek
gnb = GaussianNB()


# modeli eğitme
gnb.fit(X_train, y_train)
```

Sonuçları tahmin edin

```
y_pred = gnb.predict(X_test)

y_pred
```

Doğruluk puanını kontrol edin

```
from sklearn.metrics import accuracy_score

print('Model accuracy score: {0:0.4f}'. format(accuracy_score(y_test,
y_pred)))
```

Burada, y_test gerçek sınıf etiketleridir ve y_pred, test kümesindeki tahmin edilen sınıf etiketleridir.

Tren seti ve test seti doğruluğunu karşılaştırın

Şimdi, aşırı uyumu kontrol etmek için tren seti ve test seti doğruluğunu karşılaştıracağım.

```
y_pred_train = gnb.predict(X_train)

y_pred_train

print('Training-set accuracy score: {0:0.4f}'. format(accuracy_score(y_train,
y_pred_train)))
```

Aşırı uyum ve yetersiz uyumu kontrol edin

```
# eğitim ve test setindeki puanları yazdırın

print('Training set score: {:.4f}'.format(gnb.score(X_train, y_train)))

print('Test set score: {:.4f}'.format(gnb.score(X_test, y_test)))
```

Eğitim seti doğruluk puanı 0.8067 iken test seti doğruluğu 0.8083'tür. Bu iki değer oldukça karşılaştırılabilir. Yani, aşırı uyum belirtisi yok.

Hata matrisi

Hata matrisi, bir sınıflandırma algoritmasının performansını özetlemek için kullanılan bir araçtır. Bir kafa karışıklığı matrisi, bize sınıflandırma modeli performansının ve modelin

ürettiği hata türlerinin net bir resmini verecektir. Bize her kategoriye göre ayrılmış doğru ve yanlış tahminlerin bir özetini verir. Özet, tablo biçiminde temsil edilir.

Bir sınıflandırma modeli performansını değerlendirirken dört tür sonuç mümkündür. Bu dört sonuç aşağıda açıklanmıştır: -

Gerçek Pozitifler (TP) - Gerçek Pozitifler, bir gözlemin belirli bir sınıfa ait olduğunu ve gözlemin aslında o sınıfa ait olduğunu tahmin ettiğimizde ortaya çıkar.

Gerçek Negatifler (TN) - Gerçek Negatifler, bir gözlemin belirli bir sınıfa ait olmadığını ve gözlemin aslında o sınıfa ait olmadığını tahmin ettiğimizde ortaya çıkar.

Yanlış Pozitifler (FP) - Yanlış Pozitifler, bir gözlemin belirli bir sınıfa ait olduğunu, ancak gözlemin aslında o sınıfa ait olmadığını tahmin ettiğimizde ortaya çıkar. Bu tür bir hataya Tip I hatası denir.

Yanlış Negatifler (FN) - Yanlış Negatifler, bir gözlemin belirli bir sınıfa ait olmadığını, ancak gözlemin aslında o sınıfa ait olduğunu tahmin ettiğimizde ortaya çıkar. Bu çok ciddi bir hatadır ve Tip II hatası olarak adlandırılır.

Bu dört sonuç, aşağıda verilen bir karışıklık matrisinde özetlenmiştir.

```
# Karışıklık Matrisini yazdırın ve dört parçaya bölün

from sklearn.metrics import confusion_matrix

cm = confusion_matrix(y_test, y_pred)

print('Confusion matrix\n\n', cm)

print('\nTrue Positives(TP) = ', cm[0,0])

print('\nTrue Negatives(TN) = ', cm[1,1])

print('\nFalse Positives(FP) = ', cm[0,1])

print('\nFalse Negatives(FN) = ', cm[1,0])


# seaborn ısı haritası ile karışıklık matrisini görselleştirin

cm_matrix = pd.DataFrame(data=cm, columns=['Actual Positive:1', 'Actual
Negative:0'],
                                 index=['Predict Positive:1', 'Predict
Negative:0'])

sns.heatmap(cm_matrix, annot=True, fmt='d', cmap='YlGnBu')
```

Sınıflandırma ölçütleri

Sınıflandırma raporu, sınıflandırma modeli performansını değerlendirmenin başka bir yoludur. Model için hassasiyet, geri çağırma, f1 ve destek puanlarını görüntüler. Bu terimleri daha sonra anlattım.

```
from sklearn.metrics import classification_report

print(classification_report(y_test, y_pred))
```

Sınıflandırma doğruluğu

```
TP = cm[0,0]
TN = cm[1,1]
FP = cm[0,1]
FN = cm[1,0]

# sınıflandırma doğruluğu

classification_accuracy = (TP + TN) / float(TP + TN + FP + FN)

print('Classification accuracy : {0:0.4f}'.format(classification_accuracy))
```

Sınıflandırma hatası

```
# sınıflandırma hatası

classification_error = (FP + FN) / float(TP + TN + FP + FN)

print('Classification error : {0:0.4f}'.format(classification_error))
```

Precision

Kesinlik, tahmin edilen tüm olumlu sonuçlardan doğru şekilde tahmin edilen olumlu sonuçların yüzdesi olarak tanımlanabilir. Gerçek pozitiflerin (TP) doğru ve yanlış pozitiflerin toplamına (TP + FP) oranı olarak verilebilir.

Dolayısıyla, Kesinlik, doğru tahmin edilen olumlu sonucun oranını tanımlar. Negatif sınıftan çok pozitif sınıfla ilgilenir.

Matematiksel olarak, hassasiyet TP'nin (TP + FP) 'ye oranı olarak tanımlanabilir.

```
# precision değeri

precision = TP / float(TP + FP)


print('Precision : {0:0.4f}'.format(precision))
```

Recall

Geri çağırma, tüm gerçek olumlu sonuçlardan doğru şekilde tahmin edilen olumlu sonuçların yüzdesi olarak tanımlanabilir. Gerçek pozitiflerin (TP), gerçek pozitiflerin ve yanlış negatiflerin (TP + FN) toplamına oranı olarak verilebilir. Hatırlama aynı zamanda Hassasiyet olarak da adlandırılır.

Geri çağırma, doğru tahmin edilen gerçek pozitiflerin oranını tanımlar.

Matematiksel olarak hatırlama, TP'nin (TP + FN) 'ye oranı olarak verilebilir.

```
# recall değeri
recall = TP / float(TP + FN)

print('Recall or Sensitivity : {0:0.4f}'.format(recall))
```

True Positive Rate

```
# True Positive Rate değeri
true_positive_rate = TP / float(TP + FN)

print('True Positive Rate : {0:0.4f}'.format(true_positive_rate))
```

False Positive Rate

```
# False Positive Rate değeri
false_positive_rate = FP / float(FP + TN)

print('False Positive Rate : {0:0.4f}'.format(false_positive_rate))
```

Specificity

```
# Specificity değeri
specificity = TN / (TN + FP)

print('Specificity : {0:0.4f}'.format(specificity))
```

f1-score

f1-score, hassasiyet ve geri çağırmanın ağırlıklı harmonik ortalamasıdır. Olası en iyi f1-score 1.0 ve en kötüsü 0.0 olacaktır. f1-score, hassasiyet ve geri çağırmanın harmonik ortalamasıdır. Dolayısıyla, hesaplamalarına hassasiyet ve geri çağırma ekledikleri için f1-score her zaman doğruluk ölçümlerinden daha düşüktür. F1-score ağırlıklı ortalaması, global doğruluğu değil, sınıflandırıcı modellerini karşılaştırmak için kullanılmalıdır.

Sınıf olasılılarını hesaplayın

```
# iki sınıfın tahmin edilen ilk 10 olasılığını yazdırın - 0 ve 1
y_pred_prob = gnb.predict_proba(X_test)[0:10]

y_pred_prob
```

Gözlemler

- Her satırdaki sayıların toplamı 1'dir.
- 2 sınıfa karşılık gelen 2 sütun vardır - <= 50K ve> 50K.
- Sınıf 0 => <= 50K - Bir kişinin 50K'dan daha az yaptığı sınıf.
- Sınıf 1 =>> 50K - Bir kişinin 50K'dan fazla kazandığı sınıf.

```
# dataframe içerisinde olasılıkları saklayın

y_pred_prob_df = pd.DataFrame(data=y_pred_prob, columns=['Prob of - <=50K',
'Prob of - >50K'])

y_pred_prob_df
```

```
# 1. sınıf için tahmin edilen ilk 10 olasılığı yazdırın - Olasılık> 50K

gnb.predict_proba(X_test)[0:10, 1]

# yazı tipi boyutunu ayarla
plt.rcParams['font.size'] = 12

# 10 bölmeli grafik histogramı
plt.hist(y_pred1, bins = 10)

# tahmin edilen olasılıkların başlığını belirleyin
plt.title('Maaşların tahmin edilen olasılıklarının histogramı> 50K')

# x ekseni sınırını ayarla
plt.xlim(0,1)

plt.xlabel('Maaşların tahmini olasılıkları> 50K')
plt.ylabel('Sıklık')
```

ROC - AUC

ROC Curve

Sınıflandırma modeli performansını görsel olarak ölçmek için bir başka araç da ROC Eğrisi'dir. ROC Eğrisi, Alıcı Çalışma Karakteristik Eğrisi anlamına gelir. Bir ROC Eğrisi, çeşitli sınıflandırma eşik seviyelerinde bir sınıflandırma modelinin performansını gösteren bir grafiktir.

ROC Eğrisi, çeşitli eşik seviyelerinde Yanlış Pozitif Orana (FPR) karşı Gerçek Pozitif Oranı (TPR) çizer.

Gerçek Pozitif Oran (TPR) ayrıca Geri Çağırma olarak da adlandırılır. TP'nin (TP + FN) 'ye oranı olarak tanımlanır.

Yanlış Pozitif Oran (FPR), FP'nin (FP + TN) 'ye oranı olarak tanımlanır.

ROC Eğrisinde, tek bir noktanın TPR (Gerçek Pozitif Oran) ve FPR (Yanlış Pozitif Oran) üzerine odaklanacağız. Bu bize çeşitli eşik seviyelerinde TPR ve FPR'den oluşan ROC eğrisinin genel performansını verecektir. Dolayısıyla, bir ROC Eğrisi, TPR ile FPR'yi farklı sınıflandırma eşik seviyelerinde çizer. Eşik seviyelerini düşürürsek, daha fazla maddenin pozitif olarak sınıflandırılmasına neden olabilir. Hem Gerçek Pozitifleri (TP) hem de Yanlış Pozitifleri (FP) artıracaktır.

```
# plot ROC Curve

from sklearn.metrics import roc_curve

fpr, tpr, thresholds = roc_curve(y_test, y_pred1, pos_label = '>50K')

plt.figure(figsize=(6,4))

plt.plot(fpr, tpr, linewidth=2)

plt.plot([0,1], [0,1], 'k--' )

plt.rcParams['font.size'] = 12

plt.title('Maaşları Tahmin Etmek için Gauss Naive Bayes Sınıflandırıcısı için ROC eğrisi')

plt.xlabel('False Positive Rate (1 - Specificity)')

plt.ylabel('True Positive Rate (Sensitivity)')

plt.show()
```

ROC AUC

ROC AUC, Alıcı Çalışma Karakteristiği - Eğri Altındaki Alan anlamına gelir. Sınıflandırıcı performansını karşılaştırmak için bir tekniktir. Bu teknikte, eğrinin altındaki alanı (AUC) ölçüyoruz. Mükemmel bir sınıflandırıcı, 1'e eşit bir ROC AUC'ye sahip olurken, tamamen rastgele bir sınıflandırıcı, 0,5'e eşit bir ROC AUC'ye sahip olacaktır.

Dolayısıyla, ROC AUC, eğrinin altındaki ROC grafiğinin yüzdesidir.

```
# ROC AUC'yi hesapla

from sklearn.metrics import roc_auc_score

ROC_AUC = roc_auc_score(y_test, y_pred1)

print('ROC AUC : {:.4f}'.format(ROC_AUC))
```

Gözlemler

- ROC AUC, sınıflandırıcı performansının tek numaralı bir özetidir. Değer ne kadar yüksekse sınıflandırıcı o kadar iyidir.
- Modelimizin ROC AUC'si 1'e yaklaşıyor. Dolayısıyla, sınıflandırıcımızın yarın yağmur yağıp yağmayacağını tahmin etmede iyi bir iş çıkardığı sonucuna varabiliriz.

```
# çapraz doğrulanmış ROC AUC'yi hesaplayın

from sklearn.model_selection import cross_val_score

Cross_validated_ROC_AUC = cross_val_score(gnb, X_train, y_train, cv=5,
scoring='roc_auc').mean()

print('Cross validated ROC AUC : {:.4f}'.format(Cross_validated_ROC_AUC))
```

k-Fold Cross Validation

```
#  10-Fold Cross Validation uygulanıyor

from sklearn.model_selection import cross_val_score

scores = cross_val_score(gnb, X_train, y_train, cv = 10, scoring='accuracy')

print('Cross-validation scores:{}'.format(scores))
```

Çapraz doğrulama doğruluğunu ortalamasını hesaplayarak özetleyebiliriz.

```
# Ortalama çapraz doğrulama puanını hesapla

print('Average cross-validation score: {:.4f}'.format(scores.mean()))
```

Gözlemler

- Ortalama çapraz doğrulamayı kullanarak, modelin ortalama olarak% 80.63 civarında doğru olmasını beklediğimiz sonucuna varabiliriz.
- 10 kat çapraz doğrulama ile üretilen 10 puana bakarsak, kıvrımlar arasındaki doğrulukta% 81,35 ila% 79,64 doğruluk arasında nispeten küçük bir varyans olduğu sonucuna varabiliriz. Dolayısıyla, modelin eğitim için kullanılan belirli kıvrımlardan bağımsız olduğu sonucuna varabiliriz.
- Orijinal model doğruluğumuz 0,8083'tür, ancak ortalama çapraz doğrulama doğruluğu 0,8063'tür. Dolayısıyla, 10 kat çapraz doğrulama doğruluğu, bu model için performans artışı sağlamaz.