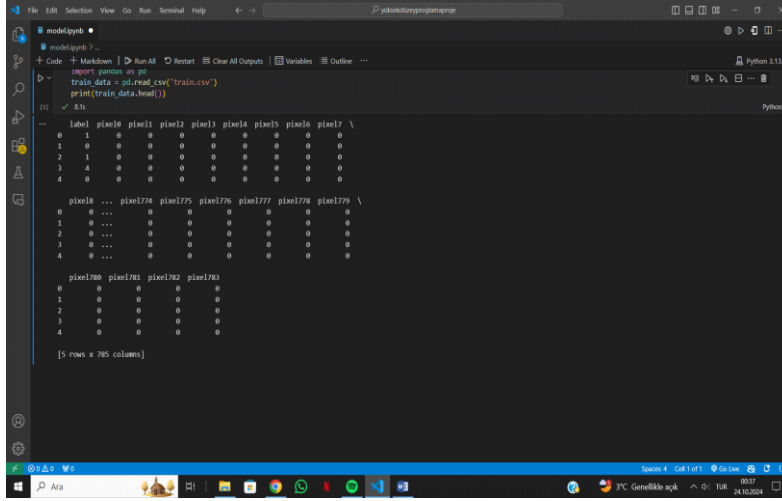


**YÜKSEK DÜZEY PROGRAMLAMA
PROJE ÖDEVİ**

**FUNDA TAŞDEMİR
202113172015**



```
File Edit Selection View Go Run Terminal Help
modelgryb
+ Code + Markdown | Run All | Restart | Clear All Outputs | Variables | Outline
Python 3.11.0
In [1]:
import pandas as pd
train_data = pd.read_csv('train.csv')
print(train_data.head())
Out[1]:
label pixel0 pixel1 pixel2 pixel3 pixel4 pixel5 pixel6 pixel7 \
0 1 0 0 0 0 0 0 0 0 0
1 0 0 0 0 0 0 0 0 0 0
2 1 0 0 0 0 0 0 0 0 0
3 4 0 0 0 0 0 0 0 0 0
4 0 0 0 0 0 0 0 0 0 0
...
pixel774 pixel775 pixel776 pixel777 pixel778 pixel779 \
0 0 ... 0 0 0 0 0 0
1 0 ... 0 0 0 0 0 0
2 0 ... 0 0 0 0 0 0
3 0 ... 0 0 0 0 0 0
4 0 ... 0 0 0 0 0 0
...
pixel780 pixel781 pixel782 pixel783
0 0 0 0 0
1 0 0 0 0
2 0 0 0 0
3 0 0 0 0
4 0 0 0 0
[5 rows x 785 columns]
```

label: Her satırda, o satırdaki el yazısı rakamının ne olduğunu gösterir.

pixel0, pixel1, ... , pixel784: 0'dan 784'e kadar olan sütunlar, el yazısı rakamlarının piksellerini gösterir.

Veri Setinin Özellikleri

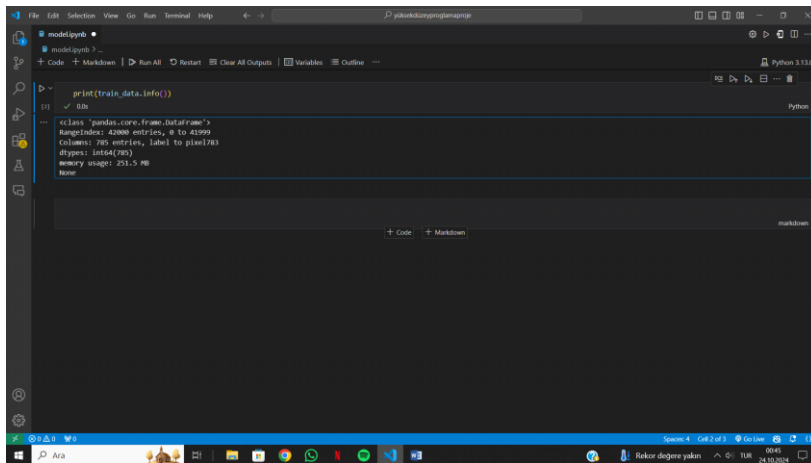
Toplam Giriş Sayısı: 42,000 örnek

Toplam Özellik Sayısı: 785 sütun

Veri Türü: Tüm sütunlar `int_64` türünde, bu tam sayı olarak saklandığını belirtir.

Sonuçların Anlamı

El yazısı rakamları tanımak için kullanılan bir veri setidir. 0'dan 9'a kadar olan rakamların el yazısı örnekleri bulunuyor.



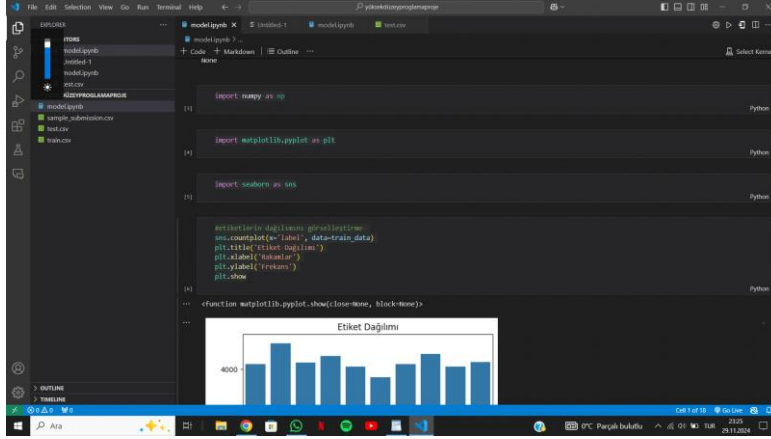
```
File Edit Selection View Go Run Terminal Help
modelgryb
+ Code + Markdown | Run All | Restart | Clear All Outputs | Variables | Outline
Python 3.11.0
In [1]:
print(train_data.info())
Out[1]:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 42000 entries, 0 to 42000
Columns: 785 entries, label to pixel783
dtypes: int64(785)
memory usage: 253.5 MB
None
```

Gerekli Kütüphanelerin İçe Aktarılması

numpy: Sayısal işlemler için kullanılır.

matplotlib.pyplot: Verileri görselleştirmek için kullanılır.

seaborn: Daha estetik görselleştirme araçları sağlar.

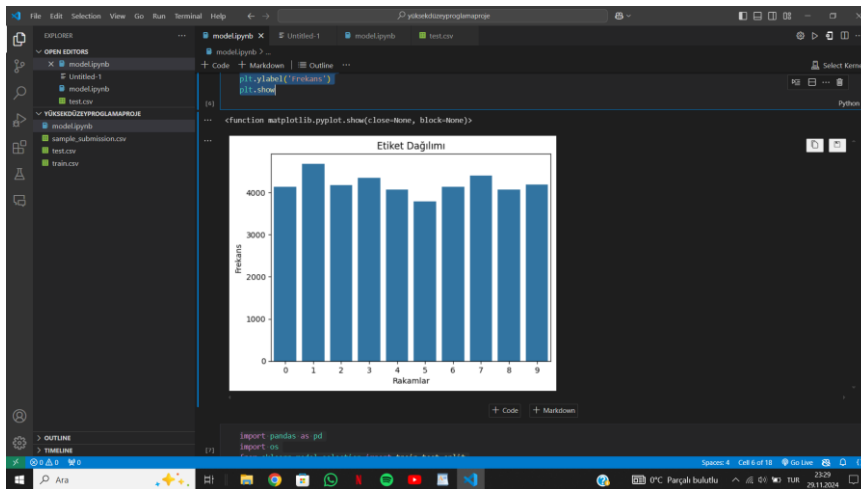


Etiketlerin Dağılımını Görselleştirme

Bu sütun, veri setindeki sınıfları gösterir.

Grafikteki yatay eksen, label sütununu temsil ediyor.

data=train_data: Grafikte kullanılacak veri.



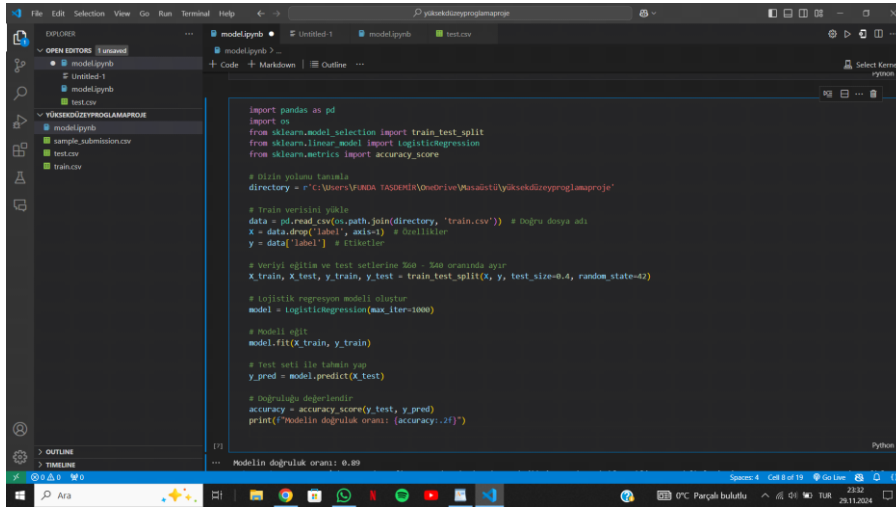
Lojistik Regresyon ile Sınıflandırma

Lojistik regresyon modeli kullanılarak veri setindeki etiketlerin sınıflandırılması beklenmektedir.

Sonuçlar:

Model Eğitimi: Eğitim seti kullanılarak lojistik regresyon modeli eğitildi.

Doğruluk Oranı: Test setinde modelin doğruluk oranı %0.89 olarak hesaplandı.



```
import pandas as pd
import os
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

# Dizin yolunu tanımla
directory = r"C:\Users\FUNDA TASDEMİR\OneDrive\Masaini\YüksekDereyProgramProje"

# Train verisini yükle
data = pd.read_csv(os.path.join(directory, 'train.csv')) # Doğru deşya adı
X = data.drop('label', axis=1) # Özellikler
y = data['label'] # Etiketler

# Veriyi eğitim ve test setlerine 80% - 20% oranında ayır
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Lojistik regresyon modeli oluştur
model = LogisticRegression(max_iter=1000)

# Modeli eğit
model.fit(X_train, y_train)

# Test seti ile tahmin yap
y_pred = model.predict(X_test)

# Doğruluk değeri
accuracy = accuracy_score(y_test, y_pred)
print("Modelin doğruluk oranı: (accuracy:.2f)")
```

Modelin doğruluk oranı: 0.89

```
... Modelin doğruluk oranı: 0.89
C:\Users\FUNDA TASDEMİR\AppData\Local\Programs\Python\Python313\Lib\site-packages\sklearn\linear_model\_logistic.py:469: ConvergenceWarning: lbfgs
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
n_iter_i = _check_optimize_result(
```

Karışıklık Matrisi ile Performans Analizi

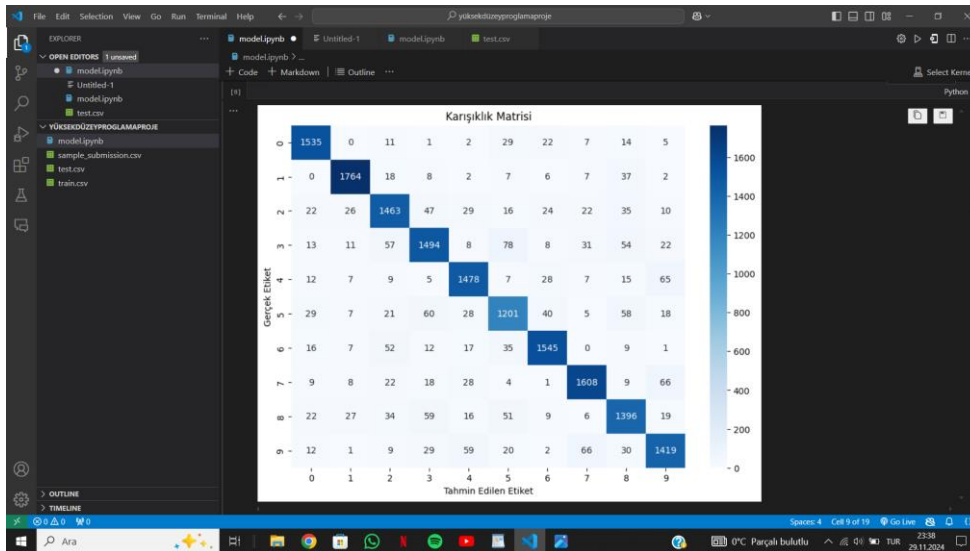
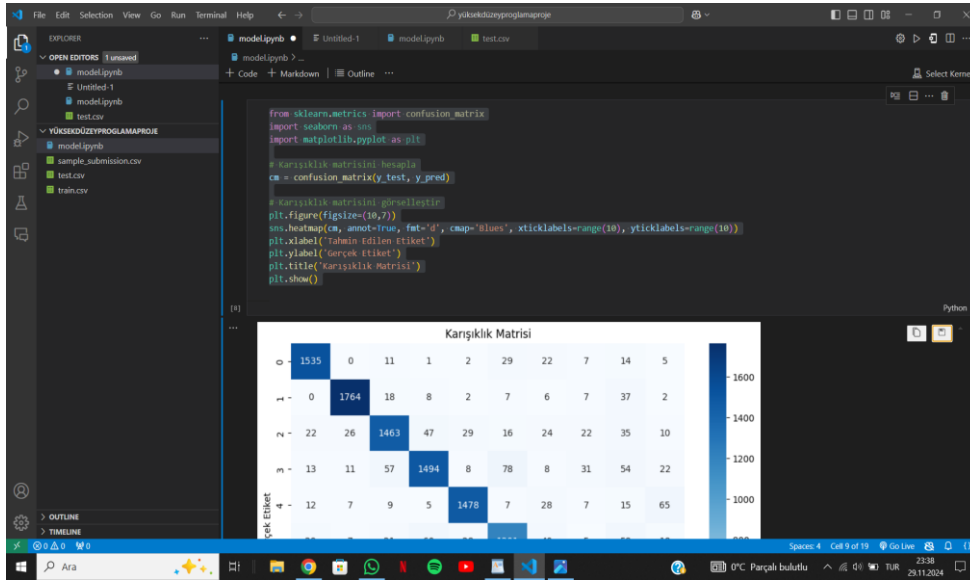
Modelin tahmin performansını analiz etmek için karışıklık matrisi kullanıldı.

Sonuçlar:

Karışıklık matrisi, bir ısı haritası olarak görselleştirildi.

Hücrelerdeki yüksek değerler, modelin doğru tahminler yaptığını gösterir.

Dışındaki hücrelerdeki değerler, modelin yaptığı hataları gösterir.

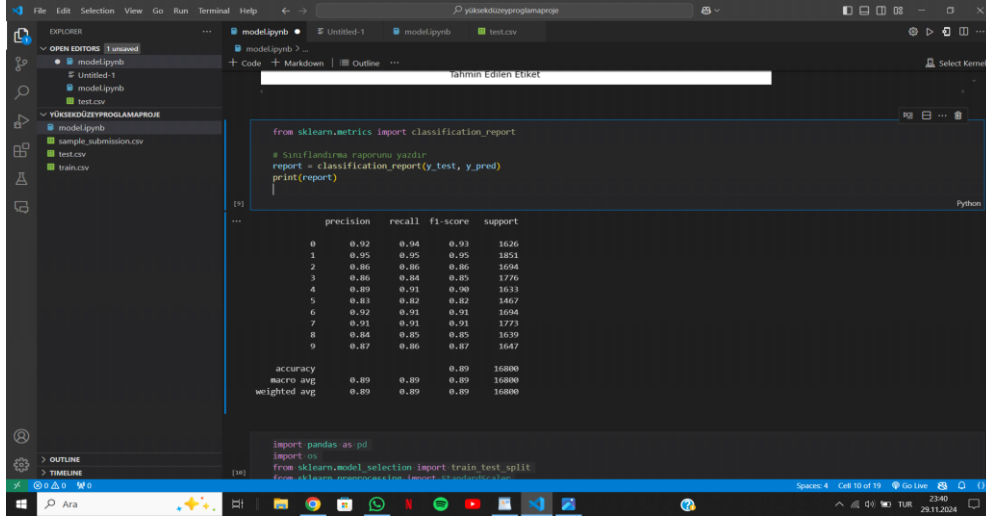


Sınıflandırma

Modelin her sınıf için doğruluk, geri çağırma, ve F1 skorlarını analiz etmek istenildi.

Precision ve Recall: Değerlerin yüksek olduğu sınıflar modelin bu sınıflarda güçlü olduğunu gösterir.

Düşük Performanslı Sınıflar: Belirli bir sınıfta düşük bir F1 skoru varsa, bu sınıfta modelin zayıf olduğunu gösterir.



```
from sklearn.metrics import classification_report

# Sınıflandırma raporunu yazdır
report = classification_report(y_test, y_pred)
print(report)
```

	precision	recall	f1-score	support
0	0.92	0.94	0.93	1626
1	0.95	0.95	0.95	1851
2	0.86	0.86	0.86	1694
3	0.86	0.84	0.85	1776
4	0.89	0.91	0.90	1611
5	0.83	0.82	0.82	1667
6	0.92	0.91	0.91	1694
7	0.91	0.91	0.91	1773
8	0.84	0.85	0.85	1638
9	0.87	0.86	0.87	1647
accuracy			0.89	16800
macro avg	0.89	0.89	0.89	16800
weighted avg	0.89	0.89	0.89	16800

```
import pandas as pd
import os
from sklearn.model_selection import train_test_split
```

KNN Algoritması ile Sınıflandırma

K-Nearest Neighbors algoritmasını kullanarak rakamları sınıflandırmak ve model performansını değerlendirmek amacıyla kullanıldı.

Doğruluk Oranı: KNN algoritmasının doğruluk oranı:0.93

```
File Edit Selection View Go Run Terminal Help
yükseközyazilgiprogramproje

EXPLORER
model.ipynb
test.csv

model.ipynb
test.csv
sample_submission.csv
train.csv

YÜKSEKÖZYAZILGIPROGRAMPROJE
model.ipynb
sample_submission.csv
test.csv
train.csv

model.ipynb
+ Code + Markdown | Outline
# Verisi eğitim ve test setlerine 360 - 340 oranında ayır
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, random_state=42)

# Özellikleri normalize et
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Knn modeli oluştur
knn_model = KNeighborsClassifier(n_neighbors=3) # Komşu sayısını 3 olarak ayarladık

# Modeli eğitim verisi ile eğit
knn_model.fit(X_train, y_train)

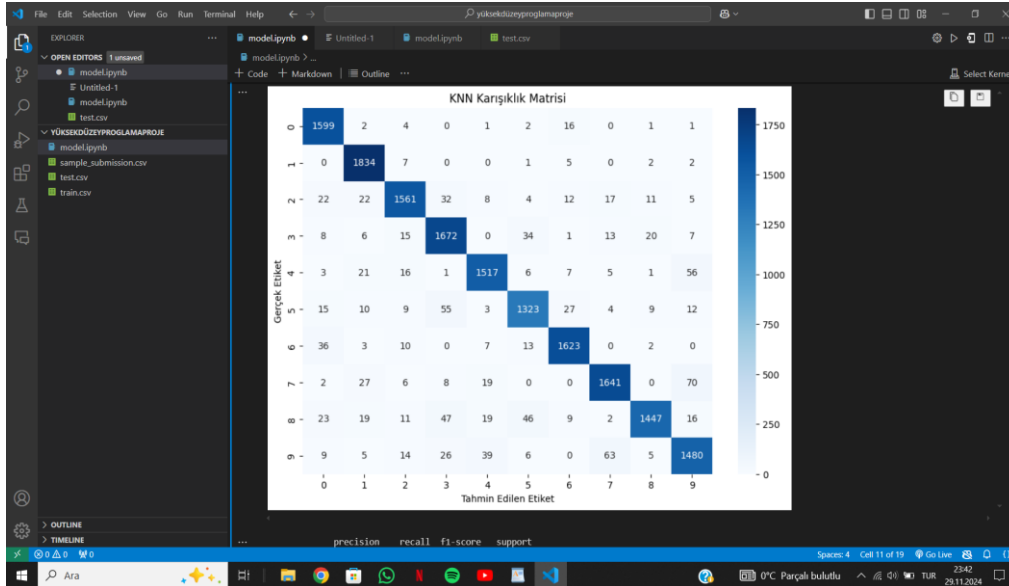
# Test seti ile tahmin yap
knn_predictions = knn_model.predict(X_test)

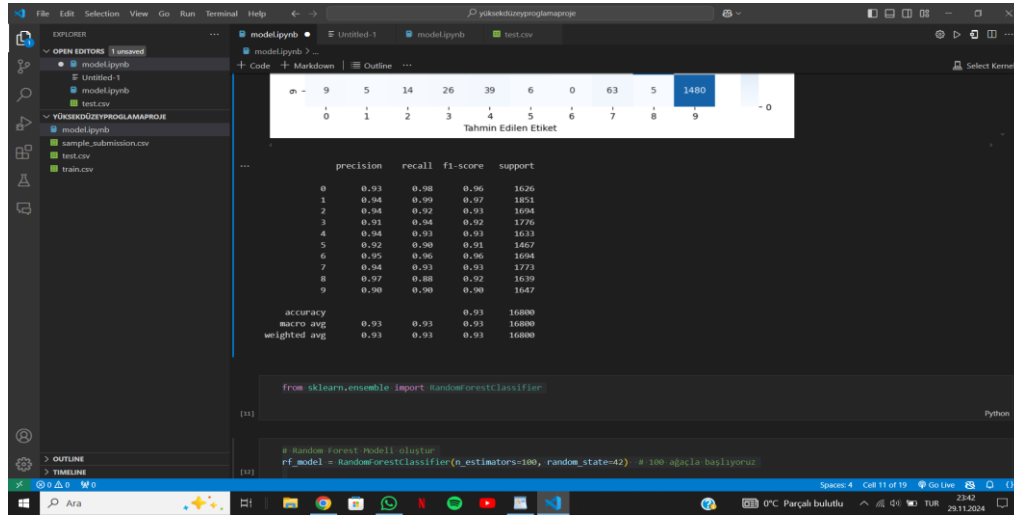
# Doğruluk değeri hesapla
knn_accuracy = accuracy_score(y_test, knn_predictions)
print(f"Knn Modelinin doğruluk oranı: {knn_accuracy:.2f}")

# Karışıklık matrisini hesapla
knn_cm = confusion_matrix(y_test, knn_predictions)

# Karışıklık matrisini görselleştir
plt.figure(figsize=(10,7))
sns.heatmap(knn_cm, annot=True, fmt='d', cmap='Blues', xticklabels=range(10), yticklabels=range(10))
plt.xlabel('Tahmin Edilen Etiket')
plt.ylabel('Gerçek Etiket')
plt.title('KNN Karışıklık Matrisi')
plt.show()

# Sınıflandırma raporunu yazdır
knn_report = classification_report(y_test, knn_predictions)
print(knn_report)
```





Random Forest Modeli ile Çalışma

Random Forest algoritması, birden fazla karar ağacını bir araya getirerek sınıflandırma yapar. Bu model, genelde daha basit algoritmalara kıyasla daha yüksek doğruluk sağlar.

Kullanılan parametreler:

`n_estimators=100`: Model, 100 adet karar ağacı kullanılarak eğitildi.

`random_state=42`: Rastgelelik kontrolü sağlamak için belirlendi.

Doğruluk Oranı:

Test verisinde modelin doğruluk oranı %93

Precision, Recall ve F1-score değerleri dengelidir, böylece modelin her sınıfta benzer performans gösterdiğini gösterir.

