İ.B.B. Trafik Verileri Kullanılarak Makine Öğrenmesi ile Trafik Yoğunluğunun Tahmin Edilmesi

Yazılım Mühendisliği Ana Bilim Dalı

Dönem Projesi

Songül ATAÇ

Proje Danışmanı: Prof. Dr. Femin Yalçın Küçükbayrak

Ocak 2024

İBB Trafik Verileri Kullanılarak Makine Öğrenmesi ile Trafik Yoğunluğunun Tahmin Edilmesi

Öz

Bu projede, İstanbul Büyükşehir Belediyesi'nin (İBB) sağladığı trafik verileri üzerinde makine öğrenimi teknikleri kullanılarak trafik yoğunluğunun tahmini amaçlanmıştır. Projenin odak noktası, İstanbul'un farklı bölgelerindeki trafik yoğunluğunu etkileyen faktörleri belirleyerek gelecekteki trafik durumunu tahmin etmektir.

Veri seti, İBB'nin trafik sensörleri ve kayıt sistemleri tarafından sağlanan geniş kapsamlı bir veri kümesini içermektedir. Bu veri kümesi, trafik yoğunluğunu etkileyebilecek değişkenlerin yanı sıra tarih, saat, bölge koordinatları gibi çeşitli özellikleri içermektedir.

Proje kapsamında, veri setinin analizi ve ön işleme adımları gerçekleştirilerek veri keşfi yapılmıştır. Model performansını arttırabilecek özellikler de eklenmiş ve makine öğrenimi algoritmaları kullanılarak trafik yoğunluğunu tahmin etmek için model oluşturma ve eğitme aşamaları gerçekleştirilmiştir. Bu aşamada, lineer regresyon, karar ağaçları, gradient boosting gibi çeşitli modelleme yöntemleri değerlendirilmiş ve en uygun model seçilmiştir.

Proje, İstanbul'da trafik yoğunluğunun tahmini için kullanılabilir veri odaklı bir yaklaşım sunarak, trafik yönetimi ve planlamasında değerli bilgiler sunmayı amaçlamaktadır.

Anahtar Sözcükler: Makine öğrenimi, modelleme, performans metrikleri, trafik verileri, veri analizi

Prediction of Traffic Congestion using Machine Learning on Istanbul Metropolitan Municipality Traffic Data

Abstract

This project aims to predict traffic congestion by utilizing machine learning techniques on traffic data provided by the Istanbul Metropolitan Municipality (IBB). The primary focus of the project is to forecast future traffic conditions by identifying the factors influencing traffic congestion in different regions of Istanbul.

The dataset comprises a comprehensive collection of data from traffic sensors and recording systems maintained by IBB. Alongside variables affecting traffic congestion, the dataset encompasses various features like date, time, and geographical coordinates.

Within the project scope, data analysis and preprocessing steps were conducted to explore the dataset. Features that could enhance model performance were incorporated. Subsequently, model building and training for predicting traffic congestion using machine learning algorithms were executed. Various modeling techniques such as linear regression, decision trees, and gradient boosting were evaluated to select the most suitable model.

The project aims to provide valuable insights into traffic management and planning in Istanbul by presenting a data-centric approach to predict traffic congestion.

Keywords: Machine learning, modeling, performance metrics, traffic data, data analysis

İçindekiler

Şekiller Listesi

Kısaltmalar Listesi

KNN	K-Nearest Neighbour
MAE	Mean Absolute Error
MSE	Mean Squared Error
Colab	Colaboratory

Bölüm 1

Giriş

İstanbul, günümüzde karmaşık trafik yapılarıyla bilinen bir metropol haline gelmiştir. Bu durum, şehirdeki yoğun trafikten kaynaklanan bir dizi sorunu da beraberinde getirmektedir. Bu proje, İstanbul Büyükşehir Belediyesi'nin (İBB) sağladığı saatlik lokasyon bazlı verileri kullanarak, şehirdeki trafik yoğunluğunu tahmin etmeyi amaçlamaktadır.

Veri seti, İstanbul'un farklı bölgelerindeki trafik akışını, yoğunluğunu ve diğer ilgili faktörleri saatlik olarak içermektedir. Makine öğrenimi tekniklerini kullanmak suretiyle, bu veriler analiz edilerek gelecekteki trafik yoğunluğunun tahmini hedeflenmektedir.

Proje, trafik tahminleme modelleri üzerine odaklanarak, farklı algoritmaların ve tekniklerin trafik yoğunluğunu tahmin etme yeteneklerini araştıracaktır. Bu çalışma, İstanbul'daki trafik yönetimine katkıda bulunma potansiyeline sahip olup, gerçek dünya senaryolarıyla uyumlu ve kullanılabilir sonuçlar üretmeyi hedefler.

Bölüm 2

Veri seti ve Araçların Tanıtımı

2.1 Kullanılan Geliştirme Ortamı

Projede geliştirme aracı olarak kullanılan Google Colab, Python programlama dilini kullanarak bulut tabanlı bir geliştirme ortamı sunar. Python kodunun tarayıcı üzerinde çalıştırılmasına olanak tanır ve ücretsiz olarak CPU, GPU veya TPU gibi hesaplama kaynaklarına erişim sağlar. Bu sayede Python tabanlı veri analizi, yapay zeka ve makine öğrenme projeleri kolaylıkla yürütülebilir. Colab, Python topluluğunun gelişimini ve işbirliğini teşvik eden açık kaynaklı bir platformdur.

Veri setinin makine öğrenmesine tabi tutulması süreci Google'a ait Colaboratory (kısaca Colab) ürünü aracılığı ile gerçekleştirildi. Colab'ın tercih edilmesinin sebebi kurulum gerektirmeyen, bulut tabanlı bir uygulama olmasının yanısıra, bilgi işlem kaynaklarına ücretsiz erişim sağlıyor oluşudur.

2.2 Veri setinin İçeriği

Veri seti İstanbul Büyükşehir Belediyesinin resmi web sitesinde yer alan İstanbul'a ait farklı lokasyonların saatlik yoğunluk ve trafik bilgisinden oluşmaktadır. Aylık olarak erişilebilir durumda olan veri setlerinden 2023 yılına ait olanların tümü, Eylül ayı da dahil olmak üzere kullanıldı. Her biri tek tek okunup dataframede birleştirildi. Drive'a yüklenmiş olan 9 adet veri seti Google Colab aracılığı ile Şekil 2.1'deki gibi içeri alındı.

```
from google.colab import drive
import pandas as pd

drive.mount('/content/drive')

dizin = '/content/drive/My Drive/Colab Notebooks2/'

dataset = glob.glob(dizin + 'traffic_density_*.csv')

dataset = pd.concat([pd.read_csv(dosya) for dosya in dataset], ignore_index=True)
dataset = dataset.sort_values(by='DATE_TIME')
```

Şekil 2.1: Veri setinin pandas kütüphanesi aracılığı ile içeri alınması

Önizleme görebilmek adına dataset.head() komutu ile veri setine ait ilk 5 satır görünümü Şekil 2.2'deki gibi elde edildi.

```
dataset.head()
```

	DATE_TIME	LONGITUDE	LATITUDE	GEOHASH	MINIMUM_SPEED	MAXIMUM_SPEED	AVERAGE_SPEED	NUMBER_OF_VEHICLES
7711471	2023-01-01 00:00:00	41.168518	28.756714	sxk6nq	12	116	79	27
7713019	2023-01-01 00:00:00	41.135559	29.064331	sxkdj2	15	47	37	9
7713020	2023-01-01 00:00:00	41.064148	28.987427	sxk9ee	3	105	33	126
7713021	2023-01-01 00:00:00	40.965271	28.833618	sxk901	9	87	51	55
7713022	2023-01-01 00:00:00	41.108093	28.558960	sxk3cg	64	140	92	15

Şekil 2.2: Veri setinin ilk beş satır görünümü

Tüm veri setleri farklı boyutta data içermektedir. Birleştirilen verilerin dataframe içerisinde ne kadar veriye sahip olduğu Şekil 2.3'deki gibi sorgulandı. 14.317.147 satır veri ve 8 adet parametreden oluştuğu gözlemlendi.

```
dataset.shape
```

```
(14317147, 8)
```

Şekil 2.3: Veri setinin boyut bilgisi

Parametre bilgileri Şekil 2.4'deki gibidir. Orijinal veri setlerinde 2 adet object tipinde, 2 adet float tipinde, 4 adet int tipinde parametre mevcuttur.

```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 14317147 entries, 7711471 to 1639022
Data columns (total 8 columns):
 #   Column              Dtype
---  ------              -----
 0   DATE_TIME           object
 1   LONGITUDE           float64
 2   LATITUDE            float64
 3   GEOHASH             object
 4   MINIMUM_SPEED       int64
 5   MAXIMUM_SPEED       int64
 6   AVERAGE_SPEED       int64
 7   NUMBER_OF_VEHICLES  int64
dtypes: float64(2), int64(4), object(2)
memory usage: 983.1+ MB
```

Şekil 2.4: Veri setinin içerdiği parametreler

2.3 Veri Temizleme ve Veri Keşfi

Veri setinde tarih alanı datetime değil, date tipine dönüştürülerek ve bu alan kullanılarak saat, ay ve gün bilgileri ayrı parametreler olarak veri setine eklenecek ve en son bu parametre veri setinden çıkartılacaktır.

Geohash parametresi ilgili konumun koordinatlarını barındıran bir kodlama sistemini temsil etmektedir. Veri setinde enlem ve boylam bilgileri bulunduğundan bu alan drop edilerek veri setinden çıkartılacaktır. Çünkü çıkartılmaması halinde string bir değer olduğundan modeli olumsuz etkileyecektir.

Veri setinde modeli olumsuz etkileyecek missing value içeren alanların olup olmadığına Şekil 2.5'deki gibi bakıldı ve isnull fonksiyonundan faydalanıldı. Herhangi bir missing value olmadığı görüldü.

```
dataset.isnull().sum()
```

```
DATE_TIME           0
LONGITUDE           0
LATITUDE            0
GEOHASH             0
MINIMUM_SPEED       0
MAXIMUM_SPEED       0
AVERAGE_SPEED       0
NUMBER_OF_VEHICLES  0
dtype: int64
```

Şekil 2.5: Missing Value kontrolü

Datetime parametresinin içeriğinden faydalanılarak veri setine ay, gün, saat bilgisi eklenilerek veri setinin makine öğrenmesi için daha stabil hale gelmesi amaçlandı. Şekil 2.6'daki gibi bu alanlar tek tek eklendi. Bununla beraber ilgili tarihin haftanın kaçıncı günü olduğu bilgisini içeren day of week parametresi de eklendi ve bir ön izlemesi alındı.

```
dataset['DATE_TIME'] = pd.to_datetime(dataset['DATE_TIME'])
dataset['HOUR'] = dataset['DATE_TIME'].dt.hour
dataset['MONTH'] = dataset['DATE_TIME'].dt.month
dataset['DAY'] = dataset['DATE_TIME'].dt.day
dataset['DAY_OF_WEEK'] = dataset['DATE_TIME'].dt.dayofweek + 1
dataset.tail()
```

	DATE_TIME	LONGITUDE	LATITUDE	GEOHASH	MINIMUM_SPEED	MAXIMUM_SPEED	AVERAGE_SPEED	NUMBER_OF_VEHICLES	HOUR	MONTH	DAY	DAY_OF_WEEK
1639229	2023-09-30 23:00:00	28.646851	41.080627	sxk3ey	68	113	91	7	23	9	30	6
1639230	2023-09-30 23:00:00	29.284058	41.080627	sxkcdq	32	131	91	53	23	9	30	6
1639231	2023-09-30 23:00:00	28.317261	41.102600	sxk1u6	69	126	97	12	23	9	30	6
1639179	2023-09-30 23:00:00	28.800659	40.970764	sxk3p6	7	75	45	30	23	9	30	6
1639022	2023-09-30 23:00:00	28.701782	41.184998	sxk6m1	2	85	54	20	23	9	30	6

Şekil 2.6: Veri setine ay, gün, saat ve haftanın günü bilgisinin eklenmesi

Model performansını arttırmaya yönelik yine veri setine ilgili tarihin hafta sonu olup olmadığı bilgisi de Şekil 2.7'deki gibi eklendi. Hafta sonu ise 1, değil ise 0 bilgisi eklenmiş oldu. Hafta sonu olup olmadığı bilgisi için veri setine eklenmiş olan day of week parametresinden faydalanıldı.

```
dataset['WEEKEND'] = np.where((dataset['DAY_OF_WEEK'] == 6) | (dataset['DAY_OF_WEEK'] == 7), 1, 0)

dataset.head()
```

	DATE_TIME	LONGITUDE	LATITUDE	GEOHASH	MINIMUM_SPEED	MAXIMUM_SPEED	AVERAGE_SPEED	NUMBER_OF_VEHICLES	HOUR	DAY	DAY_OF_WEEK	WEEKEND
7711471	2023-01-01	41.168518	28.756714	sxk6nq	12	116	79	27	0	1	7	1
7713019	2023-01-01	41.135559	29.064331	sxkdj2	15	47	37	9	0	1	7	1
7713020	2023-01-01	41.064148	28.987427	sxk9ee	3	105	33	126	0	1	7	1
7713021	2023-01-01	40.965271	28.833618	sxk901	9	87	51	55	0	1	7	1
7713022	2023-01-01	41.108093	28.558960	sxk3cg	64	140	92	15	0	1	7	1

Şekil 2.7: Veri setine hafta sonu olup olmadığı bilgisinin eklenmesi

Veri setine ilgili tarihin hangi mevsime ait olduğu bilgisini içeren bir parametre olarak season parametresi eklendi. Buna göre eklenmiş olan month alanı 12-1-2 ise kış bilgisine karşılık gelmek üzere 1 bilgisini, 3-4-5 ise ilkbahar bilgisine karşılık gelmek üzere 2 bilgisini, 6-7-8 ise yaz bilgisine karşılık gelmek üzere 3 bilgisini ve son olarak 9-10-11 ise sonbahar bilgisine karşılık gelmek üzere 4 bilgisini döndüren bir fonksiyon aracılığı ile bu bilgi de Şekil 2.8'deki gibi season parametresine eklendi.

```
def get_season(month):
    if month in [12, 1, 2]:
        return 1
    elif month in [3, 4, 5]:
        return 2
    elif month in [6, 7, 8]:
        return 3
    elif month in [9, 10, 11]:
        return 4
    else:
        return None

dataset['season'] = dataset['DATE_TIME'].dt.month.apply(get_season)
```

Şekil 2.8: Veri setine mevsim bilgisinin eklenmesi

Hangi tarihlerin resmi tatillere denk geldiği bilgisini veri setine eklemek için Şekil 2.9'daki gibi düzenleme yapıldı. Resmi tatil tarihlerini içeren bir liste tanımlandı. Resmi tatil isminde bir parametre belirlenerek önce tümü için 0 değeri atandı. Daha sonra datetime alanında bu tarihleri barındıran satırlar için 1 değeri ile değiştirildi.

```
resmi_tatil_tarihleri = ['2023-01-01', '2023-04-20', '2023-04-21','2023-04-22','2023-04-23',
                         '2023-05-01','2023-05-19','2023-06-27','2023-06-28','2023-06-29',
                         '2023-06-30','2023-07-01','2023-07-15','2023-08-30']

dataset['PUBLIC_HOLIDAY'] = 0

dataset.loc[dataset['DATE_TIME'].dt.date.astype(str).isin(resmi_tatil_tarihleri), 'PUBLIC_HOLIDAY'] = 1
```

Şekil 2.9: Veri setine resmi tatil bilgisinin eklenmesi

Okul tatil tarihlerinin de trafik durumunu son derece etkilediği bilindiğinden, öncelikle ara tatil tarihleri de aynı şekilde listeye eklendi. School holiday isminde bir parametre eklendi ve başlangıç değerleri 0 olarak atandı. Listedeki tarihleri içeren satırlar 1 değeri ile Şekil 2.10'daki gibi revize edildi.

```
dataset['SCHOOL_HOLIDAY'] = 0

okul_aratatil_tarihleri = ['2023-01-21','2023-01-22','2023-01-23', '2023-01-24', '2023-01-25','2023-01-26',
                           '2023-01-27', '2023-01-28','2023-01-29','2023-01-30','2023-01-31',
                         '2023-02-01','2023-02-02','2023-02-03','2023-02-04','2023-02-05','2023-04-15','2023-04-16',
                           '2023-04-17','2023-04-18','2023-04-19','2023-04-20','2023-04-21','2023-04-22','2023-04-23']

dataset.loc[dataset['DATE_TIME'].dt.date.astype(str).isin(okul_aratatil_tarihleri), 'SCHOOL_HOLIDAY'] = 1
```

Şekil 2.10: Veri setine okul ara tatil bilgisinin eklenmesi

Okul yaz tatili bilgisini de yine aynı school holiday parametresine entegre etmek için Şekil 2.11'deki gibi başlangıç ve bitiş tarihleri arasındaki tarihleri içeren satırlar 1 değeri ile güncellendi.

```
okul_yaztatili_baslangic = pd.to_datetime('2023-06-17')
okul_yaztatili_bitis = pd.to_datetime('2023-09-10')


dataset.loc[(dataset['DATE_TIME'] >= okul_yaztatili_baslangic) & (dataset['DATE_TIME'] <= okul_yaztatili_bitis), 'SCHOOL_HOLIDAY'] = 1
```

Şekil 2.11: Veri setine okul yaz tatili bilgisinin eklenmesi

Son olarak string tipinde veri içerdiği için modeli olumsuz etkileyebilecek ve de enlem ve boylam bilgileri zaten mevcut olduğundan Geohash parametresi ve içerisindeki ay,

gün, saat bilgileri ayrı parametreler olarak veri setine eklenmiş olan Datetime parametresi veri setinden çıkartıldı (bakınız Şekil 2.12).

```
dataset = dataset.drop(["GEOHASH", "DATE_TIME"], axis=1)
```

Şekil 2.12: Veri setinden Geohash ve Datetime parametrelerinin çıkartılması

Veri setinden gerekli temizleme ve özellik mühendisleri sonrası genel görünümü Şekil 2.13'deki halini almıştır.

```
dataset.head()
```

	LONGITUDE	LATITUDE	MINIMUM_SPEED	MAXIMUM_SPEED	AVERAGE_SPEED	NUMBER_OF_VEHICLES	HOUR	MONTH	DAY	DAY_OF_WEEK	WEEKEND	SEASON	PUBLIC_HOLIDAY	SCHOOL_HOLIDAY
7711471	41.168518	28.756714	12	116	79	27	0	1	1	7	1	1	1	0
7713019	41.135559	29.064331	15	47	37	9	0	1	1	7	1	1	1	0
7713020	41.064148	28.987427	3	105	33	126	0	1	1	7	1	1	1	0
7713021	40.965271	28.833618	9	87	51	55	0	1	1	7	1	1	1	0
7713022	41.108093	28.558960	64	140	92	15	0	1	1	7	1	1	1	0

Şekil 2.13: Veri setinin temizlik ve özellik mühendisliği sonrası görünümü

Bölüm 3

Makine Öğrenmesi

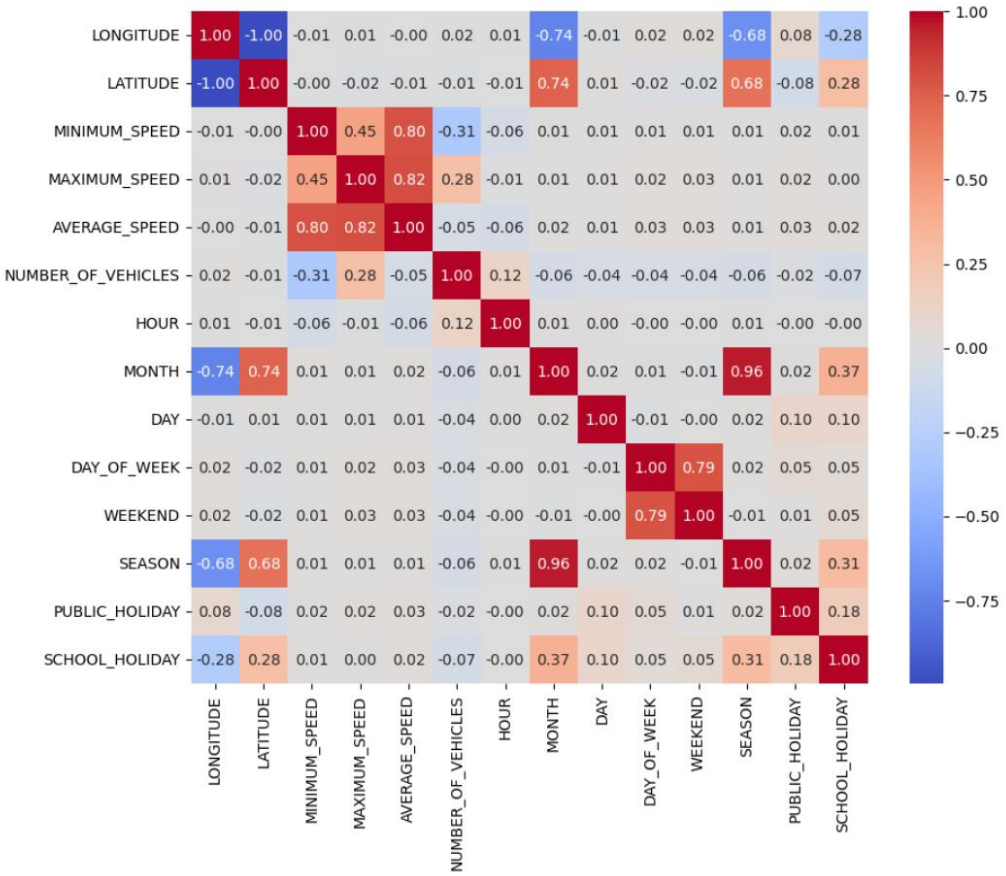
3.1 Model Seçimi

Projenin amacı "NUMBER_OF_VEHICLES" parametresi ile temsil edilen toplam araç sayısını tahmin etmektir. Bu değer sayısal bir değer olduğundan bir regresyon problemi söz konusudur. Regresyon, sayısal veya sürekli bir çıktı tahmin etmeyi hedefler.

Bu durumda, regresyon modelleri (Random Forest Regressor, Gradient Boosting Regressor gibi) kullanılarak "NUMBER_OF_VEHICLES" değerinin tahmini yapılabilir.

Veri setinin bağımlı değişkeni ve bağımsız değişkenleri arasındaki ilişkiyi gözlemlemek için Şekil 3.1'deki gibi korelasyon analizi yapılmış ve ısı haritası şeklinde gözlemlenmiştir.

Buna göre değişkenler arasında anlamlı, güçlü negatif veya pozitif yönde bir ilişki gözlemlenemediğinden doğrusal regresyon modelleri çalışmaya dahil edilmemiştir.



Şekil 3.1: Veri setinin korelasyon ısı haritası

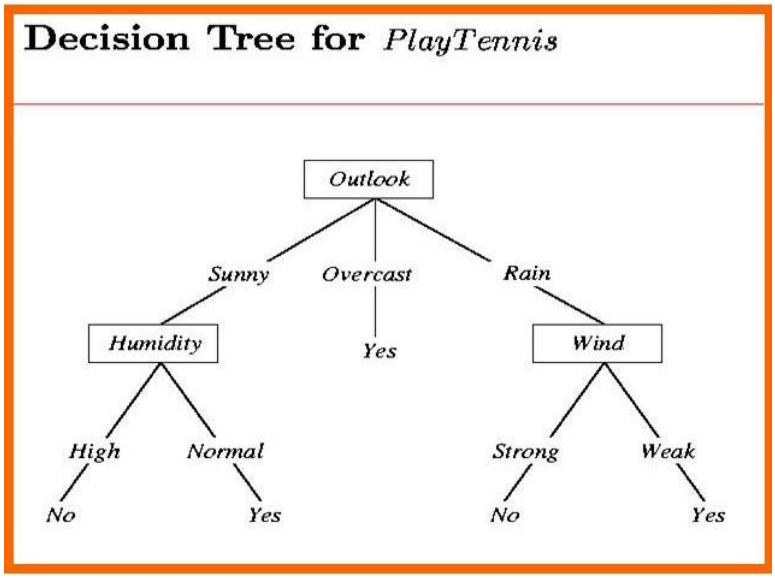
Doğrusal olmayan regresyon modelleri, veri setindeki sayısal çıktıyı tahmin etmek için uygun olacaktır. Bu modeller arasında en iyi performansı gösteren ve veri setinin özelliklerine en uygun olan seçilecektir.

3.1.1 Decision Tree Algoritması

Karar ağacı olarak da adlandırılan bu model hem sınıflandırma hem de regresyon problemlerinde kullanılabilen bir makine öğrenmesi algoritmasıdır. Veri setini bir dizi decision (karar) kuralına veya dal olarak isimlendirilen yapılara ayırır. Bu kurallar, girdilerin belirli değerlerindeki farklı sonuçları tanımlamak için kullanılır.

Karar ağacı, veri setindeki her bir özellikten yararlanarak en iyi bölünmeleri seçerek işleme başlar. Bölünmeler, veri setinin homojen yapısını arttıracak şekilde meydana gelir. Bölünmeler sonucunda Şekil 3.2'deki gibi bir ağaç yapısı oluşturulur. Her bir düğüm, bir sınıf etiketine veya bir regresyon değerine karşılık gelir.

Karar ağaçları hem sınıflandırma hem de regresyon problemlerinde kullanılabilmeleri, kolay bir şekilde görselleştirilebilir ve yorumlanabilir olmaları sebebiyle sıklıkla tercih edilen bir algoritmadır.



Şekil 3.2: Decision Tree algoritması ağaç yapısı örneği

3.1.2 Polynomial Regression Algoritması

Polinom regresyon, bağımlı ve bağımsız değişken arasındaki ilişkiyi ortaya koymak için kullanılan bir regresyon analizi yöntemidir ve bu ilişkiyi ifade etmek için polinom kullanır. Doğrusal regresyon yönteminin genişletilmiş halidir. Bağımsız parametrelerin kuvvetlerini içeren polinomlardan yararlanarak doğrusal olmayan ilişkileri modellemek için kullanılır.

Bir bağımsız değişken olan x ve bir bağımsız değişken olan y arasındaki ilişkiyi ifade etmek için doğrusal regresyon denklemi olan $y = mx + b$ denklemi kullanılır. Polinom regresyonunda ise bu ilişki daha karmaşıktır. Örneğin denklem için $y = a + bx + cx^2 + dx^3$ gibi bir ifade kullanılabilir. Bu denklem, daha yüksek dereceli terimlerle bağımsız değişkenlerin kuvvetlerini içerir.

Doğrusal olmayan veri setlerini modellemek için kullanılırken yalnızce katsayılar ve kuvvetler arasında doğrusal bir ilişki kurar. Polinom derecesinin seçimi önemlidir,

yüksek bir derece seçimi modelin aşırı öğrenme (overfitting) yaşamasına sebep olabilir.

3.1.3 Random Forest Algoritması

Random forest algoritması, birçok karar ağacının bir araya gelerek oluşturduğu bir yapıdır. Her bir ağaç, random olarak seçilen alt özelliklerle eğitilir ve bu yolla örneklem alınır. Son olarak, bütün ağaçların tahmin değerleri bir araya getirilir ve genel bir tahmin yapılır. Hem sınıflandırma hem de regresyon problemleri için kullanılabildiğinden çok yönlü bir algoritmadır.

Tek bir karar ağacı aşırı öğrenmeye sebep olabilecekken, birden fazla ağacın birleşimi olan random forest algoritması ile daha iyi genelleme yapılabilmesi mümkündür.

Algoritma kullanılırken, ormanın içerisindeki ağaçların sayısı (n_estimators), ağaçların derinliği (max_depth) ve ağaçların ne derece rastgele oluşturulduğu (max_features) bilgisi ve diğer birçok parametre ayarlanabilir.

3.1.4 LightGBM Regression Algoritması

LightGBM Regresyon, gradient boosting yöntemini kullanan bir algoritmadır. Zayıf karar ağaçlarının bir araya gelip güçlü bir model oluşturmasına dayalıdır. Önceki ağaçların hatalarını ortadan kaldırmaya çalışarak sonraki ağacı eğitmeye dayalı dengeli bir modelleme yaklaşımı bulunmaktadır. Bu da aşırı öğrenme riskini azaltır.

Büyük veri setleri ile iyi performans göstererek diğer gradient boosting kütüphanelerinden ayrılır.

Diğer algoritmalardan farklı olarak, sayısal değere dönüştürülmesine gerek kalmaksızın kategorik parametrelerle çalışabilir.

Özellikle büyük veri setleri üzerinde hızlı ve etkili bir şekilde çalıştığından sınıflandırma ve regresyon problemlerinde tercih edilen bir algoritmadır.

3.1.5 K-Nearest Neighbour Algoritması

Makine öğrenimi alanında sınıflandırma ve regresyon problemleri için kullanılan bir algoritmadır. Bu algoritma, veri noktalarının etrafındaki komşularını kullanarak tahminler yapar.

KNN'nin çalışma prensibi oldukça basittir: bir veri noktasını tahmin etmek için, ona en yakın komşularının etiketlerini veya değerlerini kullanır. KNN, denetimli bir öğrenme algoritmasıdır, yani eğitim verilerinin her biri bir girdi-çıktı çiftinden oluşur. Sınıflandırma için kullanıldığında, KNN bir veri noktasını çevresindeki komşularının çoğunluğunun sınıf etiketiyle ilişkilendirir. Regresyon için kullanıldığında ise, komşuların çıktı değerlerinin ortalamasını alarak tahmin yapar.

KNN'nin temel parametresi "K" değeridir, bu değer komşu sayısını temsil eder. Bir veri noktasının tahmin edilmesi için kaç komşunun kullanılacağını belirler. K değeri seçilirken, veri setinin boyutu, veri yapısının karmaşıklığı ve modelin genelleme performansı göz önünde bulundurulmalıdır.

KNN'nin avantajları arasında basitliği ve eğitim sürecinin olmaması yer alır. Ancak, büyük veri setlerinde ve yüksek boyutlu veri yapılarında hesaplama maliyeti artabilir. Ayrıca, veri setindeki gürültülü veya gereksiz özellikler KNN performansını olumsuz etkileyebilir.

3.2 Performans Metrikleri

3.2.1 R-Kare (Coefficient of Determination – R-squared)

R-Kare, regresyon modellerinde bağımsız değişkenin bağımlı değişken üzerindeki varyansı açıklama başarısını ölçen bir metriktir.

0 ile 1 arasında bir değer alır ve genellikle yüzde cinsinden ifade edilir ve formülü Şekil 3.3’deki gibidir.

$$R^2 = 1 - \sum_{i}^{n} \frac{(y_{i_{gerçek}} - y_{i_{tahmin}})^2}{(y_{i_{gerçek}} - ortalama(y_{gerçek}))^2}$$

Şekil 3.3: R-Kare formülü

R-Kare değerinin 0 olması, bağımsız değişkenlerin bağımlı değişken üzerindeki hiçbir varyansın model tarafından açıklanamadığı anlamına gelmektedir. En kötü tahmin durumuna işaret eder.

R-Kare değerinin 1 olması ise, bağımsız değişkenlerin bağımlı değişken üzerindeki tüm varyansın model tarafından açıklanabildiği anlamına gelir. Bu durumda tahmin değerlerinin tamamı gerçek değerlere eşit olmuştur. Ancak 1'e eşit olma durumu büyük bir olasılıkla modelin ezberleme yaptığına işarettir, istenen bir durum değildir.

1'e yakın değerlerin elde edilmesi iyi bir doğruluk sonucudur denebilir. Ancak tek başına modelin tüm performansını ifade edemez. Diğer performans metrikleri ile birlikte değerlendirilmesi gereklidir.

3.2.2 Ortalama Mutlak Hata (Mean Absolute Error - MAE)

Ortalama mutlak hata, bir regresyon modeli tahminlerinin gerçek değerlerden ne kadar farklı olduğunu gösteren bir metriktir. Gerçek ve tahmin edilen değerler arasındaki farkın mutlak değerlerinin ortalamasına işaret eder.

Formülü Şekil 3.4'deki gibi olan bu metrikte x_i gerçek değerler, x tahmin edilen değerlerdir. n ise gözlem sayısını ifade etmektedir.

$$\mathbf{MAE} = \frac{1}{n} \sum_{i=1}^{n} |\boldsymbol{x}_i - \boldsymbol{x}|$$

Şekil 3.4: MAE formülü

Hesaplanan bir MAE değeri 5 ise, bu ortalama olarak model tahminlerinin gerçek değerlerden 5 birim uzaklıkla olduğunu gösterir.

3.2.3 Ortalama Kare Hata (Mean Squared Error - MSE)

Ortalama kare hata, bir regresyon modeli için gerçek ve tahmin değerleri arasındaki farkların karesinin ortalamasını gösteren bir metriktir. Tahmin değerlerinin gerçek değerlerden ne kadar uzak olduğunun ölçüsüne işaret eder.

Formülü Şekil 3.5'deki gibi olan bu metrikte y_i gerçek değerler, $\hat{y}i$ tahmin edilen değerlerdir. n ise gözlem sayısını ifade etmektedir.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \tilde{y}_i)^2$$

Şekil 3.5: MSE formülü

MSE'nin karelerin ortalaması olduğu için, büyük hataların diğer hatalardan daha fazla ağırlığa sahip olduğu bir ölçüdür. Bu durum, büyük hataların modelin performansını daha fazla etkileyebileceği anlamına gelir. Aykırı değerlerin varlığı sonucu MSE yüksek çıkabilir.

Bölüm 4

Bulgular

4.1 Decision Tree Algoritması ile Modelin Eğitilmesi

Veri seti Şekil 4.1'deki gibi test ve eğitim setlerine ayrıldıktan sonra, veri setindeki datalar doğrusal olmayan regresyon algoritmalarından olan decision tree algoritması ile Şekil 4.2'deki gibi eğitildiğinde R^2 değeri 0,81, MAE değeri 19,24, MSE değeri 1411 olarak bulundu.

```
X = dataset.drop(["NUMBER_OF_VEHICLES","GEOHASH","DATE_TIME"], axis=1)
y = dataset["NUMBER_OF_VEHICLES"]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

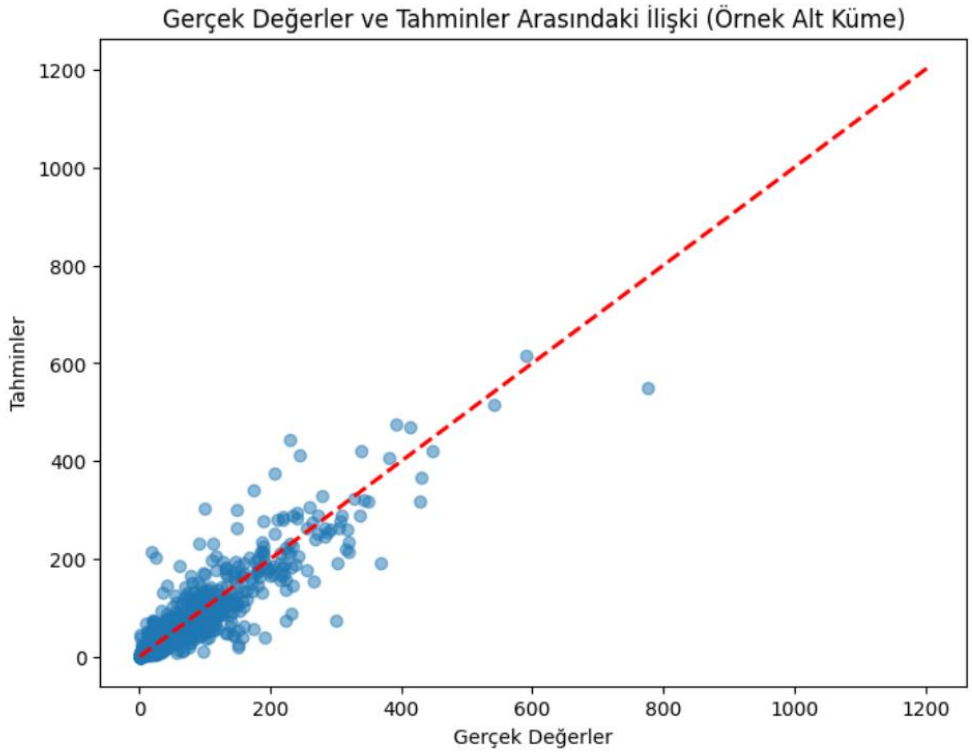
Şekil 4.1: Veri setinin test ve eğitim setlerine ayrılması

```
from sklearn.tree import DecisionTreeRegressor

model = DecisionTreeRegressor()


model.fit(X_train, y_train)


y_pred = model.predict(X_test)


print("R-squared:", r2_score(y_test, y_pred))
print("MAE:", mean_absolute_error(y_test, y_pred))
print("MSE:", mean_squared_error(y_test, y_pred))
```

```
R-squared: 0.8155865253315923
MAE: 19.240147422263973
MSE: 1411.286476940825
```

Şekil 4.2: Decision Tree algoritma metrik sonuçları

Model performansına dair tahmin edilen değerler ile gerçek değerlerin karşılaştırması Şekil 4.3'de 1000 adet örneklem seçilerek dağılım grafiği üzerinde gösterilmiştir.



Şekil 4.3: Decision Tree tahmin değerleri karşılaştırması

R^2 değeri 0,81 olarak hesaplandığından, modelin verilere oldukça iyi uyum sağladığı söylenebilir.

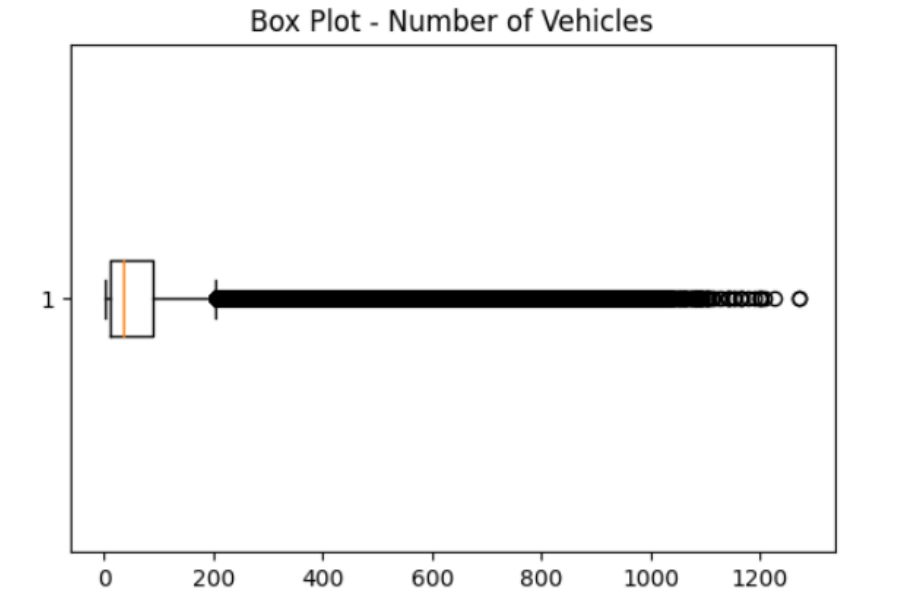
MAE değerinin 19,24 olarak hesaplanması, ortalama bir hata olduğunu gösteriyor. Ortalama olarak modelin hedef değişkeni 19,24 birimlik hata ile tahmin ettiği söylenebilir.

MSE değerinin 1411,29 olarak hesaplanması, modelin çok büyük hatalar yapmadığını, ancak bazı örneklerde daha büyük hatalar yapabildiğini gösteriyor.

Genel olarak, R^2 yüksek olduğundan modelin verilere iyi uyduğu ve tahminlerin genellikle düşük hata ile yapıldığı söylenebilir. Ancak, bazı durumlarda yüksek bir MSE, özellikle aykırı değerlerin etkisi altında kalan örneklerde daha büyük hatalar olduğunu gösterebilir. Bu durumu dikkate alarak veri setinden aykırı değerleri çıkartmanın uygun olacağına karar verildi.

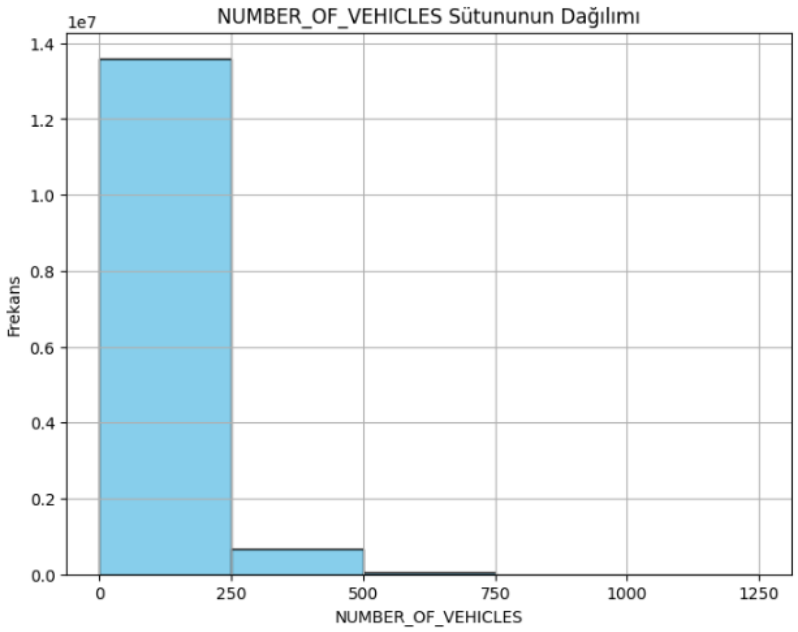
Modelin hata oranını düşürebilmek adına veri setindeki aykırı değerleri çıkartmanın modele etkisini görebilmek adına öncelikle Şekil 4.4'teki gibi aykırı değerler görüntülendi.

```
plt.figure(figsize=(6, 4))
plt.boxplot(dataset['NUMBER_OF_VEHICLES'], vert=False)
plt.title('Box Plot - Number of Vehicles')
plt.show()
```



Şekil 4.4 Veri setindeki aykırı değerlerin görselleştirilmesi

Araç sayılarının dağılımı Şekil 4.5'deki gibi görüntüleniyor. Neredeyse veri setinin tamamı 0 - 250 arasında bir değer almaktadır.



Şekil 4.5: NUMBER_OF_VEHICLES parametresinin dağılımı

Söz konusu aykırı değerler alt ve üst sınırlar hesaplanarak ve daha sonra veri seti içerisinde filtrelenerek Şekil 4.6'daki gibi bulunuyor.

```
Q1 = dataset['NUMBER_OF_VEHICLES'].quantile(0.25)
Q3 = dataset['NUMBER_OF_VEHICLES'].quantile(0.75)
IQR = Q3 - Q1

lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

outliers = dataset[(dataset['NUMBER_OF_VEHICLES'] < lower_bound) | (dataset['NUMBER_OF_VEHICLES'] > upper_bound)]
```

Şekil 4.6: Aykırı değerlerin bulunarak bir dataframe içerisine atılması

1 milyondan fazla aykırı değer olduğu Şekil 4.7'deki gibi görüntülendi. Bulunan aykırı değerler veri setinden Şekil 4.8'daki gibi çıkartıldı.

```
outliers.shape
```

```
(1086490, 16)
```

Şekil 4.7: Aykırı değerlerin toplam sayısı

```
dataset = dataset[~((dataset['NUMBER_OF_VEHICLES'] < lower_bound) | (dataset['NUMBER_OF_VEHICLES'] > upper_bound))]
```

Şekil 4.8: Veri setinden aykırı değerlerin çıkartılması

Aykırı değerlerin çıkartılması sonrası veri setinde toplamda 13.230.657 veri kaldığı Şekil 4.9'daki gibi gözlemlendi.

```
dataset.shape

(13230657, 16)
```

Şekil 4.9: Aykırı değerlerin çıkartılması sonrası datasette kalan veri bilgisi

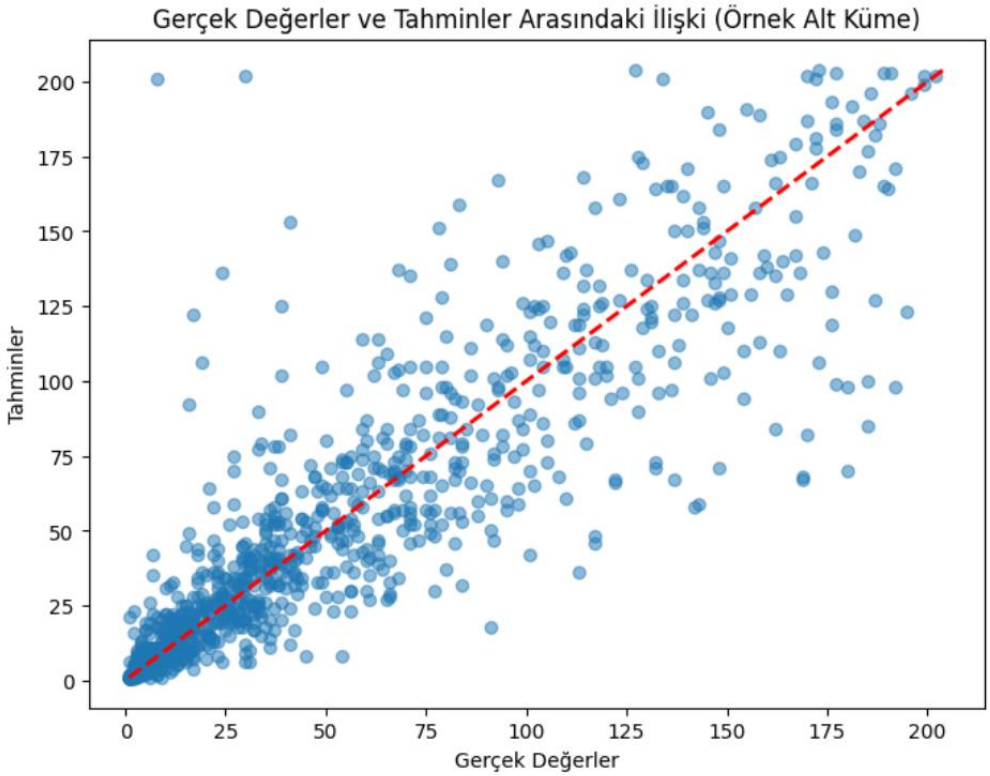
Aykırı değerlerin datasetten çıkartılması sonrası model Decision Tree algoritması ile tekrar eğitildiğinde elde edilen sonuçlar Şekil 4.10'deki gibi elde edildi.

```
R-squared: 0.7731080610360135
MAE: 13.76540374151151
MSE: 531.0299927088545
```

Şekil 4.10: Aykırı değerlerin datasetten çıkartılması sonrası Decision Tree metrikleri

İlk modelde R^2 değeri oldukça yüksekti (0,815). Aykırı değerleri çıkardıktan sonra bu değer biraz düştü, ancak hala oldukça yüksek (0,773). Aykırı değerlerin çıkarılması, modelin verilere daha iyi uymasını sağlamış olabilir. MAE ve MSE değerlerinde yarı yarıya belirgin bir düşüş gözlemlendi. Aykırı değerlerin çıkartılması bu hataları azaltmış olarak görünüyor. Genel olarak aykırı değerlerin çıkartılması modelin daha tutarlı ve iyi tahminler yapmasını sağladı olarak yorumlanabilir.

Veri setinde çok fazla veri bulunduğundan ve tüm noktaların tek bir grafikte görüntülenmesi zor olduğundan 1000 adet örneklem üzerinde tahmin grafiği Şekil 4.11'deki gibi görselleştirildi.



Şekil 4.11: Aykırı değerler çıkartıldıktan sonra 1000 örneklem üzerinde tahmin değerleri karşılaştırması

4.2 Polynomial Regression Algoritması ile Modelin Eğitilmesi

Veri setinde bağımlı değişken ile bağımsız değişkenler arasında doğrusal olmayan bir ilişki söz konusu olduğundan polinom regresyonunda nasıl bir sonuç vereceğini görebilmek adına model Şekil 4.12'deki gibi eğitildi.

```
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression

poly = PolynomialFeatures(degree=2)
X_train_poly = poly.fit_transform(X_train)
X_test_poly = poly.transform(X_test)


model = LinearRegression()
model.fit(X_train_poly, y_train)


y_pred = model.predict(X_test_poly)
```

Şekil 4.12: Polinom Regresyonu algoritması ile modelin eğitilmesi

Modelin metrikleri Şekil 4.13'deki gibi hesaplanmıştır. Veri setindeki varyansın yalnızca yaklaşık %49'una kadarın model tarafından açıklandığını gösteriyor. Modelin ortalama tahmin hatası 25,38 olarak hesaplanmış, iyi tahminler yapabildiği görülmektedir. Ancak model performansı düşük kalmıştır.

```
print("R-squared:", r2_score(y_test, y_pred))
print("MAE:", mean_absolute_error(y_test, y_pred))
print("MSE:", mean_squared_error(y_test, y_pred))
```

```
R-squared: 0.4942915402923719
MAE: 25.384416477561
MSE: 1183.5870454347562
```

Şekil 4.13: Polinom Regresyonu metrik sonuçları

4.3 Random Forest Algoritması ile Modelin Eğitilmesi

Regresyon ve sınıflandırma problemlerinde kullanılabilen Random Forest algoritmasının da vereceği sonucu gözlemleyebilmek adına model Şekil 4.14'deki gibi eğitildi.

```python
from sklearn.ensemble import RandomForestRegressor
rf_model = RandomForestRegressor(n_estimators=50, random_state=42)


rf_model.fit(X_train, y_train)


y_pred_rf = rf_model.predict(X_test)


print("R-squared:", r2_score(y_test, y_pred_rf))
print("MAE:", mean_absolute_error(y_test, y_pred_rf))
print("MSE:", mean_squared_error(y_test, y_pred_rf))
```

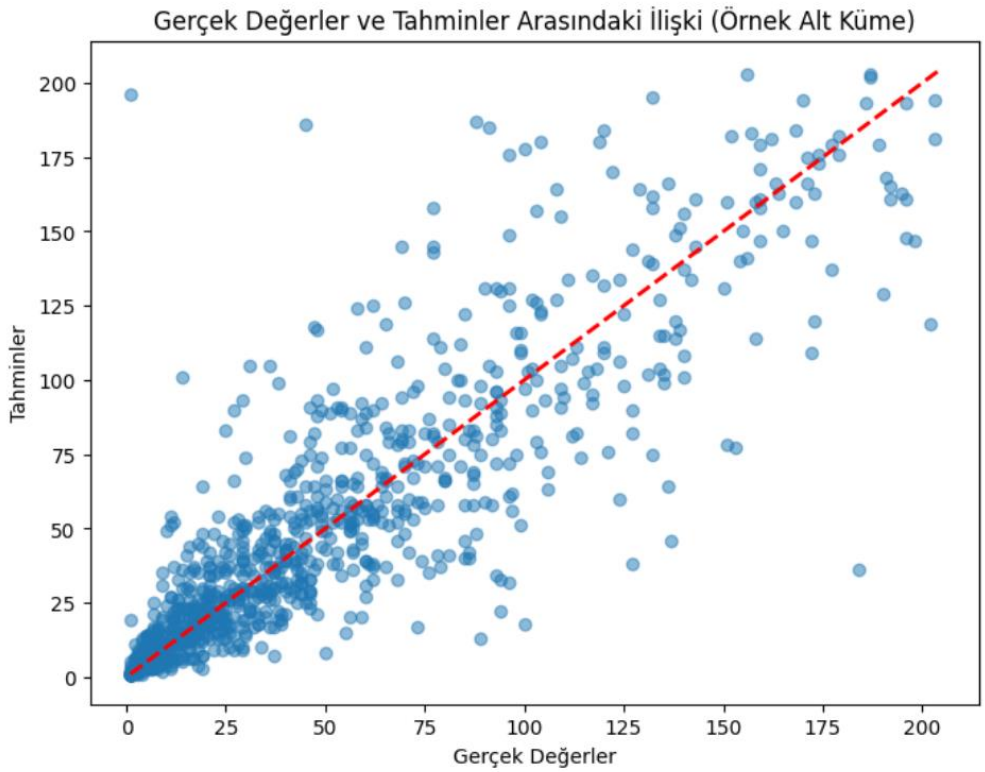
Şekil 4.14: Random Forest Regresyonu algoritması ile modelin eğitilmesi

Modelin metrikleri Şekil 4.15'deki gibi hesaplanmıştır. 0,89'luk bir R-kare değeri oldukça yüksek ve modelin veri setindeki değişikliğin %89'unu açıklayabildiğini gösterir. Ortalama Mutlak Hata (MAE) ve Ortalama Kare Hatası (MSE) de oldukça düşük olduğundan, modelin tahminlerinin gerçek değerlere oldukça yakın olduğu söylenebilir.

```python
print("R-squared:", r2_score(y_test, y_pred_rf))
print("MAE:", mean_absolute_error(y_test, y_pred_rf))
print("MSE:", mean_squared_error(y_test, y_pred_rf))
```

```
R-squared: 0.8888774033288883
MAE: 9.974192816785651
MSE: 260.0772507365972
```

Şekil 4.15: Random Forest metrik sonuçları



Şekil 4.16: Random Forest 1000 örneklem üzerinde tahmin değerleri karşılaştırması

4.4 LightGBM Regression Algoritması ile Modelin Eğitilmesi

Gradient Boosting kütüphanesini kullanarak, büyük veri kümeleri üzerinde düşük bellek kullanımı ile hızlı eğitime imkan tanıyan ve yüksek doğruluk sağlayabilecek olan LightGBM Regression algoritması ile model Şekil 4.17'deki gibi eğitildi.

```
from lightgbm import LGBMRegressor

lgbm_model = LGBMRegressor(n_estimators=100, learning_rate=0.1, max_depth=8, num_leaves=64)
lgbm_model.fit(X_train, y_train)


y_pred = lgbm_model.predict(X_test)
```

Şekil 4.17: LightGBM Regresyon algoritması ile modelin eğitilmesi

Model sonuçları Şekil 4.18'deki gibi bulunmuştur. R^2 değeri 0,73 olarak hesaplanmış olup, bağımsız değişkenlerin bağımlı değişkeni iyi şekilde açıkladığını ifade ediyor. Ortalama Mutlak Hata (MAE) değerinin 17 ve Ortalama Kare Hatası (MSE) değerinin de 639 olması kabul edilebilir değerler olarak yorumlanabilir.

```
print("R-squared:", r2_score(y_test, y_pred))
print("MAE:", mean_absolute_error(y_test, y_pred))
print("MSE:", mean_squared_error(y_test, y_pred))
```

```
[LightGBM] [Info] Auto-choosing row-wise multi-threading, the overhead of testing was 0.450164 seconds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 1025
[LightGBM] [Info] Number of data points in the train set: 9261459, number of used features: 13
[LightGBM] [Info] Start training from score 48.461853
R-squared: 0.726975841815002
MAE: 17.17996163603367
MSE: 639.0002985224314
```

Şekil 4.18: LightGBM Regresyon metrik sonuçları

4.5 K-Nearest Neighbour Algoritması ile Modelin Eğitilmesi

Regresyon problemlerinde de kullanılabilen K-Nearest Neighbours algoritması ile model Şekil 4.19'daki gibi eğitildi.

```
from sklearn.neighbors import KNeighborsRegressor

knn_model = KNeighborsRegressor(n_neighbors=5)
knn_model.fit(X_train, y_train)  #

y_pred_knn = knn_model.predict(X_test)
```

Şekil 4.19: K-Nearest Neighbour algoritması ile modelin eğitilmesi

Model sonuçları Şekil 4.20'deki gibi bulunmuştur. R^2 değeri 0,51 olarak hesaplanmış olup, elde edilen MAE ve MSE değerleri ile model performansı ancak orta seviyede denilebilir.

```
print("R-squared:", r2_score(y_test, y_pred_knn))
print("MAE:", mean_absolute_error(y_test, y_pred_knn))
print("MSE:", mean_squared_error(y_test, y_pred_knn))
```

```
R-squared: 0.5083557378983476
MAE: 22.73355534291815
MSE: 1150.6704474002063
```

Şekil 4.20: K-Nearest Neighbour metrik sonuçları

Bölüm 5

Sonuç

Bu çalışma, İstanbul'daki trafik yoğunluğunun tahmin edilmesi üzerine çeşitli regresyon ve ensemble öğrenme modellerinin performansını değerlendirmek amacıyla gerçekleştirildi. Büyük ve karmaşık yapıda veri içermesi sebebi ile derin öğrenme algoritmaları ile deneme yapılamadı. Farklı modellerin, aykırı değerlerin varlığı ve yokluğu durumunda nasıl performans gösterdiği incelendi. Elde edilen sonuçlar şu şekildedir:

- Aykırı değerlerin çıkartılmadığı senaryoda Decision Tree modeli, R-kare değerinde artış göstermiş olsa da, gerçek değerleri tahmin performansında iyileşme sağlamadı. Diğer modellere göre daha yüksek hata metrikleri ile sonuçlandı.
- Aykırı değerlerin çıkartıldığı senaryoda ise Decision Tree modeli, daha düşük R-kare değeri ile sonuçlanmasına rağmen gerçek değerleri tahmin performansında daha iyi sonuçlar verdi. Düşük hata metrikleri, modelin gerçek verilere daha yakın tahminler yapabildiğini gösterdi.
- Aykırı değerlerin çıkartılmasının model performansını iyi yönde etkilediği gözlendiğinden, sonraki modellerin tamamında aykırı değerler çıkartılarak devam edildi. Aykırı değer içermeyen tüm model sonuçları Şekil 5.1'deki gibidir.

Model	R-Kare	MAE	MSE
Decision Tree	0.77	13.77	531
KNN	0.51	22.73	1151
Polinom Regresyonu	0.49	25.38	1183
Random Forest	0.89	9.97	260
LightGBM	0.73	17.18	639

Şekil 5.1: Model sonuçlarının karşılaştırılması

- K-Nearest Neighbour ve Polinom regresyonu algoritması diğer modellere kıyasla en kötü sonuçları vermiş, veri setindeki yapıları açıklamakta yetersiz kalmışlardır.
- LightGBM algoritması orta düzeyde bir R-kare değeri ve ortalama hata metrikleri ile ortalama bir performans sergilemiştir, başarısız değildir.
- Random Forest algoritması modeller arasında hem yüksek R-kare değeri (0,89) hem de düşük hata metrikleri ile çok iyi bir performans sergilemiştir. Veri setindeki ilişkileri açıklamakta oldukça başarılı olduğu söylenebilir.

Söz konusu veri seti büyük ve karmaşık yapıda veri içerdiğinden, derin öğrenme modelleri gibi daha karmaşık yapılar ile çalışma yapılamadı.

Sonuç olarak, bu çalışma Random Forest gibi ensemble modellerinin trafik yoğunluğu tahmini için etkili bir seçenek olduğunu gösteriyor. Ancak, gelecekte daha geniş özellik setleri veya farklı model yapıları üzerine yapılan analizler, tahmin performansını iyileştirebilecektir.

Kaynaklar

Çetin Taş, İ., & Müngen, A. A. (2021). Yapay Sinir Ağları Ve Destek Vektör Makineleri Yöntemleri Ile Bölgesel Trafik Yoğunluk Tahmini. Adıyaman Üniversitesi Mühendislik Bilimleri Dergisi, 8(15), 378-390. https://doi.org/10.54365/adyumbd.971461

Çınaroğlu, S. (2017). Sağlik Harcamasının Tahmininde Makine Öğrenmesi Regresyon Yöntemlerinin Karşılaştırılması. Uludağ Üniversitesi Mühendislik Fakültesi Dergisi, 22(2), 179-200. https://doi.org/10.17482/uumfd.338805

Gök, M. (2017). Makine Öğrenmesi Yöntemleri Ile Akademik Başarinin Tahmin Edilmesi. Gazi University Journal of Science Part C: Design and Technology, 5(3), 139-148.

Nizam, H., Akın,S.S., 2014. Sosyal Medyada Makine Öğrenmesi ile Duygu Analizinde Dengeli ve Dengesiz Veri Setlerinin Performanslarının Karşılaştırılması, XIX. Türkiye'de İnternet Konferansı, İzmir

Utku, A. (2023). Deep Learning Based an Efficient Hybrid Model for Urban Traffic Prediction. Bilişim Teknolojileri Dergisi, 16(2), 107-117. https://doi.org/10.17671/gazibtd.1167140

Utku, A. (2023). Derin Öğrenme Tabanlı Trafik Yoğunluğu Tahmini: İstanbul İçin Bir Vaka Çalışması. Düzce Üniversitesi Bilim Ve Teknoloji Dergisi, 11(3), 1584-1598. https://doi.org/10.29130/dubited.1139534

Pekel, E. (2018). Farklı Makine Öğrenmesi Algoritmalarının Karşılaştırması (Yayın No. 494841) [Yüksek lisans tezi, Ondokuz Mayıs Üniversitesi]. YÖK Ulusal Merkezi. https://tez.yok.gov.tr/UlusalTezMerkezi/

Ş. İmre, & D. Çelebi, “Measuring comfort in public transport: a case study for İstanbul”, Transportation Research Procedia, 25, 2441-2449, 2017.

T. Kavzaoğlu, İ. Çölkesen, “Karar Ağaçları İle Uydu Görüntülerinin Sınıflandırılması: Kocaeli Örneği”,Harita Teknolojileri Elektronik Dergisi , vol. 2, no:1, pp. 36-45, 2010.

Ünal, Y. (2015). Makine öğrenmesi yöntemleriyle bel bölgesi rahatsızlıklarının tanısı, Doktora Tezi, Selçuk Üniversitesi

İstanbul Büyükşehir Belediyesi. Saatlik Trafik Yoğunluk Veri Seti. [İnternet]. İstanbul; 2023 [erişim tarihi 11.11.2023]. https://data.ibb.gov.tr/dataset/hourly-traffic-density-data-set

Wikipedia. Karar Ağacı. [İnternet]. İstanbul; 2023 [erişim tarihi 11.11.2023]. https://tr.wikipedia.org/wiki/Karar_a%C4%9Fac%C4%B1.