

Son Teslim tarihi: 05.01.2023 23:59

Uzman Sistemlere Giriş Projesi

OĞUZHAN TOPALOĞLU
Ç19052025 – Grup 1

*Bilgisayar Mühendisliği Bölümü,
Elektrik-Elektronik Fakültesi,
Yıldız Teknik Üniversitesi*



Istanbul, 2023

İÇERİK

1. Genel Bilgiler
2. Geliştirme Sürecinde Yaşananlar
3. Menü Fonksiyonlarının Çalıştırılması
4. Sistemin Sayısal Başarısı
5. Kaynaklar
6. Youtube Videosu
7. Program Dosyalarının Listesi

1. Genel Bilgiler

Bu projede vatan bilgisayardan telefonlarla ilgili bilgileri web scraping yaparak okuyan, Random Forest tekniğini ve bu verileri kullanarak bir model oluşturan ve bu modeli kullanarak özellikleri verilen bir telefonun fiyatını tahmin eden bir uzman sistem geliştirilmiştir.

Projede görsel arayüz yerine terminal ve dil olarak Python kullanılmıştır. Kütüphane olarak veritabanı oluşturmada pandas, model oluşturmada sklearn, web scraping'de BeautifulSoup4 (bs4) ve requests (HTTP kütüphanesi) kullanılmıştır.

Proje 4 adet dosyadan oluşmaktadır: main.py, menu.py, scrape.py, expert.py.

main.py dosyası, içinde terminalde yazı yazdıran print işlemlerini ve girdilere göre yapılacak işlemleri içerir. Menüleri yazdırırken main dosyasının çok kalabalık olmamasını sağlama amacıyla veri tutarlılığı kontrollerini yapan kodlar menu.py dosyasına konmuştur.

scrape.py dosyası, Vatan Bilgisayar'dan telefonlarla ilgili verileri çekip list[dict[str:str]] olarak dönen fonksiyonları içerir.

expert.py dosyası da içinde scrape.py'da elde edilen listeleri alıp veritabanı oluşturup, preprocessing yapıp (gelen datanın gereksiz kısımlarını atma, veri türlerini tutarlı yapma vb.), model oluşturup, bu model ile fiyat tahmini yapan fonksiyonları barındırır.

2. Geliştirme Sürecinde Yaşananlar


Proje sırasında en çok preprocessing kısmında zorlandım ve uzman sistemlerin geliştirilmesi sürecinde bilginin depolanmasının (bilgi mühendisi olmanın) ne kadar zor bir şey olduğunu fark ettim. Elde ettiğimiz veriler genelde yanlış veri türlerinde olabiliyor ve bu da RandomForestClassifier gibi sınıflarda hata almamıza neden oluyor. Ayrıca çektiğimiz verilerde bir sürü null oluşabiliyor ve bu da modelin düzgün çalışmamasına neden oluyor.

Projenin başlarında Vatan Bilgisayar'dan her veriyi çekiyor ve veritabanıma koyuyordum. Ancak sonradan fark ettim ki, bu çektiğim verilerin çoğunluğu null'lardan oluşuyor. Bunun nedeni her telefon sayfasında farklı bilgilerin verilebiliyor olmasıydı. Bazıları ayrıntılı bir şekilde bilgi tablolarını dolduruyorken bazıları yalnızca isim, fiyat, boyut, ağırlık, menşei gibi bilgileri içeriyordu. Ben de bu null durumunun önüne geçmek için en az null içeren sütunları yazdıracak şu kodu yazdım:

```
16
17 if __name__ == '__main__':
18     all_phone_urls = scrape.get_all_phone_urls()
19     data_as_list = scrape.get_data_as_list(all_phone_urls)
20     df = preprocess.get_df(data_as_list)
21
22     for col in df.columns:
23         print(f"{col} has {df[col].isna().sum()} null values")
24
25     print(df.shape)
26
```

Bu kod ile aldığım çıktıları da terminalden kopyala yapıştır ile bir txt'ye koydum ve az satır var diye manüel olarak null sayılarını kontrol ettim.

Bu txt dosyası şöyle gözüküyordu:



```
preprocessing - Notepad
File Edit Format View Help

Batarya Kapasitesi Aralığı has 145 null values
Arka Kamera Sayısı has 133 null values
Kulaklık Girişi has 133 null values
Ana Kamera Çözünürlük Ara has 133 null values
Ön Kamera Çözünürlük Aral has 135 null values
Ön Kamera Sayısı has 133 null values
Şarj Girişi has 133 null values
Otomatik Odaklama has 105 null values
Arka kamera tipi has 95 null values
Ağır Çekim Video Kaydı has 105 null values
Video Kayıt Çözünürlüğü has 105 null values
Kamera Çözünürlük has 105 null values
Digital Zoom has 105 null values
Kamera Çözünürlük Aralığı has 105 null values
Ön Kamera Çözünürlüğü has 105 null values
Ön Kamera Çöz. Aralığı has 105 null values
NFC has 128 null values
Sim Kart Yeterliliği has 102 null values
Wi-Fi standartları has 80 null values
Şebeke Frekansı has 105 null values
Mobil Veri Hızı has 131 null values
Pil Gücü has 105 null values
Kablosuz Şarj has 105 null values
Pil/Batarya Gücü has 105 null values
Parmak İzi Okuyucu has 105 null values
Yüz Tanıma has 105 null values
Suya Dayanıklılık has 121 null values
Max. Hafıza Kartı Boyutu has 105 null values
Desteklenen Hafıza Kartı has 185 null values
Mobil Ram Boyutu (GB) has 105 null values
Dahili Hafıza has 116 null values
İşletim Sistemi Türü has 105 null values
İşletim Sistemi has 121 null values
```

Gittim ve dediğim gibi manüel olarak aşağıdaki görseldeki az null olan sütunları belirledim:

Product Name has 0 null values
Price has 0 null values
Menşei has 0 null values
Garanti has 0 null values
Genişlik has 19 null values
Ağırlık has 15 null values
Yükseklik has 20 null values

Buradan da 5 tane ek bilginin olacağı ortaya çıkmış oldu, bunlar menşei, garanti, genişlik, ağırlık ve yükseklik idi.

Ardından verileri okudum, preprocessing işlemlerini yaptım ve veritabanını iki parçaya ayırdım:

```
def get_model() -> RandomForestClassifier:  
    # Web scraping part  
    all_phone_urls = scrape.get_all_phone_urls()  
    data_as_list = scrape.get_data_as_list(all_phone_urls)  
    # Data preprocessing part  
    df = __preprocess_data(data_as_list)  
    model_df, gui_df = __split_df(df, 25)  
    # Modeling part  
    model, score = __create_model(model_df)  
    return model, gui_df, score
```

Konu belirleme mailinde bana şunu demiştiniz:

FA

fatih amasyali <mfatihamasyali@gmail.com>
Kime: OĞUZHAN TOPALOĞLU

21.12.2022 Çar 14:41

bu uygundur,
ama önerilerin doğruluğunu kontrol ederken en az 20 telefon kullanmalısın.

OĞUZHAN TOPALOĞLU <oguzhan.topaloglu@std.yildiz.edu.tr>, 21 Ara 2022 Çar, 01:59 tarihinde şunu yazdı:

Yeni öneri:

Sklearn vb. veri madenciliği kütüphanelerini kullanarak verilen donanımına göre bir telefonun fiyatını tahmin eden uzman sistem. Sistemi hazırlamadan önce vatan bilgisayar gibi bir siteden web scraping ile veri çekeceğim ve telefonları, donanımları ve markaları gibi bilgilerle depolayacağım. Bunun üstüne bir model train edeceğim. GUI'de önceden belirli (modelin training'ine katılmamış) birkaç telefon olacak. Model bu telefonların özelliklerini kullanarak bir fiyat tahmin edecek. Daha sonrasında da gerçek fiyata bakarak ne kadar doğru bir cevap bulunduğunu kontrol edeceğim. Böylece önerilerin doğruluğunu da ölçmüş olacağım.

Bu uygun mudur hocam?

...

Bu yüzden veritabanını iki parçaya ayırırken bir veritabanına 25 (en az 20 dediniz, ben 25 seçtim), öbür veritabanına da geriye kalan satırları koydum.

Burada model oluşturma kısmını anlatmadan önce preprocessing’de yaptıklarımı da kısaca değinmek istiyorum. `expert.__preprocess_data()` fonksiyonunda öncelikle listemi dataframe’e dönüştürüyorum ve Türkçe sütun isimlerini İngilizce’ye çeviriyorum:

```
def __preprocess_data(data_as_list) -> pd.DataFrame:
    df = pd.DataFrame(data_as_list)

    # Türkçe sütunların ismini İngilizce yapıyorum
    df = df.rename(columns={ "Product Name" : "Name", "Menşei" : "Origin",
        "Garanti" : "Guarantee", "Genişlik" : "Width",
        "Yükseklik" : "Height", "Ağırlık" : "Weight" })
```

Daha sonra, önceden belirlediğim sütunlar dışındaki sütunları siliyorum ve en az bir null değer içeren her satırı siliyorum:

```
# Önceden yapılan (manüel) null değer testlerine göre belirlediğim az sayıda null içeren
# sütunları ayırıyorum ve daha sonrasında null satırları minimal telefon kaybıyla atıyorum
df = df[ ["Name", "Price", "Origin", "Guarantee", "Width", "Height", "Weight"] ]
df = df.dropna()
```

Bunları yaptıktan sonra fark ettim ki width, height ve weight girişi yapılırken de bir sürü farklı değerler girilmiş. Örneğin width için bazıları mm cinsinden bazıları cm cinsinden değer girmiş. Kimisi “gr” kimisi “Gr” kimisi de “gram” filan yazmış. Aynı durum height ve weight için de geçerliydi. Burada bu değerleri düzelterip, ortak bir ölçü birimine (mm ve gram seçtim) dönüştürüp sütunların veri türünü düzeltmem gerekiyordu:

```
# gr, gram, mm gibi string kısımlarını kaldırma
df['Width'] = df['Width'].map(__wh_mapper_func)
df['Height'] = df['Height'].map(__wh_mapper_func)
df['Weight'] = df['Weight'].map(__weight_mapper_func)
df['Origin'] = df['Origin'].map(lambda x: x.lower())

# veri türlerini düzeltme
df["Guarantee"] = pd.to_numeric(df["Guarantee"])
df["Price"] = pd.to_numeric(df["Price"])
df["Width"] = pd.to_numeric(df["Width"])
df["Height"] = pd.to_numeric(df["Height"])
df["Weight"] = pd.to_numeric(df["Weight"])

# marka bilgisinin oluşturulması
df["Brand"] = df["Name"].copy().apply(brand_mapper_func)
```

Bunlar için fonksiyonlar oluşturdum, ardından veri türünü nümerik yaptım ve en son olarak da kendim bir tane brand feature'ı ekledim.

```
def brand_mapper_func(s: str) -> int:
    s = s.lower()
    # Vatan bilgisayarda bulunan her marka bunlardır
    if s.count('apple') > 0 or s.count('iphone'):
        return 100
    elif s.count('general') > 0:
        return 200
    elif s.count('honor') > 0:
        return 300
    elif s.count('huawei') > 0:
        return 400
    elif s.count('realme') > 0:
        return 500
    elif s.count('samsung') > 0:
        return 600
    elif s.count('tcl') > 0:
        return 700
    elif s.count('tecno') > 0:
        return 800
    elif s.count('xiaomi') > 0:
        return 900
    elif s.count('vivo') > 0:
        return 1000
    else:
        return 1100
```

```
def __weight_mapper_func(s: str) -> str:
    s = s.lower()
    if s.count('gram') > 0:
        return s.replace('gram', '').replace(' ', '')
    elif s.count('gr') > 0:
        return s.replace('gr', '').replace(' ', '')
    elif s.count('g') > 0:
        return s.replace('g', '').replace(' ', '')
    else:
        return s if s.isdigit() else "-1"
```

```
def __wh_mapper_func(s: str) -> str:
    s = s.lower()
    if s.count('cm') > 0:
        s = s.replace('cm', '').replace(',', '.').replace(' ', '')
        s = str(float(s) * 10) # mm yap
        return s
    elif s.count('mm') > 0:
        return s.replace('mm', '').replace(',', '.').replace(' ', '')
```

Ve ardından da dediğim gibi veritabanını ikiye ayırdım:

```
# içine verilen df'yi birinde size satır olacak şekilde ikiye ayırıp döner
def __split_df(df: pd.DataFrame, size: int) -> tuple[pd.DataFrame, pd.DataFrame]:
    df = df.sample(frac=1).reset_index(drop=True) # her satırı shuffle et
    gui_df = df.iloc[:size, :] # ilk size satırı al
    model_df = df.iloc[size:, :] # son n-size satırı al
    return model_df.copy(), gui_df.copy()
```

Ve sonra da model_df kısmını kullanarak modelimi Random Forest ile oluşturdum:

```
def __create_model(df: pd.DataFrame):
    X = df.drop(['Price', 'Name', 'Origin'], axis='columns')
    Y = df['Price']

    X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size = 0.2)
    model = RandomForestClassifier(n_estimators=120)
    model.fit(X_train, y_train)
    score = model.score(X_test, y_test)
    return model, score
```

Burada modelin, test setinde (25 telefon değil, model_df'ten ayrılan test setinde) ne kadar başarılı olduğunu da test ettim ve score olarak döndüm.

Burada kısaca Random Forest tekniğini de açıklamak istiyorum. Ben bu tekniği, bu dönem aldığım Veri Madenciliğine Giriş dersinde gördüm. Kısaca açıklamak gerekirse, Random Forest, bir karar ağacı yöntemine benzer bir model oluşturma yöntemidir. Bu yöntem birden fazla karar ağacını bir araya getirerek, ağaçlar arasındaki ortak karar verme süreciyle model oluşturmaya amaçlar. Her bir karar ağacı, veri setinden rastgele seçilen örnekler üzerinden eğitilir ve her bir ağaç, veri setindeki tüm örnekler için tahminler üretir. Tahminler, birleştirilerek, ortak karar verme süreciyle bir sonuç üretilir. Bu yöntem, özellikle büyük ve karmaşık veri setlerinde oldukça etkilidir ve yüksek doğruluk oranları elde edilebilir.

Bütün bunları yaptıktan sonra da main.py menüsünde çağrılacak fiyat tahmin eden fonksiyonumu tanımladım:

```
def predict_price(model: RandomForestClassifier, values: list) -> int:
    df_val = pd.DataFrame(columns=["Guarantee", "Width", "Height", "Weight", "Brand"])
    df_val.loc[len(df_val)] = values
    y_predicted = model.predict(df_val)
    y_predicted = pd.Series(y_predicted)
    return y_predicted[0]
```


3. Menü Fonksiyonlarının Çalıştırılması

Menü aşağıdaki görseldeki gibi görünmektedir:

```
Uzman Sistemlere Giriş Projesi - Oğuzhan Topaloğlu, Ç19052025
The model is currently being trained, please wait...
Model is done training!
The model's score was 0.48484848484848486

----- MENU -----
1. Predict price by using custom information
2. Predict price for random 25 phones
Your selection: 1
Please enter some information about the phone:
Guarantee (int: month): 24
Width (float: mm): 200.4
Height (float: mm): 350.6
Weight (float: gr): 231
Brand (str): Samsung
The predicted price is 33999 TL
```

Bu görselde 1 seçeneğini seçtim ve bir tahmin yaptırdım. Bu seçenek bizden garanti, genişlik, yükseklik, ağırlık ve marka bilgilerini girmemizi istiyor ve bir fiyat tahmini yapıyor. Burada arkaplanda bir sürü veri tutarlılığı kontrolü de yapıyorum. Girilen değerler gerçekten de int, float, str şeklinde olmalı ve negatif gibi değerler bulunmamalıdır. Brand kısmında olmayan bir marka girilirse de girilebilecek marka türleri yazdırılıyor ve kullanıcıya bilgi veriliyor. (NOT: bu marka listesini doğrudan Vatan Bilgisayar'dan aldım ve bu liste asla değişmiyor, gelecekte değişirse diye n olur n olmaz diye "other" için bir sayı da tanımladım, lütfen bu ayrıntılar için koda bakınız)

Yanlış veri girişi örnekleri:

```
----- MENU -----
1. Predict price by using custom information
2. Predict price for random 25 phones
Your selection: 1
Please enter some information about the phone:
Guarantee (int: month): 1
Width (float: mm): 1
Height (float: mm): 1
Weight (float: gr): 1
Brand (str): havalı oğuzhan markası
The brand must be one of these values:
apple, general, honor, huawei, realme, samsung, tcl, tecno, xiaomi, vivo.
```

```
----- MENU -----
1. Predict price by using custom information
2. Predict price for random 25 phones
Your selection: 1
Please enter some information about the phone:
Guarantee (int: month): -1
Guarantee can't be a negative number!
```

```
----- MENU -----
1. Predict price by using custom information
2. Predict price for random 25 phones
Your selection: 1
Please enter some information about the phone:
Guarantee (int: month): 1
Width (float: mm): 1
Height (float: mm): elma
Height must be a float!
```

Burada ikinci seçenek seçildiğinde de otomatik olarak **rastgele** ayrılan 25 tane telefon için fiyat kontrolü yapılır:

```
----- MENU -----
1. Predict price by using custom information
2. Predict price for random 25 phones
Your selection: 2
Phone is: TCL 30 SE 4/64 GB AKILLI TELEFON MAVİ
Actual price was 5699
Predicted price is 5699

Phone is: SAMSUNG GALAXY S22 ULTRA 256 GB AKILLI TELEFON FANTOM BEYAZ
Actual price was 33999
Predicted price is 33999

Phone is: Xiaomi Redmi Note 11 6GB/128GB Akıllı Telefon Gece Mavisi
Actual price was 7399
Predicted price is 6499

Phone is: Xiaomi Redmi 10 64 Gb Akıllı Telefon Mavi
Actual price was 4999
Predicted price is 5999

Phone is: Xiaomi Redmi Note 11 6GB/128GB Akıllı Telefon Yıldız Mavisi
Actual price was 7399
Predicted price is 6499

Phone is: Samsung Galaxy Z Fold3 5g 256 Gb Akıllı Telefon Gümüş
Actual price was 28999
Predicted price is 28999
```

4. Sistemin Sayısal Başarısı

Oluşturulan modelin skoru Vatan Bilgisayar'da az telefon (yaklaşık 150 adet) bulunmasından dolayı çok yüksek olmamaktadır. Genelde %45-%70 arasında bir skora sahip olduğunu kodu çalıştırarak gözlemledim:

```
Uzman Sistemlere Giriş Projesi - Oğuzhan Topaloğlu, Ç19052025  
The model is currently being trained, please wait...  
Model is done training!  
The model's score was 0.5454545454545454
```

```
Uzman Sistemlere Giriş Projesi - Oğuzhan Topaloğlu, Ç19052025  
The model is currently being trained, please wait...  
Model is done training!  
The model's score was 0.48484848484848486
```

Burada ayrıca ayrılan 25 telefonla ilgili başarı ölçümü yaparken de confusion matrix ve classification report (precision, recall, accuracy ve f1 score yazdırımı yapıyor) kullandım.

Confusion Matrix, bir sınıflandırma modelinin performansını değerlendirmede kullanılan bir ölçümdür. Bu matris, bir sınıflandırma modelinin tahminlerini gerçek sınıflarla karşılaştırarak, modelin ne kadar doğru tahmin ettiğini gösterir. Confusion Matrix, 2 boyutlu bir tablo şeklinde gösterilir ve sıklıkla binary (ikili) sınıflandırma problemlerinde kullanılır. Ancak, birden fazla sınıf için de kullanılabilir. Bu projede, birden çok binary olmayan sınıf için kullanılmıştır.

Confusion Matrix'te, satırlar gerçek sınıfları temsil ederken, sütunlar ise tahmin edilen sınıfları temsil eder. Matrisin sol üst köşesi, "True Positive" (TP) olarak adlandırılan doğru pozitif tahminleri, sağ üst köşesi ise "False Positive" (FP) olarak adlandırılan yanlış pozitif tahminleri gösterir. Sol alt köşesi ise "True Negative" (TN) olarak adlandırılan doğru negatif tahminleri, sağ alt köşesi ise "False Negative" (FN) olarak adlandırılan yanlış negatif tahminleri gösterir.

Precision, bir sınıflandırma modelinin tahminlerinin doğruluk oranını ölçen bir metriktir. Precision, True Positive sayısını, True Positive ve False Positive sayılarının toplamına bölerek hesaplanır.

Matematiksel olarak, precision şu şekilde ifade edilebilir: $Precision = TP / (TP + FP)$

Recall, bir sınıflandırma modelinin gerçek pozitifleri ne kadar iyi tahmin ettiğini ölçen bir metriktir. Recall, True Positive sayısını, True Positive ve False Negative sayılarının toplamına bölerek hesaplanır.

Matematiksel olarak, recall şu şekilde ifade edilebilir: $Recall = TP / (TP + FN)$

Accuracy, bir sınıflandırma modelinin tüm tahminlerinin doğruluk oranını ölçen bir metriktir. Accuracy, True Positive ve True Negative sayılarını, tüm tahminlerin (True Positive, True Negative, False Positive, False Negative) toplamına bölerek hesaplanır.

Matematiksel olarak, accuracy şu şekilde ifade edilebilir: $\text{Accuracy} = (\text{TP} + \text{TN}) / (\text{TP} + \text{TN} + \text{FP} + \text{FN})$

F1 Score, precision ve recall değerlerinin harmonik ortalamasıdır ve sınıflandırma modelinin tahmin performansını ölçmek için kullanılır. F1 Score, precision ve recall değerlerine aynı önem verdiğinden, her iki değerin de yüksek olmasını sağlamaya çalışır.

Matematiksel olarak, F1 Score şu şekilde ifade edilebilir:
$$F1\ Score = \frac{2 * (Precision * Recall)}{(Precision + Recall)}$$

Sistemde ikinci seçenek seçildiğinde bu değerler yazdırılmaktadır ancak Vatan Bilgisayar’da bulunan telefonların sayısının az olması nedeniyle bir sürü değer 0 ve 1’e otomatik olarak atanmakta ve hesaplanamamaktadır. Bunun nedeni yukarıdaki matematiksel formüllerde 0’a bölünmeye çalışılmasıdır. Ayşe hocam Veri Madenciliği dersinde gerçek hayatta bazen bu 0’a bölmeden kaçınılması için 1 ekleniyor gibi bir şey demişti ancak ne kadar doğru bilmiyorum. Sklearn kütüphanesi böyle bir şey yapmıyor ve 0/1 değerlerini yapııştırıp bir uyarı mesajı print ediyor:

```
Confusion matrix:
[[0 0 0 1 0 0 0 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 3 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0]]
```

Dediğim gibi böyle sonuçların elde edilmesinin nedeni belirli markalardan çok telefon olması ve öbür markalardan çok telefon olmaması.

Vatan bilgisayarda Apple, Samsung, Xiaomi gibi markalardan çok telefon varken OPPO, General Mobile gibi markalardan telefon çok bulunmuyor:

- ☐ APPLE (45)
- ☐ GENERAL MOBILE (10)
- ☐ HONOR (2)
- ☐ HUAWEI (14)
- ☐ OPPO (6)
- ☐ REALME (2)
- ☐ SAMSUNG (61)
- ☐ TCL (8)
- ☐ TECNO (22)

Bu da metriklerin yazdırılırken aşağıdaki uyarıları basmasına neden oluyor:

```
Accuracy is 0.52
Classification report is:
C:\Users\oguzh\myVenv\lib\site-packages\sklearn\metrics\_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no
predicted samples. Use 'zero_division' parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\oguzh\myVenv\lib\site-packages\sklearn\metrics\_classification.py:1344: UndefinedMetricWarning: Recall and F-score are ill-defined and being set to 0.0 in labels with no tr
ue samples. Use 'zero_division' parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\oguzh\myVenv\lib\site-packages\sklearn\metrics\_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no
predicted samples. Use 'zero_division' parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\oguzh\myVenv\lib\site-packages\sklearn\metrics\_classification.py:1344: UndefinedMetricWarning: Recall and F-score are ill-defined and being set to 0.0 in labels with no tr
ue samples. Use 'zero_division' parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\oguzh\myVenv\lib\site-packages\sklearn\metrics\_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no
predicted samples. Use 'zero_division' parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\oguzh\myVenv\lib\site-packages\sklearn\metrics\_classification.py:1344: UndefinedMetricWarning: Recall and F-score are ill-defined and being set to 0.0 in labels with no tr
ue samples. Use 'zero_division' parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

Ancak bu uyarılara rağmen sklearn yine de metrikleri bize bir tablo ile veriyor:

	precision	recall	f1-score	support
4999	0.00	0.00	0.00	4
5299	1.00	1.00	1.00	1
5699	1.00	1.00	1.00	1
5999	0.00	0.00	0.00	0
6299	1.00	1.00	1.00	1
6499	0.33	1.00	0.50	1
6799	0.00	0.00	0.00	1
6999	0.00	0.00	0.00	0
7399	0.00	0.00	0.00	2
8299	0.00	0.00	0.00	1
8999	0.75	1.00	0.86	3
11999	0.00	0.00	0.00	1
15499	0.50	1.00	0.67	1
17599	0.00	0.00	0.00	1
18999	1.00	1.00	1.00	1
19999	0.00	0.00	0.00	0
26499	1.00	1.00	1.00	1
28999	0.50	1.00	0.67	1
29999	1.00	1.00	1.00	1
30999	0.00	0.00	0.00	0
33999	1.00	1.00	1.00	1
38899	0.00	0.00	0.00	1
38999	0.00	0.00	0.00	1
accuracy			0.52	25
macro avg	0.39	0.48	0.42	25
weighted avg	0.42	0.52	0.46	25

5. Kaynaklar

Projede kaynak olarak sklearn dokümantasyonunu ve stackoverflow'u kullandım. Özellikle preprocessing sırasında kullandığım fonksiyonları stackoverflow'da öğrendim.

6. Youtube Videosu

Link: <https://www.youtube.com/watch?v=BTy5sTAC0vg> (5:31 süreli)

7. Program Dosyalarının Listesi

Projede şu dosyalar vardır: src klasörü (main.py, expert.py, scrape.py, menu.py) ve bu pdf.

NOT: projeyi çalıştırırken “python main.py” komutunu terminalde kullanabilirsiniz. Projede Python 3.9.0 kullandım ancak 3 sonrası her sürümün çalışacağını düşünüyorum. :)