BURSA ULUDAĞ ÜNİVERSİTESİ BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

2024 – 2025 Eğitim Öğretim Yılı Güz Yarıyılı

Yapay Zeka

Ödev 3

032190007- Adnan Topçu

Toy Naive Bayes sınıflandırıcıyı nasıl tasarladınız?

1. Öncül Olasılıkların Hesaplanması:

- Eğitim verisinde her harfin frekansını hesaplayarak, her harfin veri içindeki öncül olasılığı elde ettim.

2. Koşullu Olasılıkların Hesaplanması:

- Ayrık özellikler için her harf ve her özellik değerinin frekansına dayalı koşullu olasılıklar hesaplandı. Bu işlemde Laplace düzeltmesi uyguladım.
- Sürekli özellikler için her harf ve her özellik değerinin ortalama ve standart sapması hesapladım. Bu parametreler Gaussian dağılımında kullanıldı.

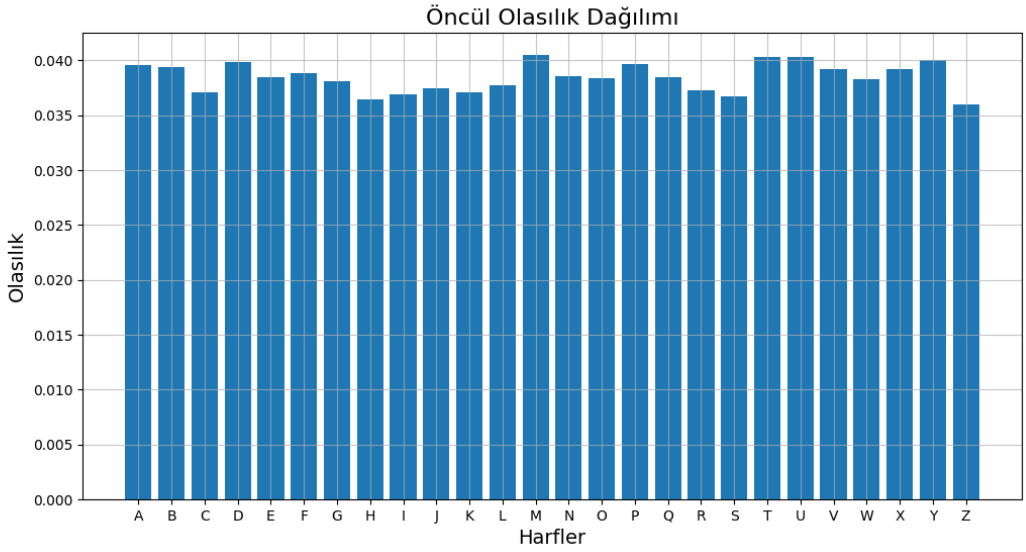
3. Tahmin:

- Test verisindeki her bir örnek için, tüm harflerin logaritmik toplam olasılıkları hesaplandı. Maksimum olasılığı veren harf, sınıf etiketi olarak atandı.

4. Başarı Oranı:

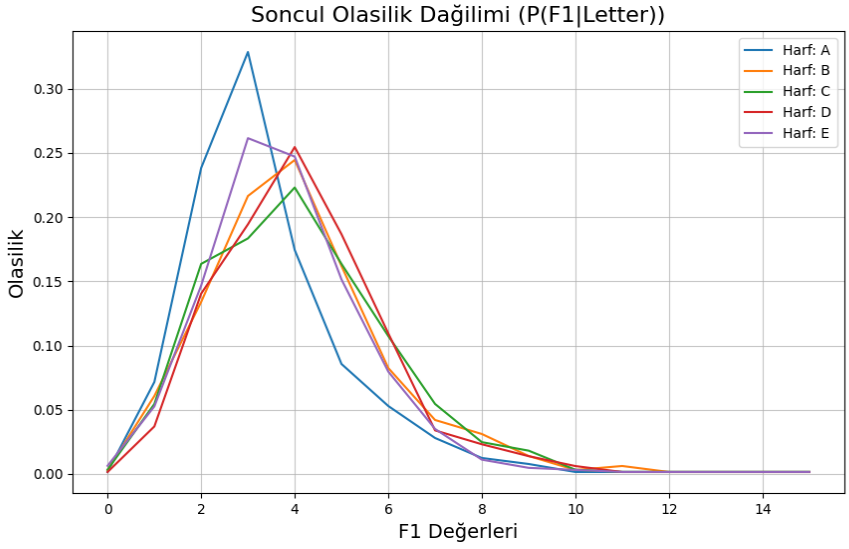
- Tahmin edilen sınıf etiketleriyle gerçek sınıf etiketleri karşılaştırılarak, doğruluk oranı hesaplandım.

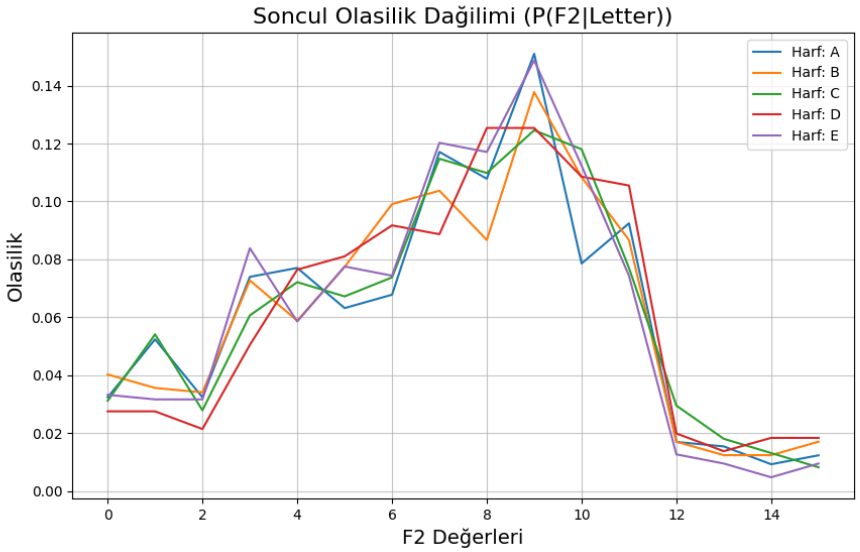
Tüm Harflerin Öncül Olasılık Dağılımı

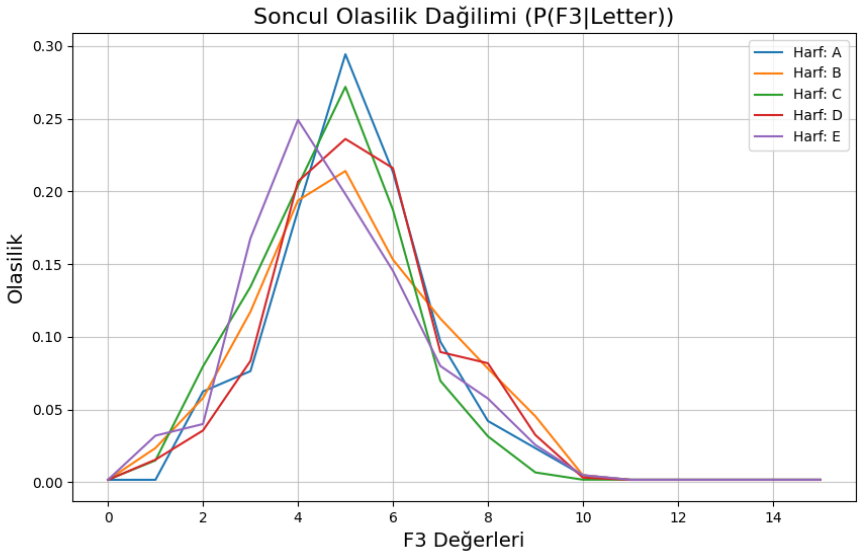


Soncul Olasılık Dağılımı

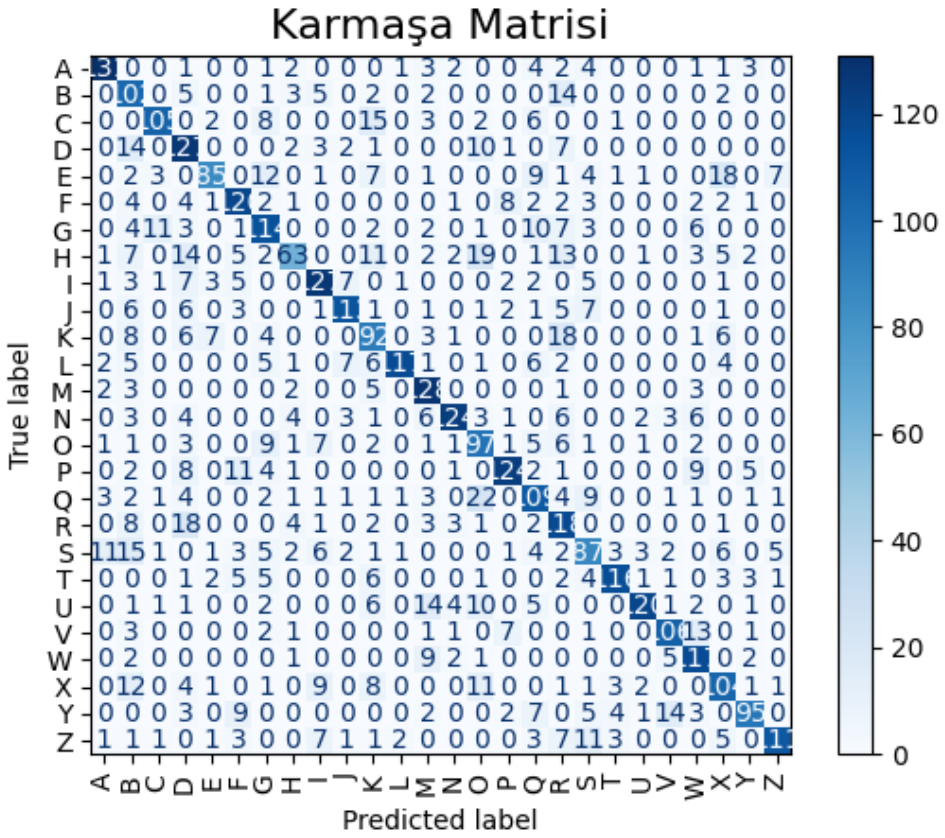
5 farklı harf (A, B, C, D, E) ve 3 farklı özellik (F1, F2, F3) için soncul olasılık dağılımını aynı grafikte gösterimi aşağıdaki grafiklerde gösterilmiştir:




Soncul Olasilik Dağilimi (P(F2|Letter))
Harf: A
Harf: B
Harf: C
Harf: D
Harf: E
Olasilik
F2 Değerleri
0.00
0.02
0.04
0.06
0.08
0.10
0.12
0.14
0
2
4
6
8
10
12
14


Soncul Olasilik Dağilimi (P(F3|Letter))
Harf: A
Harf: B
Harf: C
Harf: D
Harf: E
Olasilik
F3 Değerleri
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0
2
4
6
8
10
12
14

Karmaşa Matrisi



[A]-[A] →113

[B]-[B] →102

[C]-[C] →105

[D]-[D] →127

[E]-[E] →85

[F]-[F] →120

[G]-[G] →114

[H]-[H] →63

[I]-[I] →127

[J]-[J] →113

[K]-[K] →92

[L]-[L] →117

[M]-[M] →128

[N]-[N] →124

[O]-[O] →97

[P]-[P] →124

[Q]-[Q] →109

[R]-[R] →118

[S]-[S] →87

[T]-[T] →116

[U]-[U] →120

[V]-[V] →106

[W]-[W] →117

[X]-[X] →104

[Y]-[Y] →95

[Z]-[Z] →111

=2834

2834/4000 =0.7085

Kod:

```
import numpy as np
import pandas as pd
from collections import defaultdict
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix,
ConfusionMatrixDisplay
from math import log, exp, sqrt, pi

# Veri yükleme ve ayrıştırma
letters = pd.read_csv('letter_recognition.data')
train = letters.head(16000)
test = letters.tail(4000)

# Ön bilgiler
letter_counts =
train.groupby('Letter')['Letter'].count()
total_letters = letter_counts.sum()
prior_probs = letter_counts / total_letters
```

```
# Ayrık özellikler için koşullu olasılık hesaplama
def calculate_discrete_conditional_probs(data, features):
    conditional_probs = defaultdict(lambda: defaultdict(lambda: defaultdict(float)))
    for letter, group in data.groupby('Letter'):
        for feature in features:
            counts = group[feature].value_counts()
            total = counts.sum()
            unique_values = len(counts)
            for value in range(16):
                conditional_probs[letter][feature][value] = \
                    (counts.get(value, 0) + 1) / (total + unique_values)
    return conditional_probs

# Sürekli özellikler için ortalama ve standart sapma hesaplama
def calculate_continuous_conditional_params(data, features):
```

```
    conditional_params = defaultdict(lambda:
defaultdict(tuple))
    for letter, group in data.groupby('Letter'):
        for feature in features:
            mean = group[feature].mean()
            std = group[feature].std(ddof=1)
            conditional_params[letter][feature] = (mean,
std)
    return conditional_params

# Gaussian olasılık hesaplama
def gaussian_prob(x, mean, std):
    if std == 0:
        std = 1e-6
    return (1 / (sqrt(2 * pi) * std)) * exp(-((x - mean) ** 2)
/ (2 * std ** 2))

# Özellikleri belirle
features = train.columns[1:]
```

```
# Koşullu olasılıkları hesapla
discrete_conditional_probs = calculate_discrete_conditional_probs(train, features)
continuous_conditional_params = calculate_continuous_conditional_params(train, features)

# Tahmin ve başarı oranı
def Degerlendirme():
    correct_predictions = 0
    predicted_labels = []
    for index, row in test.iterrows():
        max_log_prob = -np.inf
        predicted_letter = None
        for letter in prior_probs.index:
            log_prob = log(prior_probs[letter])
            for feature in features:
                value = row[feature]
```

```
            if value in
discrete_conditional_probs[letter][feature]:
                log_prob +=
log(discrete_conditional_probs[letter][feature][value
])
            mean, std =
continuous_conditional_params[letter][feature]
            log_prob += log(gaussian_prob(value, mean,
std))
        if log_prob > max_log_prob:
            max_log_prob = log_prob
            predicted_letter = letter
    predicted_labels.append(predicted_letter)
    if predicted_letter == row['Letter']:
        correct_predictions += 1
accuracy = (correct_predictions / len(test)) * 100
print(f"Modelin başari orani: {accuracy:.2f}%")
return predicted_labels

# Karmaşa matrisi oluşturma ve görselleştirme
```

```
def karmasa_matrisi():
    true_labels = test['Letter'].values
    predicted_labels = Degerlendirme()
    cm = confusion_matrix(true_labels,
predicted_labels, labels=prior_probs.index)
    disp =
ConfusionMatrixDisplay(confusion_matrix=cm,
display_labels=prior_probs.index)
    disp.plot(cmap='Blues', xticks_rotation='vertical')
    plt.title('Karmaşa Matrisi', fontsize=16)
    plt.show()
    accuracy = np.trace(cm) / np.sum(cm) * 100
    print(f"Siniflandiricinin tahmin başarisi:
{accuracy:.2f}%")

# Olasılık dağılımı grafikleri
def ornek_soncul_olasilik():
    selected_letters = ['A', 'B', 'C', 'D', 'E']
    selected_features = ['F1', 'F2', 'F3']
    for feature in selected_features:
```

```
    plt.figure(figsize=(10, 6))
    for letter in selected_letters:
        values = [discrete_conditional_probs[letter][feature][v] for v in range(16)]
        plt.plot(range(16), values, label=f'Harf: {letter}')
    plt.title(f'Soncul Olasilik Dağilimi (P({feature}|Letter))', fontsize=16)
    plt.xlabel(f'{feature} Değerleri', fontsize=14)
    plt.ylabel('Olasilik', fontsize=14)
    plt.legend()
    plt.grid(alpha=0.7)
    plt.show()


def tum_harfler_oncul_olasilik():
  plt.figure(figsize=(12, 6))
  plt.bar(prior_probs.index, prior_probs.values)  # Harfler ve olasılıkları
  plt.title('Öncül Olasılık Dağılımı', fontsize=16)  # Başlık
```

```
    plt.xlabel('Harfler', fontsize=14)  # X ekseni
    plt.ylabel('Olasılık', fontsize=14)  # Y ekseni
    plt.grid(alpha=0.7)
    plt.show()

# Arayüz
def menu():
    while True:
        print("\nToy Naive Bayes Siniflandirici")
        print("1. Modeli Eğit ve Başari Oranini Hesapla")
        print("1. Tüm Harflerin Öncül Olasılık Dağılımı")
        print("3. Karmaşa Matrisini Göster")
        print("4. Olasilik Dağilimi Grafiklerini Göster")
        print("5. Çikiş")
        choice = input("Seçiminizi yapin: ")

        if choice == '1':
            Degerlendirme()
        elif choice == '2':
```

```
            tum_harfler_oncul_olasilik()
        elif choice == '3':
            karmasa_matrisi()
        elif choice == '4':
            ornek_soncul_olasilik()
        elif choice == '5':
            print("Çikiliyor...")
            break
        else:
            print("Geçersiz seçim, tekrar deneyin.")

menu()
```