T. C.
SİVAS CUMHURİYET ÜNİVERSİTESİ
FEN FAKÜLTESİ

MAKİNE ÖĞRENMESİ VE WEB KAZIMA TEKNİKLERİ İLE İKİNCİ EL OTOMOBİL PİYASASINDA FİYAT TAHMİNİ

LİSANS TEZİ

Mehmet TANRIVERDİ

(2021165010)

İstatistik ve Bilgisayar Bilimleri Bölümü

Tez Danışmanı: Öğr. Gör. Nihal DUMAN SUNA

SİVAS

HAZİRAN 2025

Bu tez, Sivas Cumhuriyet Üniversitesi Senatosu'nun 20.08.2014 tarihli ve 7 sayılı kararı ile kabul edilen Fen Bilimleri Enstitüsü Lisansüstü Tez Yazım Kılavuzu (Yönerge)'nda belirtilen kurallara uygun olarak hazırlanmıştır.

ETİK

Sivas Cumhuriyet Üniversitesi Fen Bilimleri Enstitüsü, Tez Yazım Kılavuzu (Yönerge)'nda belirtilen kurallara uygun olarak hazırladığım bu tez çalışmasında;

- ✓ Bütün bilgi ve belgeleri akademik kurallar çerçevesinde elde ettiğimi,
- ✓ Görsel, işitsel ve yazılı tüm bilgi ve sonuçları bilimsel ahlak kurallarına uygun olarak sunduğumu,
- ✓ Başkalarının eserlerinden yararlanılması durumunda ilgili eserlere, bilimsel normlara uygun olarak atıfta bulunduğumu ve atıfta bulunduğum eserlerin tümünü kaynak olarak gösterdiğimi,
- ✓ Bütün bilgilerin doğru ve tam olduğunu, kullanılan verilerde herhangi bir değişiklik yapmadığımı,
- ✓ Tezin herhangi bir bölümünü, Sivas Cumhuriyet Üniversitesi veya bir başka üniversitede, bir başka tez çalışması olarak sunmadığımı; beyan ederim.

10.06.2025

Mehmet TANRIVERDİ

KATKI BELİRTME VE TEŞEKKÜR

Bilgi ve deneyimlerinden sürekli yararlandığım, tezin her aşamasında yardımlarını esirgemeyen danışman hocam Öğr. Gör. Nihal DUMAN SUNA'ya çok teşekkür ederim.

ÖZET

MAKİNE ÖĞRENMESİ VE WEB KAZIMA TEKNİKLERİ İLE İKİNCİ EL OTOMOBİL PİYASASINDA FİYAT TAHMİNİ

Mehmet TANRIVERDİ

Lisans Tezi

İstatistik ve Bilgisayar Bilimleri

Danışman: Öğr. Gör. Nihal DUMAN SUNA

2025, 42+xii

Günümüzde teknolojinin gelişmesiyle birlikte dijital ortamda üretilen veri miktarı büyük ölçüde artmıştır ve bu verilerin analiz edilmesi önemli bir ihtiyaç haline gelmiştir. Ancak, ihtiyaç duyulan veriler çoğu zaman tek bir merkezde derlenmiş halde sunulmamaktadır. Bu nedenle, veri biliminde yaygın olarak kullanılan web kazıma yöntemleri devreye girmektedir. Bu çalışmada, ikinci el otomobil piyasasına yönelik fiyat tahmini yapabilmek amacıyla, ilgili veriler BeautifulSoup kütüphanesi desteğiyle Python programlama dili kullanılarak toplanmıştır. Veri çekim süreci, zaman verimliliğini artırmak amacıyla asenkron programlama teknikleriyle yapılandırılmış ve binlerce otomobil ilanı eşzamanlı olarak elde edilmiştir. Elde edilen veriler üzerinde ön işleme adımları gerçekleştirilmiş, eksik ve aykırı değerler temizlenmiştir. Sonuç olarak oluşturulan veri seti, makine öğrenmesi modelleri ile analiz edilerek otomobil fiyatlarının tahmin edilmesinde kullanılmıştır. Analizde K-En Yakın Komşu Regresyon, Karar Ağaçları Regresyon, Rastgele Orman Regresyon, XGBoost Regresyon, Lineer Regresyon ve Yapay Sinir Ağları modelleri kullanılmıştır. Elde edilen analiz sonuçları Determinasyon Katsayısı (R^2), Ortalama Mutlak Hata (MAE), Ortalama Kare Hata (MSE) ve Kök Ortalama Kare Hata (RMSE) metrikler ile değerlendirilmiştir. Oluşturulan veri seti için sonuçlar incelendiğinde en iyi sonucu veren model 0,963 R^2, 66.878,508 MAE ve 147.386,12 RMSE değerleri ile XGBoost modeli olmuştur. Analiz sonucun en kötü sonuçları veren model Lineer Regresyon modeli olmuştur.

Anahtar kelimeler: Makine Öğrenmesi, Otomobil Fiyat Tahmini, Web Kazıma, BeautifulSoup, Python, Asenkron Programlama, MAE, MSE, RMSE, R^2, XGBoost

ABSTRACT

PRİCE PREDİCTİON İN THE SECOND-HAND CAR MARKET USİNG MACHİNE LEARNİNG AND WEB SCRAPİNG TECHNİQUES

Mehmet TANRIVERDİ

License Thesis

Department of Statistics and Computer Sciences

Thesis advisor: Lecturer Nihal DUMAN SUNA

2025, 42+xii

With the advancement of technology, the amount of data generated in the digital environment has increased significantly, and analyzing this data has become an important necessity. However, the required data is often not available in a single centralized location. Therefore, web scraping methods, which are widely used in data science, come into play. In this study, data related to the used car market was collected using the Python programming language with the support of the BeautifulSoup library to enable price prediction. The data collection process was structured using asynchronous programming techniques to increase time efficiency, and thousands of car listings were obtained simultaneously. Preprocessing steps were performed on the obtained data, and missing and outlier values were cleaned. The resulting dataset was analyzed using machine learning models and used to predict car prices. The analysis utilized K-Nearest Neighbor Regression, Decision Tree Regression, Random Forest Regression, XGBoost Regression, Linear Regression, and Artificial Neural Network models. The analysis results were evaluated using the Coefficient of Determination (R^2), Mean Absolute Error (MAE), Mean Squared Error (MSE), and Root Mean Squared Error (RMSE) metrics. When the results for the created dataset were examined, the XGBoost model yielded the best results with 0.963 R^2, 66,878.508 MAE, and 147,386.12 RMSE values. The Linear Regression model yielded the worst results in the analysis.

Key Words: Machine Learning, Car Price Prediction, Web Scraping, BeautifulSoup, Python, Asynchronous Programming, MAE, MSE, RMSE, R^2, XGBoost

İÇİNDEKİLER

ŞEKİLLER DİZİNİ

TABLOLAR DİZİNİ

KISALTMALAR DİZİNİ

ANN : Artificial Neural Network

CSV : Comma Separated Values

DT : Decision Tree

HTTP : Hyper Text Transfer Protocol

IDE : Integrated Development Environment

KNN : K-Nearset Neighbors

LR : Lineer Regresyon

MAE : Mean Absolute Error

MSE : Mean Squared Error

RFR : Random Forest Regressor

RMSE : Root Mean Squared Error

URL : Uniform Resource Loader

1. GİRİŞ

Gelişen teknoloji ile dijital ortamda üretilen veri miktarı ve veri trafiği her geçen gün artış göstermektedir. Bu durum, veri analizinin farklı sektörlerde ve alanlarda kritik bir rol oynamasına neden olmuştur. Doğru ve etkin bir şekilde yapılan veri analizi, işletmelerin daha isabetli kararlar almasına yardımcı olurken, beraberinde rekabet avantajı, müşteri memnuniyeti, operasyonel verimlilik ve risk yönetimi gibi birçok fayda da sağlamaktadır.

Ancak yalnızca veri analizine odaklanmak yeterli değildir; verinin doğru biçimde toplanması da en az analiz kadar önemlidir. Günümüzde ihtiyaç duyulan veriler çoğunlukla tek bir kaynakta toplanmış halde sunulmamaktadır. Bu nedenle, çeşitli web sitelerinde yer alan verilerin sistematik bir biçimde toplanması için web kazıma (web scraping) yöntemleri kullanılmaktadır.

Web siteleri büyük hacimli ve değerli veriler barındırır. Bu verilerin manuel olarak toplanması hem zaman alıcı hem de verimsizdir. Web kazıma teknikleri sayesinde, çok sayıda veriye otomatik olarak ulaşmak ve bu verileri yapılandırılmış biçimde analiz edilebilir hale getirmek mümkündür. Ancak her web sitesi farklı yapıda tasarlandığı için web kazıma işlemi çoğu zaman basit bir süreç değildir ve kullanılan kazıyıcılar web sitesinin yapısına göre özelleştirilmelidir.

Bu noktada, hangi verilerin hangi sitelerden çekileceğinin belirlenmesi sürecin en kritik adımlarından biridir. Python programlama dilinde yer alan BeautifulSoup kütüphanesi, web kazıma işlemleri için yaygın olarak tercih edilen güçlü araçlardandır. İhtiyaca uygun kütüphane seçimi, veri çekiminin başarısını doğrudan etkiler.

Veri çekimi tamamlandıktan sonra analiz aşamasına geçmeden önce çeşitli ön işleme adımları uygulanmalıdır. Veri seti üzerinde eksik veya hatalı değerler temizlenir, farklı ölçeklerdeki değişkenler normalize edilir ve gereksiz verilerden arındırılarak analiz için uygun bir yapı elde edilir. Sonrasında ise veri setinin özelliklerine ve hedeflenen çıktıya uygun olarak makine öğrenmesi, istatistiksel analiz veya veri madenciliği gibi yöntemlerle analiz süreci başlatılır.

Bu çalışma kapsamında, Türkiye' deki ikinci el otomobil piyasasında, otomobillerin teknik ve fiziksel özelliklerine göre fiyatlarının tahmin edilmesi amaçlanmıştır. Bu doğrultuda, gerekli veriler belirlenmiş ve web kazıma yöntemleriyle elde edilmiştir. Toplanan veriler ön işleme süreçlerinden geçirilmiş, ardından çeşitli makine öğrenmesi algoritmalarıyla analiz edilerek, tahmin modelleri geliştirilmiştir.

2. LİTERATÜR TARAMASI

Dere (2023), İstanbul Ticaret Üniversitesi' nde yaptığı yüksek lisans tezinde, Selenium test aracı ile 13.000 üzerinde ikinci el otomobil ilanından topladığı verilerle otomobil fiyatlarını tahmin etmeye çalışmıştır. Çalışmada Doğrusal Regresyon, Rastgele Orman (Random Forest), Gradyan Artırıcı (GBoosted), Ağaç Topluluğu (Tree Ensemble) ve Yapay Sinir Ağları (Artificial Neural Network) gibi yöntemler karşılaştırılmış ve en başarılı sonuç 0,85 R^2 ile Rastgele Orman (Random Forest) modeli elde etmiştir.

Yılmaz (2023), Sakarya Üniversitesi' nde yaptığı yüksek lisans tezinde, ikinci el otomobillere ait fiyat verilerini web kazıma yöntemleriyle elde etmiştir. Verileri Selenium ve BeautifulSoup kütüphaneleri kullanarak toplamış, aynı özelliklere sahip iki farklı veri seti oluşturmuştur. Veri seti-1 5557 gözlem ve 25 öznitelik, veri seti-2 ise 11688 gözlem ve yine 25 öznitelikten oluşmaktadır. Çalışmada, veri ön işleme sürecinde Lasso Regresyon ve Temel Bileşenler Analizi (PCA) ile boyut indirgeme yapılmış, hiperparametre optimizasyonu GridSearchCV yöntemiyle gerçekleştirmiştir. Tez kapsamında Random Forest, K-En Yakın Komşu (KNN), Gradyan Artırıcı (GBoosted), AdaBoost, SVR ve XGBoost regresyon modelleri kullanmıştır. Analiz sonuçlarına göre, birinci veri setinde en yüksek performans XGBoost Regresyon 0,973 R^2, ikinci veri setinde ise K-En Yakın Komşu Regresyon 0,978 R^2 modelleri tarafından elde edilmiştir.

Asilkan (2008), Akdeniz Üniversitesi' nde yaptığı doktora tezinde ikinci el otomobillerin hem güncel hem de gelecekteki fiyatlarının tahmini amacıyla veri madenciliği tekniklerine dayalı modeller geliştirmiştir. 2005-2007 yıllarına ait internetten toplanan otomobil ilan verileri kullanılarak oluşturulan veri seti, veri madenciliğine uygun hale getirilmiş ve çeşitli analizlere tabi tutulmuştur. Güncel fiyat tahmini için temel, opsiyonel ve zamanla değişen otomobil özellikleri ile fiyat arasındaki ilişki Regresyon analizi ve Yapay Sinir Ağları (ANN) kullanarak modellemiştir. Karşılaştırmalar sonucunda, ANN' nin Regresyon analizine kıyasla daha yüksek doğruluk sağladığı görülmüştür. Gelecekteki fiyat tahminleri için en çok tercih edilen marka ve modellerin üç yıllık verileri üzerinde zaman serisi analizi ile ANN uygulanmış; bu aşamada da ANN' nin üstün performans sergilediği tespit edilmiştir. Çalışmanın genel sonucunda, ikinci el otomobil fiyatlarının tahmininde ANN yönteminin başarılı bir yaklaşım sunduğu ifade edilmiştir.

Namlı, Ünlü ve Gül (2019), Türkiye' de satılan ikinci el otomobil fiyatlarının tahminine yönelik yürüttükleri vaka çalışmasında, makine öğrenmesi teknikleri ile doğrusal regresyon yöntemlerini karşılaştırmışlardır. Fiyat tahmini sürecinde Yapay Sinir Ağları (ANN), Destek Vektör Makineleri (SVM) ve Doğrusal Regresyon yöntemleri uygulamışlar. Modeller farklı değerlendirme kriterlerine göre kıyaslanmıştır. Elde edilen bulgulara göre, makine öğrenmesi algoritmalarının klasik doğrusal regresyon yöntemine kıyasla daha başarılı sonuçlar verdiği ve bu tekniklerle ikinci el otomobil piyasasında fiyatlandırma standardizasyonunun mümkün olduğu ortaya konmuştur.

Gülmez ve Kulluk (2023), çalışmalarında Türkiye' deki ikinci el otomobil piyasasına büyük veri analitiği ve makine öğrenmesi teknikleriyle inceleyerek otomobil fiyat tahmini gerçekleştirmişlerdir. Çeşitli otomobil özelliklerinin fiyat üzerindeki etkileri analiz edilmiş ve bu doğrultuda Doğrusal Regresyon, Karar Ağacı (Decision Tree), Rastgele Orman (Random Forest), GBT ve İzotonik Regresyon modelleri ve veri seti için Apache Spark kullanmışlardır. Kullanılan beş farklı model içinde fiyat tahmininde en yüksek başarı 0,887 R^2 değeriyle Rastgele Orman (Random Forest) modeli olmuştur.

Muti ve Yıldız (2023), çalışmalarında Türkiye' deki 2020 yılına ait ikinci el otomobil verileri üzerinde Doğrusal Regresyon modeli uygulayarak fiyat tahmini yapmışlardır. Veri setinin üçte biri test verisi olarak ayrıldığında, modelin R^2 skoru %73 olarak elde etmişler. Yazarlar, sonuçların geliştirilmesi için veri ön işleme süreçlerinin detaylandırılabileceğini belirtmişlerdir.

Özçalıcı (2017), çalışmasında ikinci el otomobil satış fiyatlarının tahmininde Karar Ağaçları (DecisionTree) ve Genetik algoritmaları birlikte kullanmıştır. Türkiye' deki bir e-ticaret sitesinden elde edilen 252.645 otomobil ilanı ve 139 değişken üzerinde yapılan çalışmada, Genetik algoritma ile en uygun değişken kombinasyonları seçilmiş ve Karar Ağaçları (Decision Tree) ile tahmin modelleri oluşturulmuştur. Farklı değişken sayılarına göre kurulan modellerde %65,67' ye kadar doğruluk oranı elde edilmiştir. Elde edilen bu yöntem, ikinci el otomobil piyasasında karar destek sistemi olarak kullanılabilecek potansiyele sahiptir.

Çelik ve Osmanoğlu (2019), çalışmalarında ikinci el otomobil fiyatlarının tahmini için doğrusal regresyon analizi ve makine öğrenmesi teknikleri kullanmışlardır. 5.041 araca ait 78 değişken arasında 23' ü seçilerek oluşturulan modelin açıklayıcılık oranı 0,891 R^2 olarak bulunmuş. Eğitim ve test veri setleriyle yapılan tahminlerde, %10' luk hata payı

sınırı içinde tahmin başarı oranı 0,8115 olarak elde edilmiştir. Sonuçlar makine öğrenmesi ile ikinci el otomobil fiyatlarının yüksek doğrulukta tahmin edilebileceğini göstermektedir.

Gültekin (2017), Pamukkale Üniversitesi' nde yaptığı yüksek lisans tezinde ikinci el otomobil piyasasında otomobil fiyatlarının tahminine yönelik olarak veri madenciliği teknikleri kullanmıştır. Çalışmada, Doğrusal Regresyon ve Yapay Sinir Ağları (ANN) yöntemlerini karşılaştırmıştır. Elde ettiği bulgular, ANN' nin ile yapılan tahminlerin sapma miktarlarının genellikle Doğrusal Regresyona göre daha düşük olduğunu göstermiştir. Bu sonuçlar, ikinci el otomobil fiyat tahmininde ANN yönteminin daha etkili olduğunu ortaya koymuştur.

3. MATERYAL VE YÖNTEM

Bu çalışmada web kazıma teknikleri kullanılarak web sitesinden eş zamanlı olarak verilerin toplanması, toplanan verilerin görselleştirilmesi ve makine öğrenmesi modelleri ile analiz edilerek fiyat tahmini yapılması hedeflenmiştir.

3.1 Materyal

Bu çalışmada kullanılan veriler, ikinci el otomobil satışı yapılan bir web sitesinden Python programlama dili ile yazılan kodlar yardımıyla toplanmış ve ardından bir veritabanına kaydedilerek analiz için hazır hale getirilmiştir.

3.1.1 Web Kazımada Python Kullanımı

Python veri alanında en yaygın kullanılan programlama dillerinden biri haline gelmiştir. Bunun başlıca sebepleri arasında anlaşılır ve sade sözdizimine sahip olmasıdır. Ayrıca, dünya çapında geniş bir kullanıcı topluluğuna sahip olması sayesinde, çeşitli kaynaklara ulaşmak ve topluluklardan destek almak oldukça mümkündür.

Bu avantajları sayesinde Python, özellikle web kazıma (web scraping) gibi veri toplama işlemleri için de yaygın olarak kullanılmaktadır. Web kazıma işlemlerinde verinin çekileceği internet sitesinin yapısı analiz edildikten sonra Python'un sunduğu zengin kütüphane desteği sayesinde hızlı ve etkili veri toplama algoritmaları geliştirilebilmektedir.

3.1.2 Visual Studio Code IDE

Visual Studio Code (VS Code), Python programlama dili olmak üzere birçok programlama diliyle uyumlu çalışabilen, açık kaynaklı ve çok yönlü bir IDE' dir. IDE (Tümleşik geliştirme ortamı) kullanıcıların daha hızlı ve daha verimli kod yazmasına yardımcı olan bir ortamdır. Metin düzenleme, eklenti desteği, terminal entegrasyonu, hata ayıklama ve sürüm kontrolü gibi pek çok işlevi içerisinde barındırır. IDE, geliştirme hızının artırılmasına olanak tanır.

3.1.3 Jupyter Notebook

Bir web tarayıcısı aracılığıyla çalıştırılabilen, not defteri tabanlı bir sunucu-istemci uygulamasıdır. Kullanıcıların hem kod yazabildiği hem de açıklayıcı metinler, görseller ve çıktılarla birlikte çalışmaları belgeleyebildiği bir geliştirme ortamı sunar.

Blok mantığıyla çalışması sayesinde, kodun istenilen bölümlerinde değişiklikler yapılarak çıktıları gözleme imkânı sağlamsı sebebiyle çok fazla tercih edilmektedir.

3.1.4 Kullanılan Kütüphaneler

Bu çalışmada verilerin toplanması için web kazıma aşamasında Aiohttp, Asyncio ve BeautifulSoup kütüphaneleri kullanılmıştır. Web sitesine eş zamanlı olarak istek atmak için Aiohttp kütüphanesi kullanılmıştır.

Aiohttp kütüphanesi, Python'da asenkron HTTP istemcisi ve sunucusu oluşturmaya olanak sağlayan bir kütüphanedir. Özellikle çok sayıda HTTP isteğinin eşzamanlı olarak gönderilmesi gereken uygulamalarda, klasik senkron yöntemlere kıyasla çok daha yüksek verimlilik sunmaktadır. Asyncio kütüphanesi ile birlikte çalışarak zaman kaybını azaltır ve eş zamanlı veri çekme işlemlerinde performansı artırmaktadır.

Asyncio kütüphanesi, Python programlama dilinde asenkron programlama yapmak için kullanılan standart bir kütüphanedir. Bu sayede, özellikle G/Ç tabanlı işlemler (örneğin ağ istekleri, dosya okuma/yazma) sırasında işlem beklenmeden diğer görevlerin yürütülmesine olanak tanır.

BeautifulSoup kütüphanesi ise HTML dosyalarını işlemek için oluşturulmuş bir kütüphanedir. Web sitesinde istenilen yerin HTML kodlarını ayrıştırarak, hedeflenen veri bölümlerin kolayca çıkarılmasını sağlamaktadır.

Verilerin ön işleme, görselleştirme ve analiz aşmalarında temel olarak Numpy, Pandas, Scikit-learn ve Matplotlib kütüphaneleri kullanılmıştır. Numpy, diziler, matrisleri ve lineer cebir üzerinde çalışmayı kolaylaştırması ve matematiksel işlemleri gerçekleştirmedeki yeteneği sebebiyle başta veri bilimi olmak üzere istatistik ve matematik alanlarında da sıkça tercih edilen bir kütüphanedir.

Pandas, Python programlama dili için geliştirilmiş, veri analizi ve veri işleme alanlarında yaygın olarak kullanılan güçlü bir kütüphanedir. Bir kaynak üzerindeki verilerin

okunması, işlenmesi, filtrelenmesi ve değişiklerin yapılması gibi amaçlar için kullanılır. Pandas kütüphanesi özellikle veri biliminde yaygın olarak kullanılmaktadır.

Scikit-learn, NumPy, SciPy ve Matplotlib kütüphaneleri üzerine inşa edilmiş, Python dili ile geliştirilen güçlü bir makine öğrenmesi kütüphanesidir. İçerisinde sınıflandırma, regresyon, kümeleme gibi pek çok istatistiksel modelleme algoritmasını hazır olarak barındırır. Veri bilimi projelerinde yaygın bir şekilde kullanılmaktadır.

Matplotlib, Python programlama dilinde kullanılan en temel ve yaygın görselleştirme kütüphanesidir. Bu kütüphane ile veriyi ifade edecek çizgi grafiği, çubuk grafik, pasta grafiği, histogram gibi birçok grafiklerin oluşturabilmesi sağlanmaktadır.

3.1.5 Veritabanı

Web kazıma yöntemleriyle elde edilen veriler, gerekli ön işlemlerden geçirilerek uygun formata dönüştürülmüş ve ardından tablo yapısında MySQL veritabanına kaydedilmiştir. Açık kaynak kodlu ve ücretsiz olan MySQL, veri yönetimi için tercih edilmiştir. Bu süreçte Python programlama dili kullanılarak, veritabanı ile etkileşim mysql.connector kütüphaneleri aracılığıyla sağlanmıştır. Bu kütüphaneler sayesinde Python ile veritabanına bağlanılmış ve veri ekleme, silme, güncelleme gibi temel SQL işlemleri başarıyla gerçekleştirilmiştir.

3.2 Yöntem

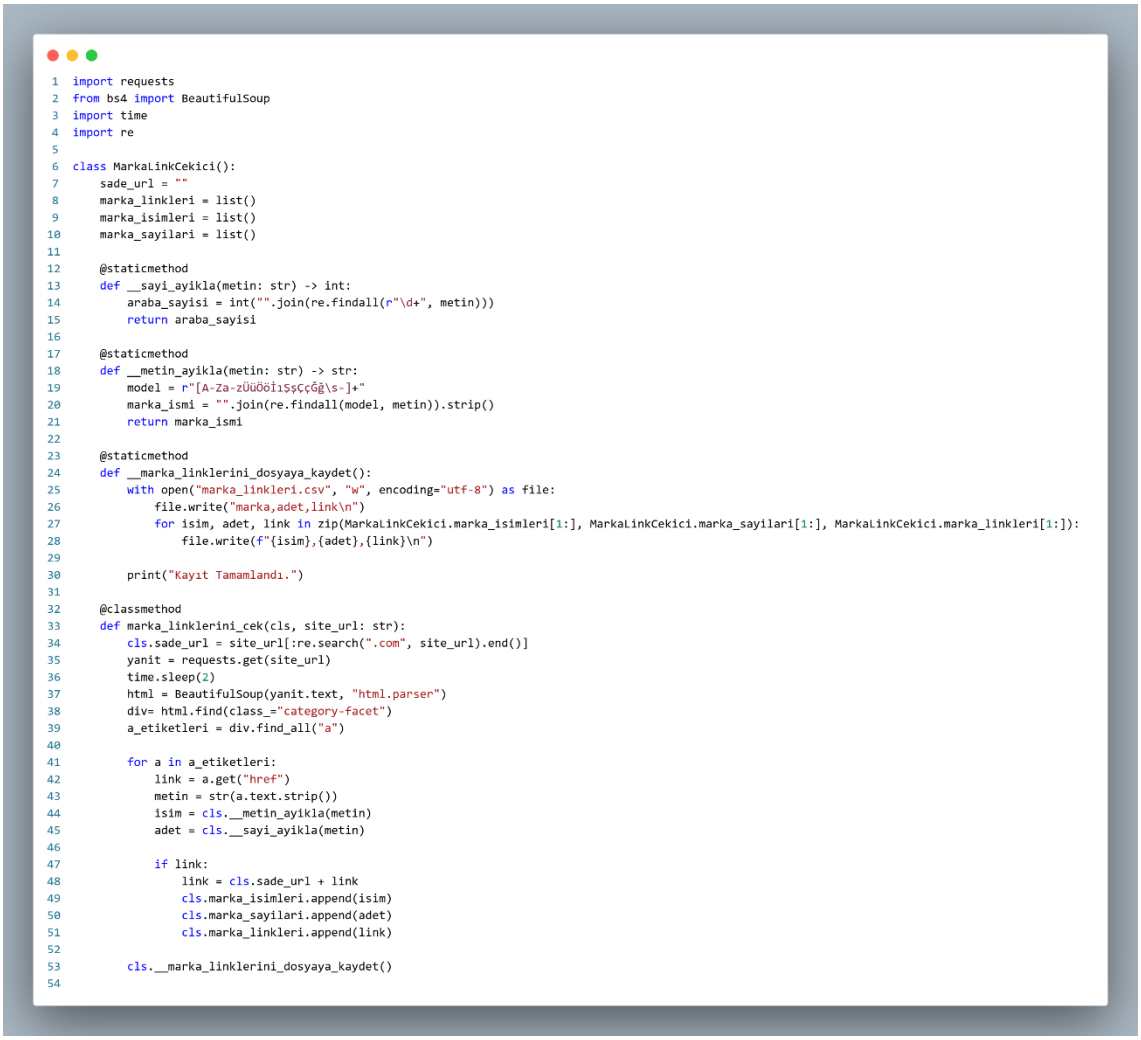
Çalışma kapsamında ikinci el otomobil sitesinden belirtilen kriterler doğrultusunda veriler toplanmış ve analiz için uygun hale getirilmiştir. Çalışma aşağıdaki adımlar izlenerek gerçekleştirilmiştir:

- Her bir marka ait marka adı, adet bilgisi ve bağlantı (link) adresleri toplanmıştır.
- Elde edilen marka bağlantılarına göre, ilgili otomobillere ait detay sayfa linkleri çekilmiştir.
- Her otomobile ait bağlantı üzerinden, teknik ve donanım özellikleri alınmıştır.
- Toplanan otomobil özellikleri, tablolar halinde bir veritabanına kaydedilmiştir.
- Oluşturulan veri setine, veri ön işleme adımları uygulanmıştır.
- Ön işleme sonrası veri görselleştirme yapılmıştır.
- Eğitim ve test veri setleri oluşturulmuştur.
- Belirlenen makine öğrenmesi modelleri çalıştırılmıştır.

- Son olarak, modellerin tahmin performansları karşılaştırılmış ve değerlendirilmiştir.

3.2.1 Verinin Toplanması

Türkiye' de aktif olarak hizmet veren bir ikinci el otomobil ilan platformundan, Mayıs 2025 tarihine ait güncel veriler çekilmiştir. Otomobillere ait detay bilgiler üç aşamalı bir süreç ile toplanmıştır. İlk aşamada, sitede listelenen her bir otomobil markasına ait bağlantılar (URL'ler) elde edilmiş ve kaydedilmiştir. İkinci aşamada, bu bağlantılar kullanılarak yaklaşık 50.755 otomobil ilanına ait detay sayfası bağlantısı toplanmış ve kaydedilmiştir. Son aşamada ise, her bir otomobil ilanının detay sayfası ziyaret edilerek otomobillere ait marka, seri, model, yıl, kilometre, vites tipi, yakıt tipi, kasa tipi, renk, motor hacmi, motor gücü, değişen sayısı, boyalı sayısı, kimden bilgisi ve fiyat gibi temel özellikler çıkarılmış ve yapılandırılmış bir biçimde veritabanına kaydedilmiştir.



```python
import requests
from bs4 import BeautifulSoup
import time
import re

class MarkaLinkCekici():
    sade_url = ""
    marka_linkleri = list()
    marka_isimleri = list()
    marka_sayilari = list()

    @staticmethod
    def __sayi_ayikla(metin: str) -> int:
        araba_sayisi = int("".join(re.findall(r"\d+", metin)))
        return araba_sayisi

    @staticmethod
    def __metin_ayikla(metin: str) -> str:
        model = r"[A-Za-zÜüÖöİı\şŞçÇğĞ\s-]+"
        marka_ismi = "".join(re.findall(model, metin)).strip()
        return marka_ismi

    @staticmethod
    def __marka_linklerini_dosyaya_kaydet():
        with open("marka_linkleri.csv", "w", encoding="utf-8") as file:
            file.write("marka,adet,link\n")
            for isim, adet, link in zip(MarkaLinkCekici.marka_isimleri[1:], MarkaLinkCekici.marka_sayilari[1:], MarkaLinkCekici.marka_linkleri[1:]):
                file.write(f"{isim},{adet},{link}\n")

        print("Kayıt Tamamlandı.")

    @classmethod
    def marka_linklerini_cek(cls, site_url: str):
        cls.sade_url = site_url[:re.search(".com", site_url).end()]
        yanit = requests.get(site_url)
        time.sleep(2)
        html = BeautifulSoup(yanit.text, "html.parser")
        div= html.find(class_="category-facet")
        a_etiketleri = div.find_all("a")

        for a in a_etiketleri:
            link = a.get("href")
            metin = str(a.text.strip())
            isim = cls.__metin_ayikla(metin)
            adet = cls.__sayi_ayikla(metin)

            if link:
                link = cls.sade_url + link
                cls.marka_isimleri.append(isim)
                cls.marka_sayilari.append(adet)
                cls.marka_linkleri.append(link)

        cls.__marka_linklerini_dosyaya_kaydet()

```

Şekil 3. 1 Marka bilgilerini çekmeye yönelik Python sınıfı

Şekil 3.1’ de verilen kod bloğunda, otomobil ilan sitesinde yer alan marka isimleri, bu markalara ait otomobil sayıları ve sayfa bağlantılarının (URL) elde edilmesini sağlayan bir Python sınıfı yer almaktadır. “MarkaLinkCekici” adlı bu sınıf, temel olarak ilgili sayfanın HTML içeriğini requests kütüphanesi ile talep eder ve BeautifulSoup yardımıyla ayrıştırır. Site yapısında marka bilgilerini içeren bölüm “category-facet” tespit edilerek içerisindeki her bir bağlantı etiketi “<a>” üzerinden marka adı, otomobil sayısı ve detay sayfasına yönlendiren bağlantı çıkarılır.

Bu işlemlerin ardından her markaya karşılık gelen isim, adet ve link bilgisi liste yapılarında toplanmıştır. Elde edilen veriler, “__marka_linklerini_dosyaya_kaydet()” methodu aracılığıyla “marka_linkleri.csv” adlı dosyaya yazıldı ve ilerleyen aşamalarda kullanılmak üzere kaydedilmiştir.

```python
import aiohttp
import asyncio
import pandas as pd
from bs4 import BeautifulSoup
import math
import re
import time

class ArabaLinkToplayici():
    def __init__(self, marka_dosyasi: str, site_url: str):
        self.marka_dosyasi = marka_dosyasi
        self.site_url = site_url
        self.sade_url = self.site_url[:re.search(".com", self.site_url).end()]
        self.sayfa_basi_url = 50
        self.oturum = None
        self.marka_linkleri = pd.read_csv(self.marka_dosyasi)
        self.semafor = asyncio.Semaphore(40)
        self.tum_linkler = list()


    async def baslat(self):
        baslangic_zamani = time.time()
        async with aiohttp.ClientSession() as oturum:
            self.oturum = oturum
            gorevler = []

            for _, seri in self.marka_linkleri.iterrows():
                isim = seri["marka"]
                adet = seri["adet"]
                link = seri["link"]
                gorevler.append(asyncio.create_task(self.__marka_araba_linkleri_cek(isim, adet, link)))
            await asyncio.gather(*gorevler)

            with open("tum_araba_linkleri.csv", "w", encoding="utf-8") as file:
                file.write("link\n")
                for link in self.tum_linkler:
                    file.write(link + "\n")
                print(f"Toplam {len(self.tum_linkler)} araba linki kaydedildi.")

        bitis_zamani = time.time()
        gecen_sure = bitis_zamani - baslangic_zamani
        print(f"İşlem tamamlandı. Yaklaşık olarak geçen süre: {(gecen_sure // 60) } dakika.")


    def __html_ayristir(self, html):
        satirlar = html.find_all("tr", class_="listing-list-item")
        for satir in satirlar:
            a = satir.find("a", class_="link-overlay")
            if a:
                link = a.get("href")
                if link:
                    link = self.sade_url + link
                    self.tum_linkler.append(link)
        del satirlar, a


    async def __marka_araba_linkleri_cek(self, marka_ismi: str, araba_adeti: int, marka_linki: str):
        async with self.semafor:
            if araba_adeti > 2500:
                toplam_sayfa = 50
            else:
                toplam_sayfa = math.ceil(araba_adeti / self.sayfa_basi_url)

            for sayfa in range(1, toplam_sayfa + 1):
                url = f"{marka_linki}?take=50&page={sayfa}"
                try:
                    async with self.oturum.get(url, timeout=15) as yanit:
                        if yanit.status == 200:
                            html_text = await yanit.text()
                            html = BeautifulSoup(html_text, "html.parser")
                            self.__html_ayristir(html)
                            del html_text, html
                        else:
                            print(f"{marka_ismi} - {sayfa}. sayfa hatali. Status: {yanit.status}")
                    await asyncio.sleep(0.3)
                except Exception as e:
                    print(f"{marka_ismi} - {sayfa}. sayfada hata: {e}")

```

Şekil 3. 2 Otomobil ilanı bağlantılarının asenkron olarak toplanması

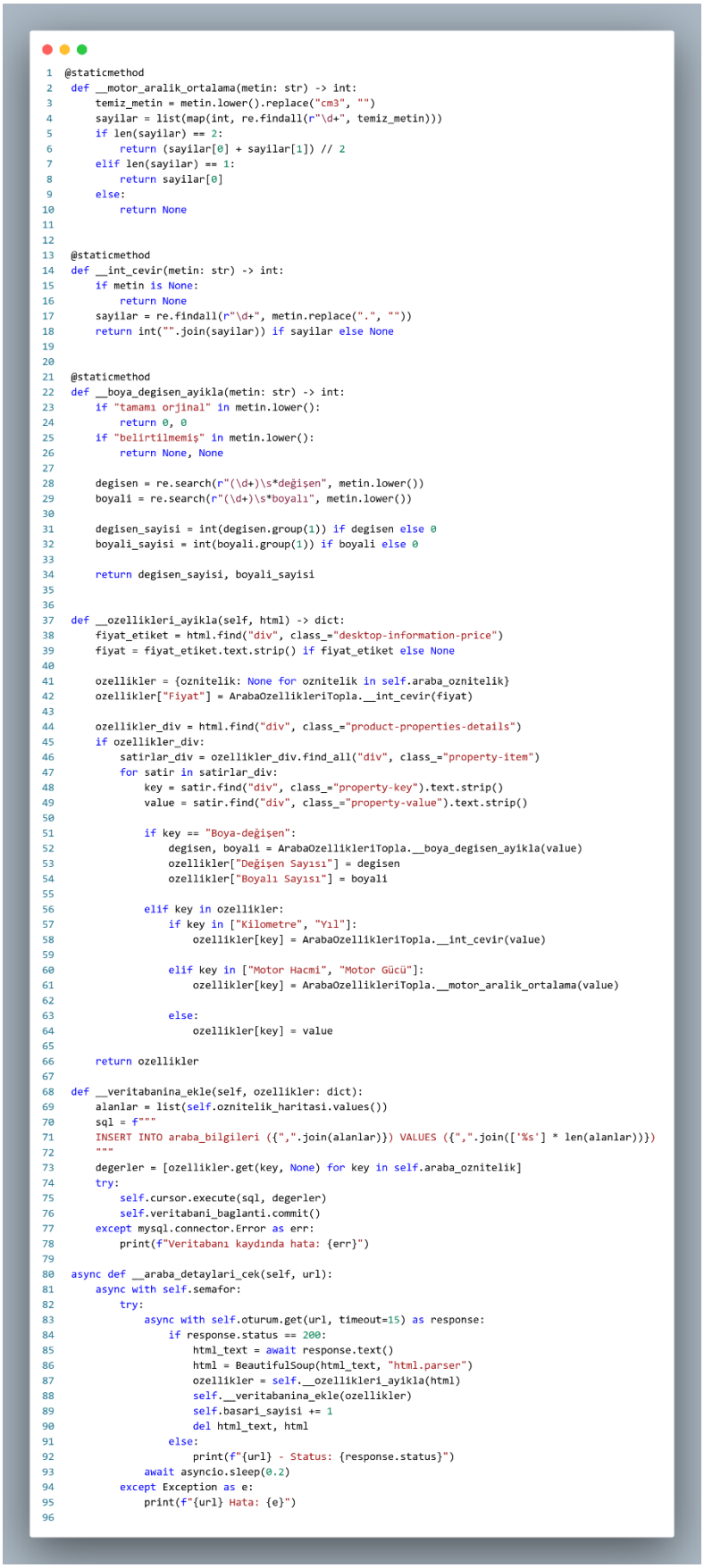
Şekil 3.2’ de yer alan Python sınıfı, marka bağlantılarını kullanarak her bir markaya ait ikinci el otomobil ilanlarının linklerini toplamak amacıyla geliştirilmiştir. Bu işlemde, daha önce oluşturulan “marka_linkleri.csv” dosyasındaki bağlantılar temel alındı ve her markanın sayfa yapısı asenkron olarak taranmıştır. Veri çekme işlemini hızlandırmak için aiohttp ve asyncio kütüphaneleriyle eş zamanlı istekler gönderilmiştir. Sayfalarda yer alan ilanlar, BeautifulSoup kütüphanesiyle ayrıştırıldı ve ilgili bağlantılar belirlenerek "tum_araba_linkleri.csv" adlı dosyaya kaydedilmiştir. Aynı anda çok sayıda istekte bulunmamak ve sunucuyu zorlamamak adına, sınıfa Semaphore uygulandı ve her isteğin ardından kısa bir bekleme süresi eklendi. Bu sayede yaklaşık 50.000’ den fazla otomobil ilanına ait bağlantı hızlı ve düzenli bir şekilde elde edilmiştir.

```python
import aiohttp
import asyncio
from bs4 import BeautifulSoup
import pandas as pd
import time
import re
import mysql.connector

class ArabaOzellikleriTopla():
    def __init__(self, araba_linkleri_dosyasi):
        self.araba_linkleri_dosyasi = araba_linkleri_dosyasi
        self.araba_linkleri = pd.read_csv(self.araba_linkleri_dosyasi)
        self.oturum = None
        self.semafor = asyncio.Semaphore(40)
        self.oznitelik_haritasi = {
            "Marka":"marka",
            "Seri":"seri",
            "Model":"model",
            "Yıl":"yil",
            "Kilometre":"kilometre",
            "Vites Tipi":"vites_tipi",
            "Yakıt Tipi":"yakit_tipi",
            "Kasa Tipi":"kasa_tipi",
            "Renk":"renk",
            "Motor Hacmi":"motor_hacmi",
            "Motor Gücü":"motor_gucu",
            "Değişen Sayısı":"degisen_sayisi",
            "Boyalı Sayısı":"boyali_sayisi",
            "Kimden":"kimden",
            "Fiyat":"fiyat"
        }
        self.araba_oznitelik = list(self.oznitelik_haritasi.keys())
        self.basari_sayisi = 0
        self.veritabani_baglanti = mysql.connector.connect(
            host="localhost",
            user="root",
            password="Mehmet123.",
            database="araba_db"
        )
        self.cursor = self.veritabani_baglanti.cursor()


    async def baslat(self):
        baslangic_zamani = time.time()
        async with aiohttp.ClientSession() as oturum:
            self.oturum = oturum
            gorevler = []
            for _, seri in self.araba_linkleri.iterrows():
                link = seri["link"]
                gorevler.append(asyncio.create_task(self.__araba_detaylari_cek(link)))

            await asyncio.gather(*gorevler)

        self.cursor.close()
        self.veritabani_baglanti.close()

        bitis_zamani = time.time()
        gecen_sure = bitis_zamani - baslangic_zamani
        print(f"İşlem tamamlandı. Geçen süre: {gecen_sure // 60} dakika.")
        print(f"Toplam başarılı çekilen ilan sayısı: {self.basari_sayisi}")


```

Şekil 3. 3 Otomobillere ait özellikleri çekmeye yönelik Python sınıfı (I. Bölüm)

Şekil 3.3'te yer alan "ArabaOzellikleriTopla" adlı Python sınıfı, her bir otomobil ilanının detay sayfasına ulaşılarak ilgili teknik özelliklerinin toplanmasını sağlayan yapıyı içermektedir. Daha önce elde edilmiş olan ilan bağlantılarını içeren CSV dosyası giriş olarak alınmakta ve aiohttp ile tanımlanan oturum üzerinden her bağlantı eş zamanlı (asenkron) şekilde ziyaret edilmiştir. Bu sınıf içinde tanımlı "baslat()" metodu, veri toplama işlemini başlatmakta ve her bağlantı için "__araba_detaylari_cek()" adlı özel metodun görev olarak kuyruklanmasını sağlanmıştır.

Her başarılı bağlantıdan alınan HTML içeriği BeautifulSoup kütüphanesi yardımıyla ayrıştırılmakta ve otomobil detaylarına ilişkin nitelikler çıkarılmıştır. Çekilen nitelikler; marka, seri, model, yıl, kilometre, vites tipi, yakıt tipi, kasa tipi, renk, motor hacmi, motor gücü, değişen sayısı, boyalı sayısı, kimden bilgisi ve fiyat gibi özellikleri kapsamaktadır. Bu nitelikler "oznitelik_haritasi" adlı sözlük ile veritabanındaki alanlara eşlenmekte ve her başarılı veri kaydı sonrasında sayaç artırılmaktadır. Tüm veriler çekildikten sonra MySQL bağlantısı kapatılarak işlem tamamlanmıştır. Asenkron yapının içine tanımlanan Semaphore sayesinde aynı anda en fazla 40 isteğin gönderilmesine izin verilerek ağ trafiği dengelenmiştir.



```python
@staticmethod
def __motor_aralik_ortalama(metin: str) -> int:
    temiz_metin = metin.lower().replace("cm3", "")
    sayilar = list(map(int, re.findall(r"\d+", temiz_metin)))
    if len(sayilar) == 2:
        return (sayilar[0] + sayilar[1]) // 2
    elif len(sayilar) == 1:
        return sayilar[0]
    else:
        return None


@staticmethod
def __int_cevir(metin: str) -> int:
    if metin is None:
        return None
    sayilar = re.findall(r"\d+", metin.replace(".", ""))
    return int("".join(sayilar)) if sayilar else None


@staticmethod
def __boya_degisen_ayikla(metin: str) -> int:
    if "tamamı orjinal" in metin.lower():
        return 0, 0
    if "belirtilmemiş" in metin.lower():
        return None, None

    degisen = re.search(r"(\d+)\s*değişen", metin.lower())
    boyali = re.search(r"(\d+)\s*boyalı", metin.lower())

    degisen_sayisi = int(degisen.group(1)) if degisen else 0
    boyali_sayisi = int(boyali.group(1)) if boyali else 0

    return degisen_sayisi, boyali_sayisi


def __ozellikleri_ayikla(self, html) -> dict:
    fiyat_etiket = html.find("div", class_="desktop-information-price")
    fiyat = fiyat_etiket.text.strip() if fiyat_etiket else None

    ozellikler = {oznitelik: None for oznitelik in self.araba_oznitelik}
    ozellikler["Fiyat"] = ArabaOzellikleriTopla.__int_cevir(fiyat)

    ozellikler_div = html.find("div", class_="product-properties-details")
    if ozellikler_div:
        satirlar_div = ozellikler_div.find_all("div", class_="property-item")
        for satir in satirlar_div:
            key = satir.find("div", class_="property-key").text.strip()
            value = satir.find("div", class_="property-value").text.strip()

            if key == "Boya-değişen":
                degisen, boyali = ArabaOzellikleriTopla.__boya_degisen_ayikla(value)
                ozellikler["Değişen Sayısı"] = degisen
                ozellikler["Boyalı Sayısı"] = boyali

            elif key in ozellikler:
                if key in ["Kilometre", "Yıl"]:
                    ozellikler[key] = ArabaOzellikleriTopla.__int_cevir(value)

                elif key in ["Motor Hacmi", "Motor Gücü"]:
                    ozellikler[key] = ArabaOzellikleriTopla.__motor_aralik_ortalama(value)

                else:
                    ozellikler[key] = value

    return ozellikler

def __veritabanina_ekle(self, ozellikler: dict):
    alanlar = list(self.oznitelik_haritasi.values())
    sql = f"""
    INSERT INTO araba_bilgileri ({",".join(alanlar)}) VALUES ({",".join(['%s'] * len(alanlar))})
    """
    degerler = [ozellikler.get(key, None) for key in self.araba_oznitelik]
    try:
        self.cursor.execute(sql, degerler)
        self.veritabani_baglanti.commit()
    except mysql.connector.Error as err:
        print(f"Veritabanı kaydında hata: {err}")

async def __araba_detaylari_cek(self, url):
    async with self.semafor:
        try:
            async with self.oturum.get(url, timeout=15) as response:
                if response.status == 200:
                    html_text = await response.text()
                    html = BeautifulSoup(html_text, "html.parser")
                    ozellikler = self.__ozellikleri_ayikla(html)
                    self.__veritabanina_ekle(ozellikler)
                    self.basari_sayisi += 1
                    del html_text, html
                else:
                    print(f"{url} - Status: {response.status}")
            await asyncio.sleep(0.2)
        except Exception as e:
            print(f"{url} Hata: {e}")

```

Şekil 3. 4 Otomobillere ait özellikleri çekmeye yönelik Python sınıfı (II. Bölüm)

Şekil 3.4' te, "ArabaOzellikleriTopla" sınıfının veri ayrıştırma ve işleme sürecine ait metotlar yer almaktadır. Sayfa içeriğinden elde edilen metinler, "__ozellikleri_ayikla()" fonksiyonu ile analiz edilmekte ve her bir nitelik özel kurallara göre işlenmiştir. Örneğin, motor hacmi veya motor gücü gibi alanlar bazen aralık şeklinde verildiğinden "__motor_aralik_ortalama()" metodu yardımıyla ortalaması alınmaktadır. Benzer şekilde kilometre, yıl ve fiyat gibi sayısal değerler "__int_cevir()" fonksiyonu ile metin içinden temizlenerek tam sayı formuna dönüştürülmüştür.

İlanlarda sıkça yer alan "Boya-Değişen" bilgisi, birden fazla bilgiyi aynı anda içerdiğinden, bu değerler "__boya_degisen_ayikla()" fonksiyonu ile hem değişen hem de boyalı parça sayısı olarak ayrı ayrı çıkarılmıştır. Elde edilen tüm özellikler, veritabanı şemasına uygun olacak şekilde "__veritabanina_ekle()" metodu ile "araba_bilgileri" adlı tabloya eklenmiştir. Bu işlem sırasında hata oluşması durumunda kullanıcı bilgilendirilmiştir.

Asenkron olarak çalışan "__araba_detaylari_cek()" metodu, her ilan linkini ziyaret ederek içerik çeker, ardından ilgili bilgileri ayrıştırır ve veritabanına kaydeder. Bu süreç, yaklaşık 50.755 otomobil ilanı için başarıyla gerçekleştirildi ve analiz süreci için kapsamlı bir veri kümesi oluşturulmuştur.

3.2.1.1 Veri Seti

Bu çalışmada oluşturduğumuz veri seti, Mayıs 2025 dönemine ait Türkiye'deki aktif bir ikinci el otomobil ilan sitesinden elde edilen 50.000' den fazla otomobil ilanından oluşmaktadır. Toplamda 50.755 adet otomobile ait veri toplanmış ve bu veriler 15 değişken içerecek şekilde yapılandırılmıştır. Veriler, MySQL veritabanına kaydedilmiş ve daha sonra analizlerde kullanılmak üzere CSV formatına dönüştürülmüştür.

Veri setinde bulunan değişkenlerin isimleri ve açıklamaları:

- Marka: Otomobilin markası
- Seri: Otomobilin serisi veya alt modeli
- Model: Otomobilin model tipi
- Yıl: Otomobilin üretim yılı
- Kilometre: Otomobilin o zamana kadar yaptığı toplam yol (km cinsinden)
- Vites Tipi: Otomobil vites türü

- Yakıt Tipi: Otomobilin yakıt tipi
- Kasa Tipi: Otomobilin kasa tipi
- Renk: Otomobilin dış rengi
- Motor Hacmi: Otomobilin motor hacminin ortalaması (cm3)
- Motor Gücü: Otomobilin motor gücünün ortalaması (hp)
- Değişen Sayısı: Otomobilde kaç adet değişen parça olduğunun sayısı
- Boyalı Sayısı: Otomobilde kaç adet boyalı parça olduğunun sayısı
- Kimden: İlan sahibinin türü
- Fiyat: Otomobilin satış fiyatı

Şekil 3.5' de ham veri setinin bir kısmı gösterilmiştir.

```
[19]: df.head(7)
```

[19]:

	id	marka	seri	model	yil	kilometre	vites_tipi	yakit_tipi	kasa_tipi	renk	motor_hacmi	motor_gucu	degisen_sayisi	boyali_sayisi	kimden	fiyat
0	1	Lexus	ES	300h Business Plus	2023.0	43000.0	Otomatik	Benzin	Sedan	Siyah	2250.0	213.0	1.0	1.0	Galeriden	4950000.0
1	2	Lexus	GS	200t Luxury	2023.0	25000.0	Otomatik	Benzin	Hatchback/5	Mavi	NaN	213.0	1.0	0.0	Galeriden	1265000.0
2	3	Lexus	LS	500h Exclusive	2021.0	23665.0	Otomatik	Benzin	Sedan	Siyah	3250.0	363.0	0.0	0.0	Galeriden	9000000.0
3	4	Kuba	City	NaN	2024.0	0.0	Otomatik	Elektrik	Hatchback/3	Kahverengi	NaN	50.0	0.0	0.0	Galeriden	279999.0
4	5	Lexus	GS	200t F Sport	2016.0	90500.0	Otomatik	Benzin	Sedan	Beyaz	1900.0	238.0	0.0	0.0	Galeriden	2745000.0
5	6	Infiniti	I30	3.0	1996.0	230000.0	Otomatik	LPG & Benzin	Sedan	Yeşil	2988.0	190.0	NaN	NaN	Sahibinden	225000.0
6	7	Infiniti	I30	3.0	1996.0	298000.0	Otomatik	LPG & Benzin	Sedan	Yeşil	2988.0	190.0	0.0	7.0	Sahibinden	475000.0

Şekil 3. 5 Veri setinin ilk 7 değeri

3.2.2 Veri Ön İşleme

Bu çalışmada, toplanan veri seti analiz ve modelleme aşamalarına geçilmeden önce çeşitli veri ön işleme adımlarından geçirilmiştir. Bu adımlar, eksik değerlerin giderilmesi, kategorik değişkenlerin dönüştürülmesi, sayısal değişkenlerin ölçeklendirilmesi ve veri setinin eğitim-test olarak ayrılması şeklinde adımlar uygulanmıştır. Veri setinde hangi sütunlar ve veri tipleri nelerdir şekil 3.6' da gösterilmiştir.

```
[8]: df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50755 entries, 0 to 50754
Data columns (total 16 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   id              50755 non-null  int64
 1   marka           50597 non-null  object
 2   seri            50597 non-null  object
 3   model           50424 non-null  object
 4   yil             50597 non-null  float64
 5   kilometre       50597 non-null  float64
 6   vites_tipi      50572 non-null  object
 7   yakit_tipi      50597 non-null  object
 8   kasa_tipi       50580 non-null  object
 9   renk            50596 non-null  object
 10  motor_hacmi     49128 non-null  float64
 11  motor_gucu      49094 non-null  float64
 12  degisen_sayisi  40339 non-null  float64
 13  boyali_sayisi   40339 non-null  float64
 14  kimden          50597 non-null  object
 15  fiyat           50597 non-null  float64
dtypes: float64(7), int64(1), object(8)
memory usage: 6.2+ MB
```

Şekil 3. 6 Veri seti veri tipi bilgisi

Şekil 3.6' da id sütunu veri hakkında herhangi bir bilgi içermediği için veri setinden çıkarılmıştır.

```
[18]: df.isnull().sum().sort_values(ascending=False)

[18]: degisen_sayisi    10416
      boyali_sayisi     10416
      motor_gucu         1661
      motor_hacmi        1627
      model               331
      vites_tipi          183
      kasa_tipi           175
      renk                159
      marka               158
      seri                158
      yil                 158
      kilometre           158
      yakit_tipi          158
      kimden              158
      fiyat               158
      dtype: int64
```

Şekil 3. 7 Eksik veri sayıları

Şekil 3.7’ de veri setinde yer alan her bir sütundaki eksik değerlerin dağılımı gösterilmiştir. Veri setinde toplam 26.074 adet eksik gözlem bulunduğu tespit edilmiştir. Analizlerin sağlıklı bir şekilde yürütülebilmesi adına, bu eksik değerlerin uygun yöntemlerle tamamlanması gerekmektedir.

Tüm değişkenleri eksik olan gözlemler veri setinden çıkarılmıştır. Bu işlem, analiz sürecinde anlamlı bilgi içermeyen ve model performansını olumsuz etkileyebilecek kayıtların veri setinden çıkarılması amacıyla gerçekleştirilmiştir.

```
[56]: kayip_veri_yuzdesel = df.isnull().mean() * 100
      kayip_veri_ozet = kayip_veri_yuzdesel[kayip_veri_yuzdesel > 0].sort_values(ascending=False)

      for sutun, yuzde in kayip_veri_ozet.items():
          print(f"{sutun}: %{yuzde:.2f} eksik")

degisen_sayisi: %20.27 eksik
boyali_sayisi: %20.27 eksik
motor_gucu: %2.97 eksik
motor_hacmi: %2.90 eksik
model: %0.34 eksik
vites_tipi: %0.04 eksik
kasa_tipi: %0.03 eksik
renk: %0.00 eksik
```

Şekil 3. 8 Eksik verilerin oranı

Şekil 3.8’ de eksik veriler yüzdesel olarak hesaplanmış. Yapılan değerlendirme sonucunda, toplam veri setinin %1’ inden daha az eksik değere sahip olan değişkenlerdeki eksik gözlemler veri kaybının ihmal edilebilir düzeyde olduğu kabul edilerek doğrudan veri setinden çıkarılmıştır. Bu yaklaşım veri bütünlüğünü büyük ölçüde korurken analiz sürecinin güvenirliğini arttırmaya amaçlamaktadır.

Motor hacmindeki eksik değerler, otomobillerin marka ve model bilgilerine göre gruplanarak her grup için medyan (ortanca) değeri kullanılarak doldurulmuştur. Bu yöntem, aynı marka ve modele sahip otomobillerin benzer teknik özelliklere sahip olacağı varsayımına dayanır. Medyan kullanılması ise, uç değerlerin etkisini minimize ederek daha sağlam ve temsil edici bir merkezi eğilim ölçüsü sunar. Böylece hem grup içi tutarlılık korunmuş hem de aşırı değerlerin veri bütünlüğünü bozması önlenmiştir.

Motor hacmi değişkenine benzer şekilde, motor gücü değişkenindeki eksik değerler de otomobillerin marka ve model bilgilerine göre gruplandırılmış ve her grup için hesaplanan medyan değer ile doldurulmuştur. Bu yaklaşım, aynı otomobil segmentine ait modellerin benzer motor performans özellikleri taşıdığı varsayımına dayanmaktadır.

Medyan tercih edilerek, uç değerlerin etkisi azaltılmış ve daha güvenilir bir tahminleme yapılması sağlanmıştır. Böylece eksik verilerin doldurulması sürecinde hem tutarlılık hem de istatistiksel sağlamlık korunmuştur.

Değişen sayısı ve boyalı sayısı değişkenlerindeki eksik değerler, ilk olarak marka, model ve yıl bilgilerine göre gruplanmış ve her grup için en sık görülen değer (mod) kullanılarak doldurulmuştur. Bu yöntem, aynı üretim yılına ait benzer otomobillerin geçmiş hasar durumlarının da benzer olabileceği varsayımıyla uygulanmıştır.

Ancak bazı gruplarda mod değerinin hesaplanamaması veya eksik değerlerin devam etmesi durumunda, ikinci bir işlem olarak veriler bu kez yalnızca marka ve modele göre gruplanmış ve her grup için mod değeri kullanılarak kalan eksik gözlemler tamamlanmıştır. Bu iki aşamalı yaklaşım sayesinde merkezi eğilim ölçütleri kullanılarak eksik veriler en doğru şekilde tahmin edilmeye çalışılmış ve veri setinin bütünlüğü korunmuştur.

Kilometre değişkeni için kutu grafiğine dayalı aykırı değer analizi yapılmıştır. Bu analiz sonucunda, istatistiksel olarak alt ve üst sınırların dışında kalan uç değerler veri setinden çıkarılarak temizlik işlemi gerçekleştirilmiştir. Bu sayede, analizlerin güvenilirliğini olumsuz etkileyebilecek aşırı değerlerin veri seti üzerindeki etkisi azaltılmıştır.

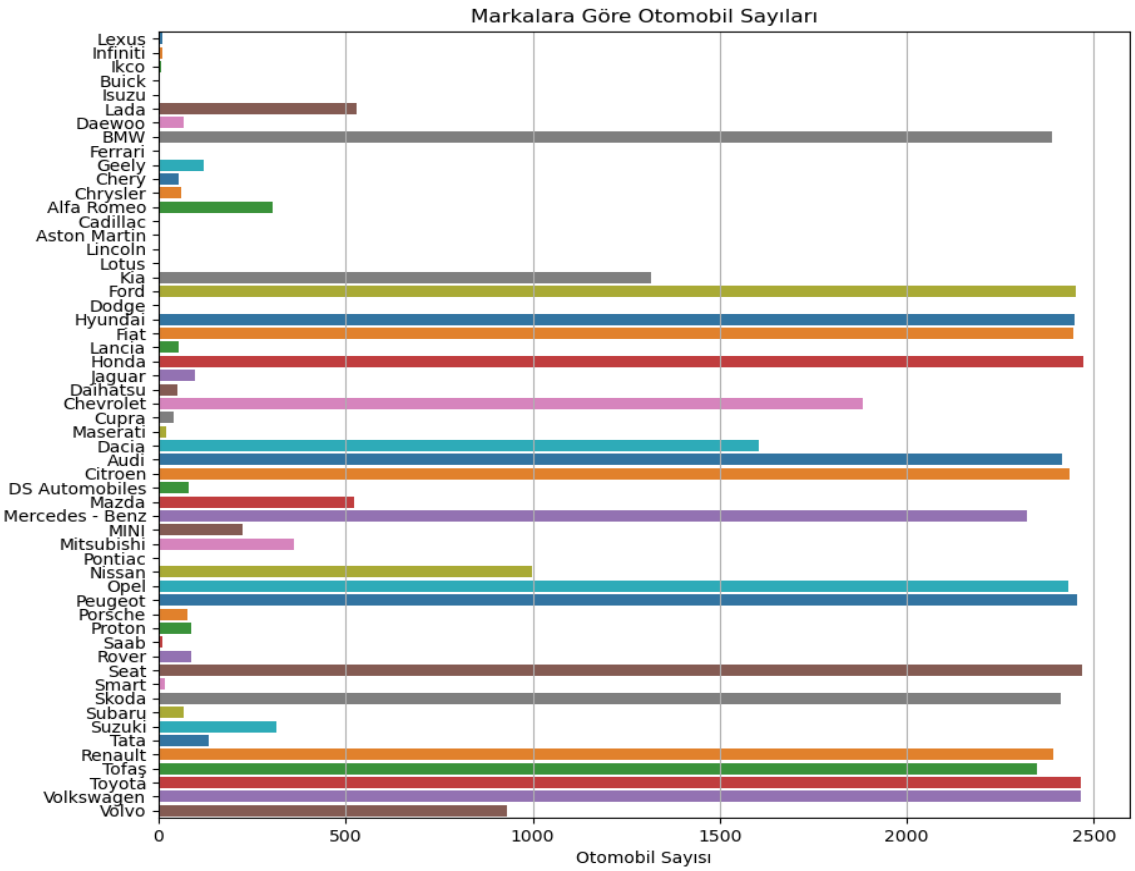
Motor gücü değişkeni için yapılan inceleme sonucunda, 400 beygir gücünün üzerindeki değerler uç değer olarak değerlendirilmiş ve veri setinden çıkarılmıştır. Bu işlem, veri setindeki olağan dışı yüksek motor gücü değerlerinin analiz sonuçlarını bozmasını engellemek ve daha tutarlı bir dağılım elde etmek amacıyla gerçekleştirilmiştir.

Fiyat değişkeninin dağılımı incelendiğinde, 15 milyon TL üzerindeki değerlerin veri setindeki genel eğilimin dışında kaldığı gözlemlenmiştir. Bu nedenle, 15 milyon TL' nin üzerindeki fiyatlar uç değer olarak değerlendirilmiş ve veri setinden çıkarılmıştır. Bu adım, analizlerin daha sağlıklı sonuçlar vermesi ve modelleme sürecinin aşırı uç değerlerden etkilenmemesi amacıyla uygulanmıştır.

3.2.3 Keşifsel Veri Analizi ve Görselleştirme

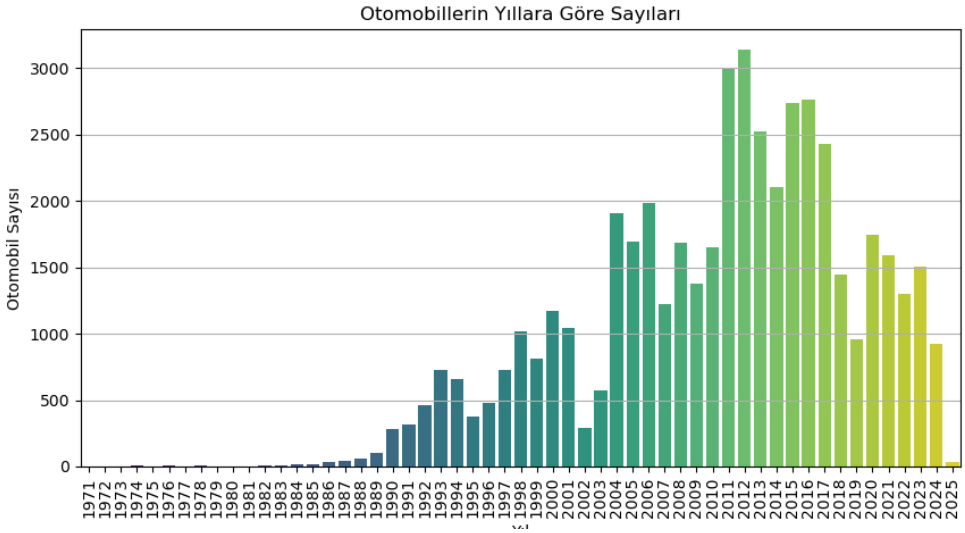
Bu bölümde, veri setine ilişkin genel eğilimlerin, dağılımların ve değişkenler arasındaki ilişkilerin incelendiği keşifsel veri analizi gerçekleştirilmiştir.

Şekil 3.9’ da, veri setinde yer alan otomobillerin markalarına göre sayısal dağılımı görselleştirilmiştir.



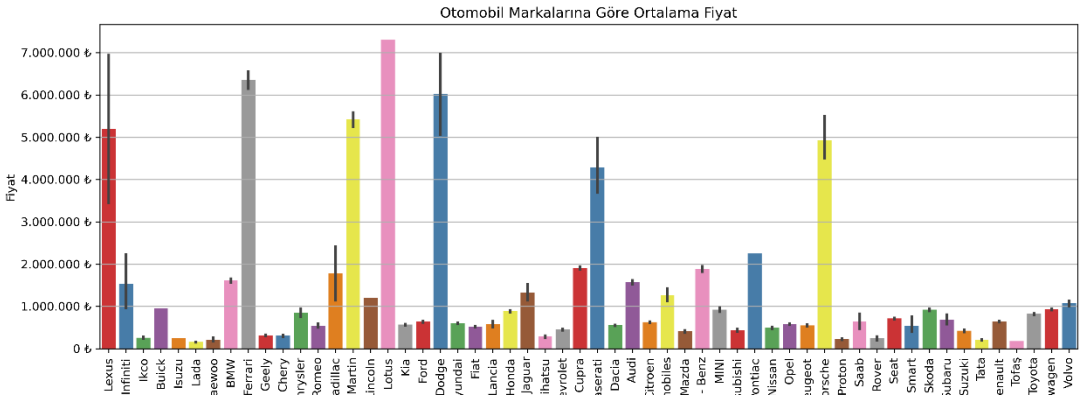
Şekil 3. 9 Markaların ilan sayı dağılımları

Şekil 3.10’ da, veri setindeki otomobillerin üretim yıllarına göre sayısal dağılımı görselleştirilmiştir. En fazla otomobil sayısının 2012 yılına ait olduğu gözlenmektedir.



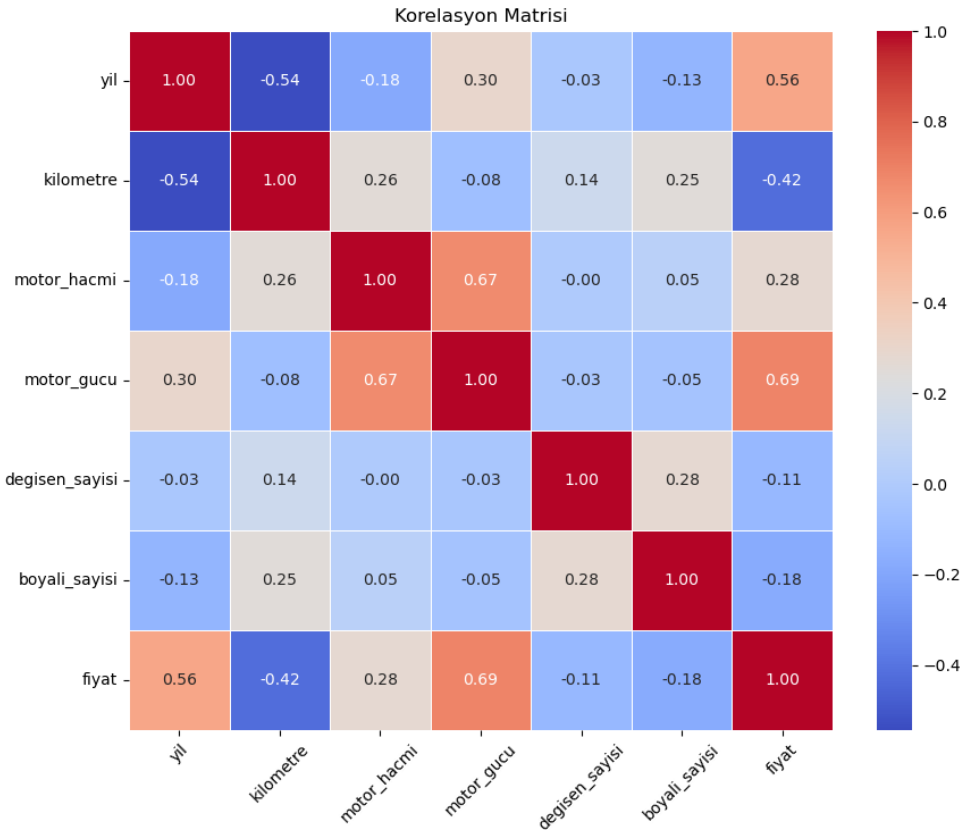
Şekil 3. 10 Otomobillerin üretim yıllarına göre sayısal dağılımı

Şekil 3.11' de, veri setindeki otomobil markalarına göre ortalama fiyatlar görselleştirilmiştir. Bu grafik, farklı markaların ortalama fiyat düzeylerini karşılaştırma olarak inceleme imkânı sunmaktadır.



Şekil 3. 11 Otomobillerin markalarına göre ortalama fiyatları

Şekil 3.12' de, veri setindeki sayısal değişkenler arasındaki ilişkileri gösteren korelasyon matrisi görselleştirilmiştir. Bu matris aracılığıyla, özellikle fiyat değişkeniyle yüksek korelasyona sahip olan değişkenler belirlenmiş ve bu değişkenler üzerinde daha ayrıntılı analizler gerçekleştirilmiştir.



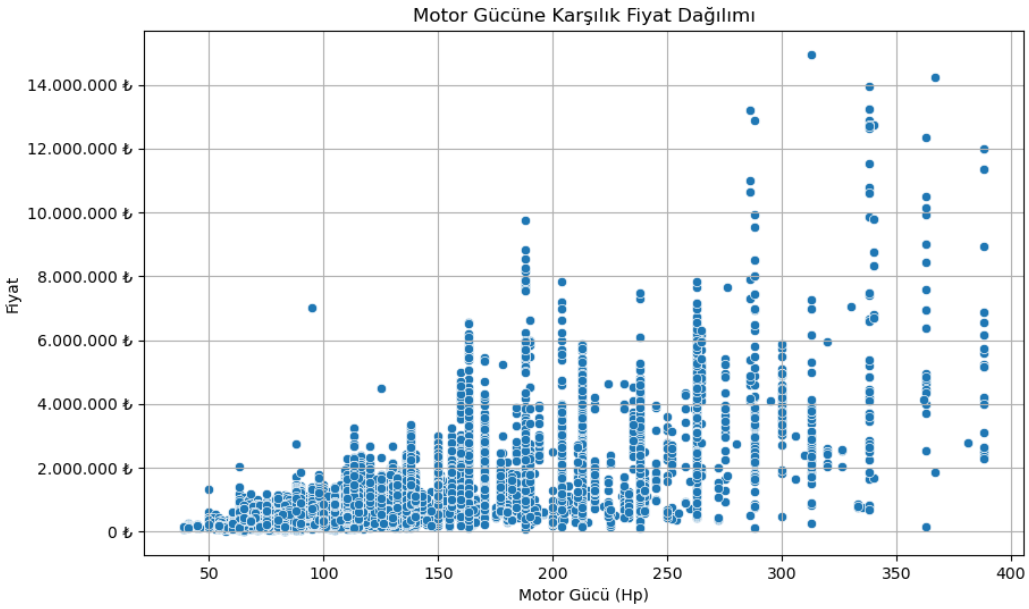
Şekil 3. 12 Korelasyon matrisi

Şekil 3.12' de, korelasyon matrisi incelendiğinde, fiyat değişkeniyle en yüksek korelasyona sahip üç değişkenin: motor gücü 0,69, üretim yılı 0,56 ve kilometre değişkeni de -0,42 korelasyona sahip olduğu görülmektedir.

Bu sonuçlar, otomobilin motor performansının ve üretim yılının fiyat üzerinde pozitif, kilometrenin ise negatif yönde etkili olduğu gösterilmektedir.

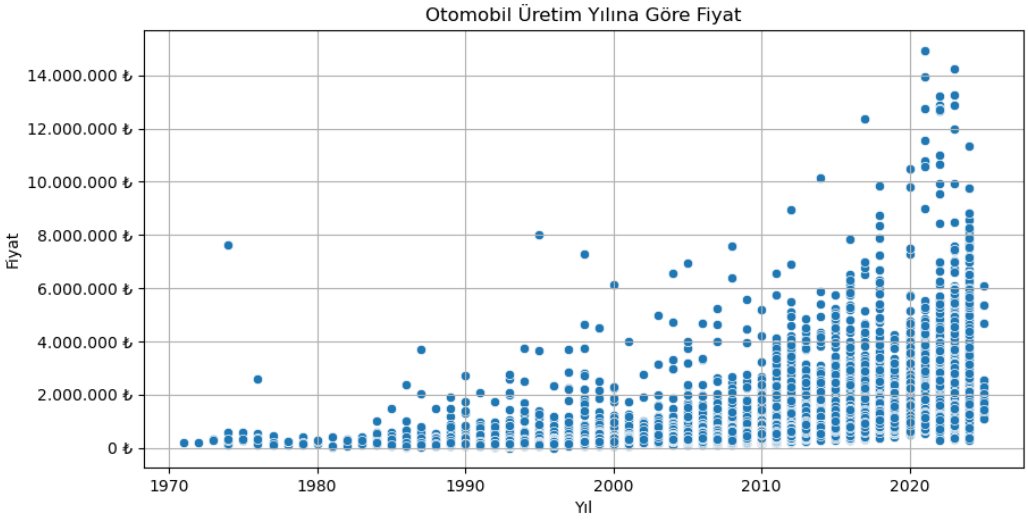
Ayrıca, motor gücü ile motor hacmi arasında 0,67 oranında güçlü bir pozitif korelasyon bulunurken, motor gücü ile üretim yılı arasında da 0,30 düzeyinde orta şiddette bir pozitif ilişki tespit edilmiştir.

Şekil 3.13' te, motor gücü ile fiyat değişkeni arasındaki ilişki görselleştirilmiştir. Grafik incelendiğinde, motor gücü arttıkça otomobil fiyatının da genel olarak yükseldiği gözlemlenmektedir. Bu durum, motor gücünün fiyat üzerinde önemli bir belirleyici olduğunu ve aralarındaki ilişkin pozitif yönde olduğunu desteklemektedir. Bu bulgu, korelasyon analizinde elde edilen yüksek korelasyon katsayısı 0,69 ile tutarlıdır.



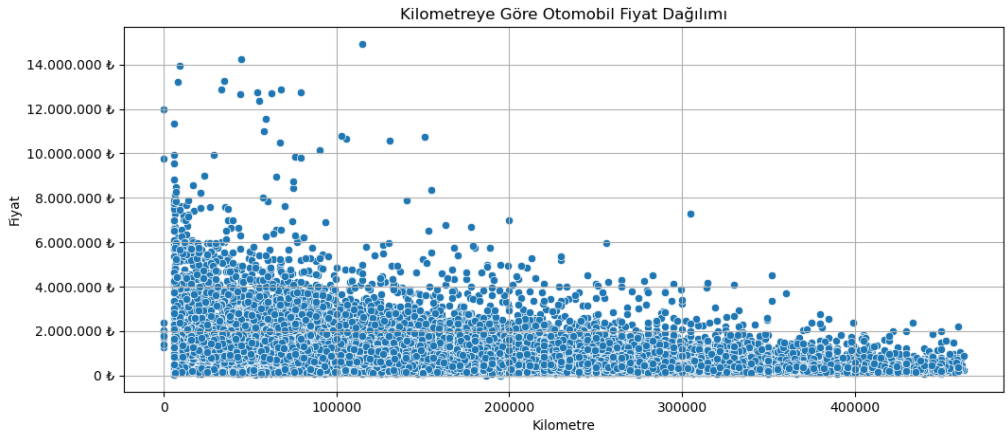
Şekil 3. 13 Motor gücü fiyat dağılımı

Şekil 3.14' te, otomobillerin üretim yıllarına göre fiyat dağılımı görselleştirilmiştir. Grafik incelendiğinde, genel olarak yeni model otomobillerin daha yüksek fiyatlara sahip olduğu açıkça görülmektedir. Özellikle son yıllarda üretilen otomobillerin ortalama fiyatlarının belirgin şekilde arttığı gözlenmektedir. Bu durum, üretim yılı ile fiyat arasında korelasyon katsayısı 0,56 pozitif bir ilişki olduğunu ve model yılı ilerledikçe otomobillerin piyasa değerinin artığını göstermektedir.



Şekil 3. 14 Otomobil üretim yılı ile fiyat arasındaki ilişki

Şekil 3.15' te, kilometre ile fiyat değişkeni arasındaki ilişki görselleştirilmiştir. Grafik incelendiğinde, kilometre arttıkça otomobil fiyatlarının genellikle azaldığı gözlenmektedir. Bu durum, kilometre ile fiyat arasında negatif bir ilişki olduğunu göstermektedir. Korelasyon katsayısının -0,42 olması da bu ilişkinin orta düzeyde ve ters yönlü olduğunu ortaya koymaktadır. Yani otomobillerin kullanım süresi arttıkça piyasa değerlerinde belirgin bir düşüş yaşanmaktadır.



Şekil 3. 15 Kilometre fiyat dağılımı

3.2.4 Eğitim ve Test Verilerinin Ayrılması

Modelleme sürecine başlamadan önce, veri seti rastgele karıştırılmış, böylece veri sıralamasından kaynaklanabilecek olası önyargılar engellenmiştir. Ardından, kategorik değişkenler uygun kodlama yöntemleriyle sayısal formata dönüştürülmüştür. Bu işlemlerden sonra veri seti, modellerin performansını değerlendirmek amacıyla %80 eğitim ve %20 test olacak şekilde ikiye ayrılmıştır. Son olarak, değişkenler üzerinde standartlaştırma işlemi uygulanarak, tüm değişkenlerin aynı ölçek aralığında olması sağlanmıştır. Bu adımlar, modelin öğrenme sürecini iyileştirmek ve daha tutarlı sonuçlar elde etmek amacıyla gerçekleştirilmiştir.

3.2.5 Search Yöntemleri

Makine öğrenmesi modellerinde en iyi sonuçları elde etmek için hiperparametrelerin doğru ayarlanması gerekir. Bu amaçla kullanılan iki yöntem GridSearchCV ve RandomizedSearchCV'dir.

3.2.5.1 GridSearchCV (Grid Search Cross-Validation)

GridSearchCV, makine öğrenmesi modellerinde en uygun hiperparametre kombinasyonlarını belirlemek için kullanılan sistematik bir arama yöntemidir. Modelin başarı performansını artırmak amacıyla farklı hiperparametre değerleri denenerek, her kombinasyon çapraz doğrulama (cross-validation) yöntemiyle değerlendirilir. Böylece eğitim verisi üzerinde aşırı uyum (overfitting) yapmadan, genellenebilir en iyi parametre kümesi seçilir.

GridSearchCV, özellikle karmaşık modellerin optimizasyonu sürecinde önem kazanır. Kullanıcı, her hiperparametre için denenecek değerleri belirtir ve GridSearchCV bu değerlerin tüm kombinasyonlarını dener. Bu işlem, tüm parametre uzayını taradığı için kapsamlı ve sistematik bir yaklaşımdır.

3.2.5.2 RandomizedSearchCV (Random Search Cross-Validation)

RandomizedSearchCV, makine öğrenmesi modellerinde hiperparametre optimizasyonu yapmak için kullanılan, rastgele örneklemeye dayalı bir arama yöntemidir. GridSearchCV'nin tüm parametre kombinasyonlarını denemesi yerine, RandomizedSearchCV kullanıcı tarafından belirlenen sayıda rastgele kombinasyon

seçerek modeli değerlendirir. Bu sayede hem hesaplama süresi kısalır hem de yüksek boyutlu parametre alanlarında verimli sonuçlar elde edilebilir.

RandomizedSearchCV, özellikle parametre uzayının çok geniş olduğu durumlarda zaman ve kaynak tasarrufu sağlar. Her parametrenin aralığı veya dağılımı kullanıcı tarafından belirlenir ve bu değerlerden rastgele örnekler çekilerek çapraz doğrulama yoluyla test edilir. Bu yöntem, sınırlı kaynaklarla iyi bir hiperparametre tahmini elde etmek isteyen uygulamalarda tercih edilir.

GridSearchCV, her kombinasyonu tarayarak en iyiyi garanti etmeye çalışırken; RandomizedSearchCV, daha hızlı fakat olasılığa dayalı bir yaklaşık en iyi çözüm sunar.

3.2.6 Makine Öğrenmesi Modelleri

Çalışmada 6 farklı makine öğrenmesi modeli kullanılmıştır. Modellerde en iyi parametrelerin belirlenmesi için GridSearchCV ve RandomizedSearchCV kullanıldı.

Makine öğrenmesi modellerinin performansını değerlendirmek için bu çalışmada aşağıdaki metrikler kullanılmıştır:

- R-Kare (R-Squared): Bağımsız değişken tarafından açıklanan varyansın toplam varyansa oranını ifade eder. Modelin ne kadar iyi uyum sağladığını gösterir.
- Ortalama Mutlak Hata (Mean Absolute Error): Gerçek ve tahmin edilen değerler arasındaki ortalama mutlak farkı ifade eder.
- Ortalama Kare Hata (Mean Squared Error): Gerçek ve tahmin edilen değerler arasındaki ortalama karesel farkı ifade eder.
- Kök Ortalama Kare Hata (Root Mean Squared Error): MSE' nin kareköküdür. Gerçek ve tahmin edilen değerler arsındaki ortalama karesel farkın kareköküdür.

3.2.6.1 K-En Yakın Komşu

K-En Yakın Komşu (K-Nearset Neighbors - KNN) modeli, makine öğrenimi alanında sınıflandırma ve regresyon problemlerini çözmek için kullanılan basit ve popüler bir algoritmadır. Gözetimli öğrenme yöntemlerindendir. Temelde, bir veri noktasının sınıfını tahmin etmek veya bir değeri öngörmek için en yakın komşuların etrafındaki veri noktalarının etiketlerini veya değerlerini kullanılır.

KNN modeli, belirli bir veri noktasına en yakın k adet komşuyu belirleyerek bu komşuların değerlerinin ortalamasını alarak tahminleme yapan bir makine öğrenmesi yöntemidir. Komşuluk hesaplamaları çoğunlukla Euclidean mesafesi tercih edilmekle birlikte Manhattan, Minkowski ve Hamming gibi farklı mesafe ölçütleri de kullanılabilmektedir.

KNN algoritması, sade yapısı ve kolay uygulanabilirliğiyle öne çıkan bir yöntemdir. Model, eğitim süreci gerektirmeden doğrudan veri üzerinden tahmin gerçekleştirir. Ancak her tahmin sırasında tüm veri kümesiyle karşılaştırma yaptığı için büyük veri setlerinde yavaş çalışabilir. Ayrıca, işlem sırasında tüm veriyi bellekte tutması gerektiğinden yüksek hafıza kullanımı gerektirebilir (Altman, 1992).

3.2.6.2 Karar Ağaçları

Karar ağaçları (Decision Trees), gözetimli öğrenme kapsamında yer alan ve hem sınıflandırma hem de regresyon problemlerinde yaygın olarak kullanılan etkili modellerden biridir. Bu yöntem, veri kümesini dallara ayırarak hedef değişkenin tahmin edilmesini sağlar. Modelin yapısı, karar verme sürecini temsil eden bir ağaç yapısına benzer şekilde; kök düğüm, iç düğümler, dallar ve yaprak düğümlerden oluşur.

Modelin temel çalışma prensibi, veriyi belirli özniteliklere ve eşik değerlerine göre ardışık biçimde alt parçalara ayırmaktır. Her bir iç düğümde, veri kümesinin hangi özniteliğe göre bölüneceğine karar verilirken, bu bölmenin ne kadar "bilgi" kazandırdığı hesaplanır. Özellikle sınıflandırma problemlerinde, bu amaçla bilgi kazancı kriteri sıklıkla kullanılır. Bilgi kazancı, her öznitelik için hesaplanan entropi farklarını ölçerek en yüksek bilgi artışını sağlayan özniteliği seçer. Böylece model, veriyi en iyi şekilde ayrıştıran değişkenler üzerinden yapılandırılır.

Regresyon problemlerinde ise bilgi kazancı yerine hedef değişkenin varyansını azaltan bölmeler tercih edilir. Her yaprak düğüm, belirli bir hedef değeri (örneğin ortalama) temsil eder ve bu değer, o bölgedeki örneklerin tahmininde kullanılır. Karar ağaçlarının sezgisel yapısı sayesinde model sonuçları kolayca yorumlanabilir.

Ancak karar ağaçlarının çok fazla dallanması, modelin eğitim verisine fazla uyum sağlamasına yani aşırı öğrenmeye (overfitting) yol açabilir. Bu durumu önlemek amacıyla daha küçük ve anlaşılabilir alt ağaçlara ayrılarak modelin sadeleştirilmesi, performansı ve yorumlanabilirliği artırabilir (Quinlan, 1986).

3.2.6.3 Rastgele Orman

Rastgele Ormanlar (Random Forests), karar ağaçlarının topluluk (ensemble) yöntemiyle bir araya getirilmesiyle oluşturulan güçlü bir makine öğrenmesi algoritmasıdır. Bu yöntem hem sınıflandırma hem de regresyon problemlerinde kullanılabilir ve yüksek doğruluk oranları ile aşırı öğrenmeye karşı dayanıklılığıyla öne çıkar.

Model, eğitim verisinden rastgele örnekleme (bootstrap) yöntemiyle farklı alt kümeler oluşturur ve bu alt kümelerle çok sayıda karar ağacı eğitir. Her bir ağacın çıktısı, sınıflandırma problemlerinde çoğunluk oyu, regresyon problemlerinde ise ortalama alınarak nihai tahmin üretilir. Bu yapı sayesinde model, tek bir karar ağacına kıyasla daha düşük varyans ve daha yüksek genelleme performansı sergiler.

Ayrıca, her bir karar ağacı, özniteliklerin rastgele bir alt kümesi üzerinden bölünme yaparak çeşitliliği artırır. Bu rastgelelik hem modelin aşırı öğrenmesini engeller hem de bağımsız ağaçların katkısını optimize eder. Rastgele Ormanlar, değişken önem derecesi (feature importance) hesaplamalarında da yaygın olarak kullanılır; bu sayede hangi özniteliklerin model kararlarına ne kadar katkı sağladığı değerlendirilebilir (Breiman, 2001).

3.2.6.4 XGBoost Regresyon

XGBoost (eXtreme Gradient Boosting), gradyan artırma yöntemine dayalı, karar ağaçları temelli genişletilebilir ve yüksek performanslı bir öğrenme algoritmasıdır. Bu model, zayıf öğreniciler (genellikle karar ağaçları) olan sınıflandırma ve regresyon ağaçlarını (CART) ardışık olarak eğiterek güçlü ve genelleme yeteneği yüksek bir tahmin modeli inşa eder. Özellikle büyük veri kümelerinde yüksek doğruluk sağlayan XGBoost, paralel hesaplama desteği, düzenlileştirme (regularization) bileşenleri ile aşırı öğrenmeye karşı direnç, eksik veri ile doğrudan çalışabilme ve değişken önem düzeylerini hesaplayabilme gibi avantajlara sahiptir. Ayrıca modelin parametrik esnekliği, farklı veri yapıları üzerinde etkili bir şekilde uygulanmasını mümkün kılmaktadır (Chen & Guestrin, 2016).

3.2.6.5 Lineer Regresyon

Lineer regresyon, bağımlı bir değişken ile bir veya birden fazla bağımsız değişken arasındaki doğrusal ilişkiyi modelleyen temel bir istatistiksel yöntemdir. Model, bağımlı değişkenin, bağımsız değişkenlerin doğrusal kombinasyonu olarak ifade edilmesini esas

alır. Katsayılar, genellikle en küçük kareler yöntemi kullanılarak tahmin edilir. Bu yaklaşım, tahmin edilen değerler ile gözlenen değerler arasındaki hata karelerinin toplamını minimize etmeyi amaçlar. Parametrik bir model olması nedeniyle, model varsayımları doğrultusunda güvenilir ve yorumlanabilir sonuçlar üretir. Özellikle modelin anlamlılığı, bağımsız değişkenlerin katkısı ve hata terimlerinin özellikleri, regresyon analizi kapsamında değerlendirilir (Montgomery, Peck & Vining, 2012).

3.2.6.6 Yapay Sinir Ağları

Yapay Sinir Ağları (ANN), biyolojik sinir sistemlerinden esinlenerek geliştirilmiş, veri içerisindeki karmaşık ve doğrusal olmayan ilişkileri öğrenebilen bir makine öğrenmesi modelidir. Temel yapı taşları, birbirine bağlanan yapay nöronlardan oluşur. Bu nöronlar, girişten alınan bilgileri ağırlıklarla çarpar, aktivasyon fonksiyonu yardımıyla işler ve sonucu bir sonraki katmana aktarır. Bir sinir ağı genellikle üç katmandan oluşur: giriş katmanı, bir veya daha fazla gizli katman ve çıktı katmanı.

ANN'ler genellikle ileri yayılım (forward propagation) yapıda tasarlanır ve modelin öğrenme süreci geri yayılım (backward propagation) algoritması ile gerçekleşir. Bu algoritma, modelin çıktısı ile gerçek değer arasındaki hatayı minimize edecek şekilde ağırlıkları güncelleyerek öğrenmeyi sağlar. Yapay sinir ağları, esnek yapıları sayesinde sınıflandırma, regresyon ve örüntü tanıma gibi pek çok görevde yaygın olarak kullanılmaktadır (Aggarwal, 2018).

4. ANALİZ VE BULGULAR

Bölüm 3' te yer alan materyal ve yöntemler kullanılarak web kazıma ile elde edilen veri seti, makine öğrenmesi modelleri ile analiz edilmiş ve çeşitli değerlendirme metrikleri aracılığıyla performansları karşılaştırılmıştır. Bu analiz sürecinde, modellerin en iyi performansı göstermesi amacıyla hiperparametre ayarlamaları gerçekleştirilmiştir. Bu amaçla, parametre kombinasyonlarını sistematik biçimde tarayan GridSearchCV yöntemi ve rastgele parametre örneklemeleri üzerinden arama yapan RandomizedSearchCV yöntemi kullanılmıştır.

4.1 K-En Yakın Komşu

GridSearchCV ve RandomizedSearchCV ile en iyi parametreler belirlendi ve veri seti analizi gerçekleştirilmiştir. Sonuçlar aşağıdaki gibidir.

GridSearchCV için en iyi parametreler: {'n_neighbors': 4, 'p': 1, 'weights': 'distance'}

RandomizedSearchCV için en iyi parametreler: {'weights': 'distance', 'p': 1, 'n_neighbors': 5}

Tablo 1 KNN model sonuçları

	Train R^2	Test R^2	Test MAE	Test MSE	Test RMSE
GridSearchCV	0.929	0.926	97690.036	42953766951.479	207252.906
RandomizedSearchCV	0.928	0.926	97690.036	42953766951.479	207252.906

Tablo 1' deki sonuçlara göre, GridSearchCV ve RandomizedSearchCV ile KNN modeli için en uygun hiperparametreler belirlendi ve her iki yöntemle de benzer sonuçlar elde edilmiştir. Her iki modelde yüksek doğruluk test verisinde 0,926 R^2 sağlanmış, test MAE yaklaşık 97.690 TL, test RMSE ise 207.252 TL olarak hesaplanmıştır. Bu sonuçlara göre her iki yöntem de benzer performans gösterse de, GridSearchCV ile elde edilen modelin train R^2 değerinin daha yüksek 0,929 olması nedeniyle GridSearchCV parametreleri tercih edilmiştir. Elde edilen bu model, fiyat tahmin arayüz uygulamasında kullanılmak üzere kaydedilmiştir.

4.2 Karar Ağaçları

GridSearchCV ve RandomizedSearchCV ile en iyi parametreler belirlendi ve veri seti analizi gerçekleştirildi. Sonuçlar aşağıdaki gibidir.

GridSearchCV için en iyi parametreler: {'max_depth': None, 'max_features': 'log2', 'min_samples_leaf': 2, 'min_samples_split': 5}

RandomizedSearchCV için en iyi parametreler: {'min_samples_split': 5, 'min_samples_leaf': 2, 'max_features': 'sqrt', 'max_depth': 15}

Tablo 2 DT model sonuçları

	Train R^2	Test R^2	Test MAE	Test MSE	Test RMSE
GridSearchCV	0,901	0,890	112299,186	63954156923,827	252891,591
RandomizedSearchCV	0,883	0.903	104773,544	56599486429,639	237906.466

Tablo 2' deki sonuçlara göre, her ne kadar GridSearchCV ile elde edilen model eğitim verisinde daha yüksek performans train verisinde 0,901 R^2 göstermiş olsa da RandomizedSearchCV yöntemiyle oluşturulan model, test verisinde daha yüksek doğruluk test verisinde 0,903 R^2 ve daha düşük hata oranları MAE 104.773 TL, RMSE 237.906 TL sunmuştur.

Bu nedenle, gerçek veri üzerindeki tahmin başarısı göz önünde bulundurularak, fiyat tahmin arayüz uygulamasında kullanılmak üzere RandomizedSearchCV ile elde edilen karar ağacı modeli kaydedilmiştir.

4.3 Rastgele Orman

GridSearchCV ve RandomizedSearchCV ile en iyi parametreler belirlendi ve veri seti analizi gerçekleştirildi. Sonuçlar aşağıdaki gibidir.

GridSearchCV için en iyi parametreler: {'max_depth': None, 'max_features': 'log2', 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 200}

RandomizedSearchCV için en iyi parametreler: {'n_estimators': 200, 'min_samples_split': 2, 'min_samples_leaf': 1, 'max_features': 'log2', 'max_depth': 20}

Tablo 3 RFR model sonuçları

	Train R^2	Test R^2	Test MAE	Test MSE	Test RMSE
GridSearchCV	0,952	0,953	70701,080	27.296.511.509,502	165216,559
RandomizedSearchCV	0,951	0,954	71.032,33	26.715.438.454,46	163448,58

Tablo 3' teki sonuçlara göre, GridSearchCV ve RandomizedSearchCV yöntemleriyle oluşturulan Random Forest modelleri benzer performans göstermiştir.

Her ne kadar GridSearchCV ile elde edilen model eğitim verisinde biraz daha yüksek performans train verisinde 0.952 R^2 sunsa da, RandomizedSearchCV yöntemiyle oluşturulan model, test verisinde daha yüksek doğruluk test verisinde 0,954 R^2 ve daha düşük hata değerleri RMSE 163.448 TL sağlamıştır.

Bu nedenle, gerçek veri üzerindeki başarısı dikkate alınarak, fiyat tahmin arayüz uygulamasında kullanılmak üzere RandomizedSearchCV ile oluşturulan Random Forest modeli kaydedilmiştir.

4.4 XGBoost Regresyon

GridSearchCV ve RandomizedSearchCV ile en iyi parametreler belirlendi ve veri seti analizi gerçekleştirildi. Sonuçlar aşağıdaki gibidir.

GridSearchCV için en iyi parametreler: {'learning_rate': 0.1, 'max_depth': 5, 'n_estimators': 500}

RandomizedSearchCV için en iyi parametreler: {'n_estimators': 500, 'max_depth': 7, 'learning_rate': 0.1}

Tablo 4 XGB model sonuçları

	Train R^2	Test R^2	Test MAE	Test MSE	Test RMSE
GridSearchCV	0,956	0,964	70.592,586	21.162.840.064	145.474,531
RandomizedSearchCV	0,956	0.963	66.878,508	21.722.667.008	147.386,109

Tablo 4' teki sonuçlara göre, modelin performansı GridSearchCV ve RandomizedSearchCV yöntemleriyle optimize edilerek değerlendirilmiştir. GridSearchCV yöntemiyle elde edilen model, eğitim verisi üzerinde 0,956 R^2, test verisi üzerinde ise 0,964 R^2 skoruna ulaşmıştır. Bu durum, modelin öğrenilen bilgiyi genelleme konusunda başarılı olduğunu ve aşırı öğrenme (overfitting) göstermediğini ortaya koymaktadır.

RandomizedSearchCV yöntemi ile benzer bir başarı elde edilmiş, eğitim verisi üzerinde 0,956 R^2 ve test skoru 0,963 R^2 olarak hesaplanmıştır.

Modelin hata oranları incelendiğinde, GridSearchCV için ortalama mutlak hata MAE 70.592 TL, kök ortalama kare hata RMSE ise 145.474 TL olarak hesaplanmıştır. RandomizedSearchCV ile ise MAE 66.878 TL, RMSE ise 147.386 TL olarak elde edilmiştir. Bu değerler, modelin gerçek otomobil fiyatlarına oldukça yakın tahminler üretebildiğini göstermektedir.

XGBoost modeli için hiperparametre ayarlamalarında, RandomizedSearchCV yöntemi daha kısa sürede benzer ve hatta bazı metriklerde daha iyi sonuçlar verdiği için fiyat tahmin arayüz uygulamasında kullanılmak üzere model kaydedilmiştir.

4.5 Lineer Regresyon

Lineer regresyon modeli veri seti üzerinde uygulanmış ve aşağıdaki sonuçlar elde edilmiştir:

- R^2 skoru: 0,678
- MAE: 236.671,064
- MSE: 187.236.918.782,541
- RMSE: 432.708,815

Bu sonuçlar, lineer modelin veri setindeki ilişkileri tam olarak yakalayamadığını, doğrusal olmayan yapılar karşısında yetersiz kaldığını göstermektedir. Bu nedenle, daha karmaşık yapıları öğrenebilen modellerin tercih edilmesi gerektiği sonucuna varılmıştır.

4.6 Yapay Sinir Ağları

ANN modelinde, sırasıyla 256, 128 ve 64 nöron içeren 3 gizli katmandan oluşan, her katmanda ReLU aktivasyon fonksiyonu uygulanmıştır. Aşırı öğrenmeyi (overfitting)

önlemek amacıyla her katmandan sonra Dropout katmanları eklenmiş, bu sayede azı bağlantıları eğitim sırasında rastgele devre dışı bırakılarak modelin genelleme yeteneği arttırılmıştır.

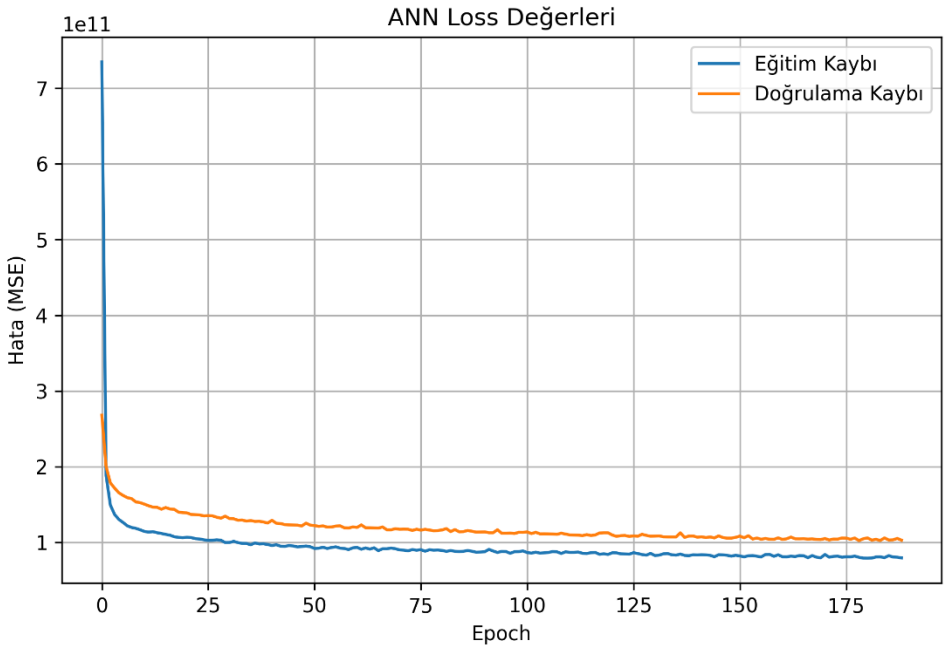
Modelin son katmanı tek nöronlu olup, fiyat tahmini yapmak üzere yapılandırılmıştır. Modelin derlenmesinde "mean squared error (MSE)" kayıp fonksiyonu ve Adam optimizasyon algoritması (learning_rate = 0.001) kullanılmıştır. Eğitim sürecinde ise EarlyStopping uygulanarak doğrulama kaybı 10 epoch boyunca iyileşmediğinde eğitim durdurulmuş ve en iyi ağırlıklar geri yüklenmiştir.

Modelin test verisi üzerindeki performansı şu şekildedir:

- R^2 skoru: 0,869
- MAE: 126.666,508
- MSE: 76.416.589.824
- RMSE: 276.435,5

Yapay Sinir Ağı modeli, test verisinde 0,869 R^2 ile iyi bir genelleme başarısı göstermiştir. Ancak MAE 126.667 TL ve RMSE 276.436 TL gibi hata değerleri, diğer bazı modellerle kıyaslandığında nispeten daha yüksektir. Bu sonuçlar, modelin genel eğilimleri yakalayabildiğini ancak bazı bireysel tahminlerde sapmaların olabileceğini göstermektedir.

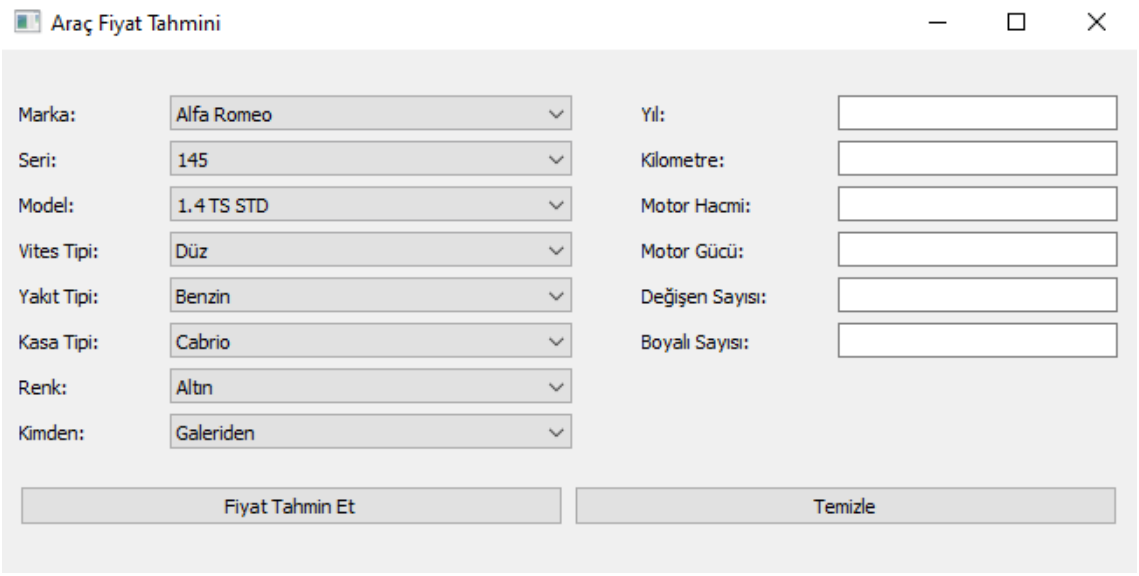
Şekil 4.1’ de ANN modelin loos grafiği incelendiğinde, eğitim ve doğrulama hatalarının başlangıçta yüksek olduğu ancak kısa sürede düşüşe geçtiği görülmektedir. Belirli bir noktadan sonra her iki kayıp değeri de dengelenmiş ve birbirine yakın seyretmiştir. Bu durum, modelin istikrarlı bir şekilde öğrenme gerçekleştirdiğini ve aşırı öğrenmeden kaçındığını göstermektedir. Eğitim ve doğrulama hataları arasındaki uyum, modelin genel performansının güvenilir olduğunu desteklemektedir.



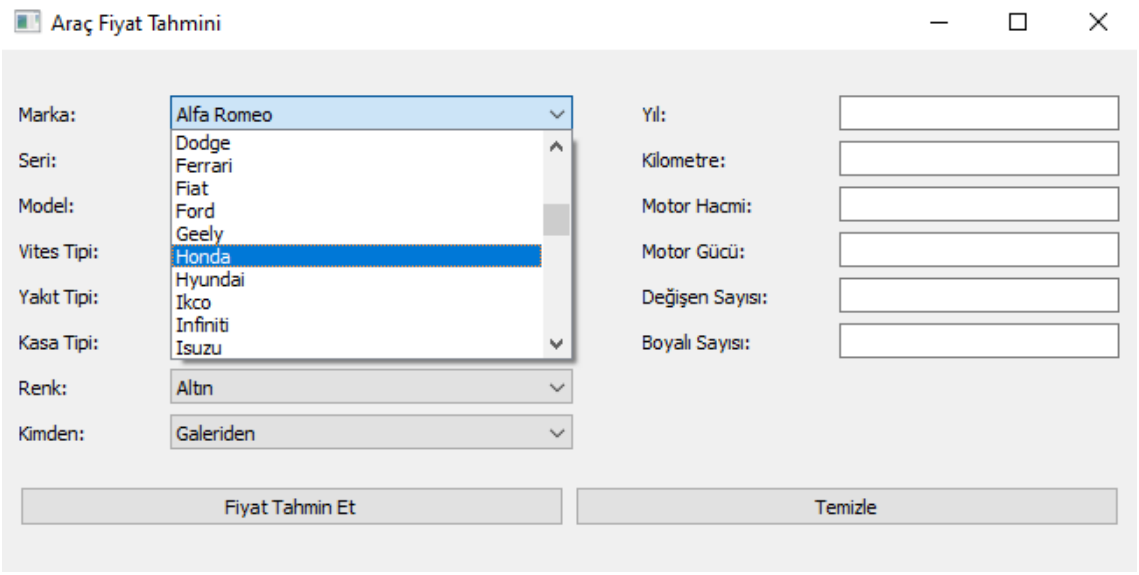
Şekil 4. 1 ANN Loss eğrisi

4.7 Fiyat Tahmin Arayüz

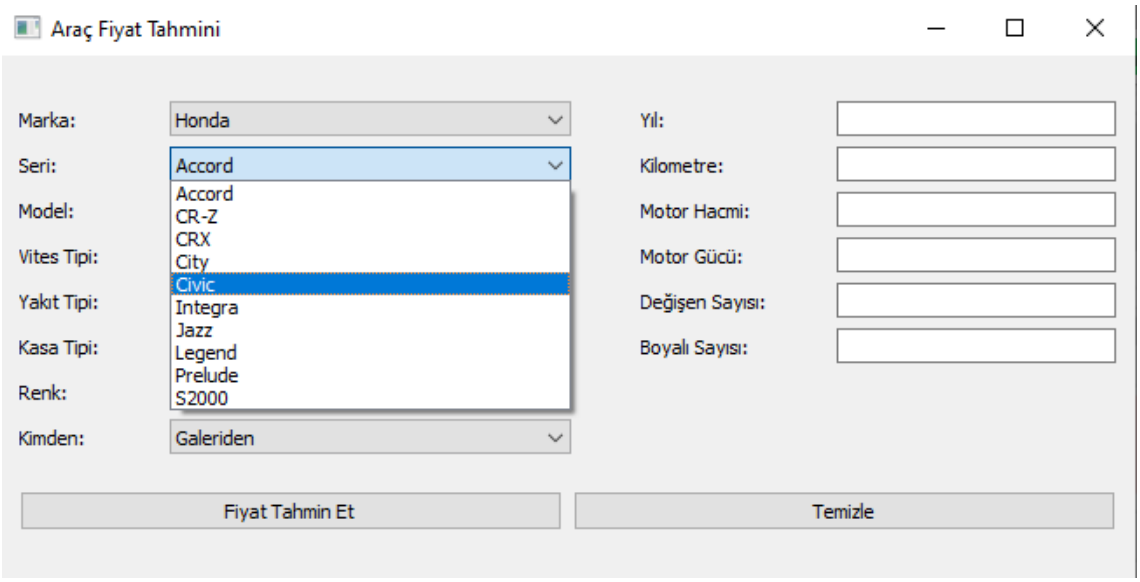
Makine öğrenmesi analizi sonucunda, en iyi sonucu veren XGBoost modeli temel alınarak bir fiyat tahmin arayüz uygulaması tasarlanmıştır. Bu arayüz üzerinden kullanıcıdan ilgili giriş verileri alınmakta ve bu veriler veri seti ile eğitilen XGBoost modeline aktarılmaktadır. Model, girilen özellikler doğrultusunda tahmin işlemini gerçekleştirmekte ve sonuç kullanıcıya sunulmaktadır. Şekil 4.2' de gösterildiği gibi kullanıcıdan otomobil özellikleri alınmıştır. Marka, seri, model gibi alanlar için Şekil 4.3' teki gibi açılır pencereden kullanıcın seçim yapabilmesi sağlandı. Seçilen markaya ait serilerin bir sonraki pencere otomatik bir şekilde filtrelenmesi özelliği eklendi seri seçildikten sonra da model için de filtreleme özelliği eklenmiştir. Şekil 4.3, 4.4, 4.5 ve 4.6' da kullanıcıdan alınan bilgiler girildi ve tahminleme örneği gösterildi. Marka Honda, seri Civic, model 1.6 i-VTEC Elegance, vites tipi otomatik, yakı tipi benzin, kasa tipi Sedan, renk beyaz, kimden galeriden, yıl 2013, kilometre 188000, motor hacmi 1500, motor gücü 113, değişen sayısı 2 ve boyalı sayısı 4 olarak girildi. Tahmin değeri Şekil 4.8' de görüldüğü gibi 888.076 TL otomobilin gerçek değeri Şekil 4.7' de görüldüğü gibi 890.000 TL, tahmin edilen ile gerçek değer arsındaki fark 1924 TL' dir.



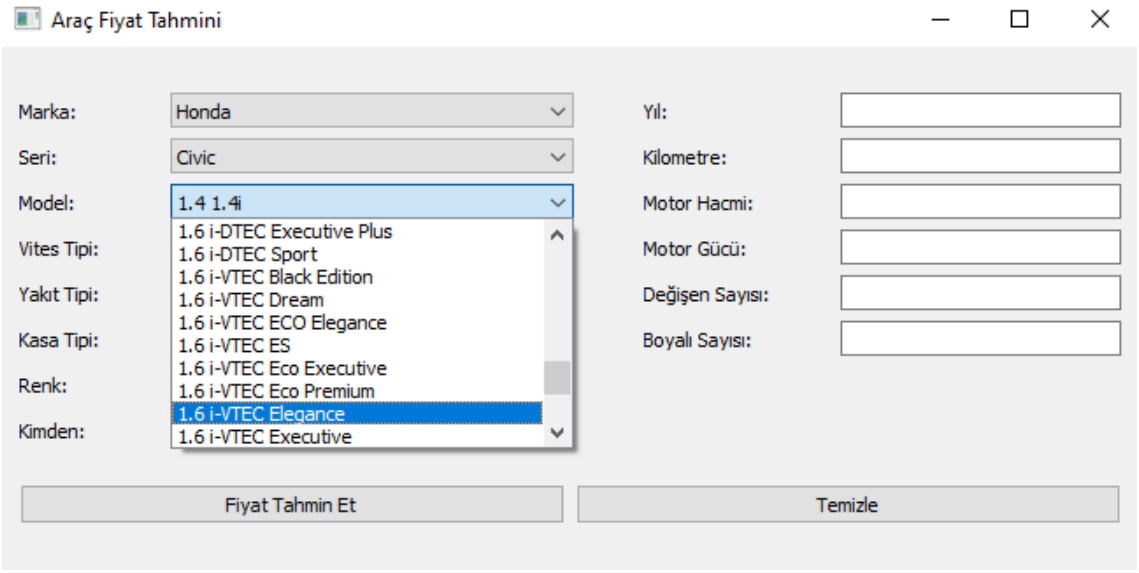
Şekil 4. 2 Uygulama tahmin arayüzü



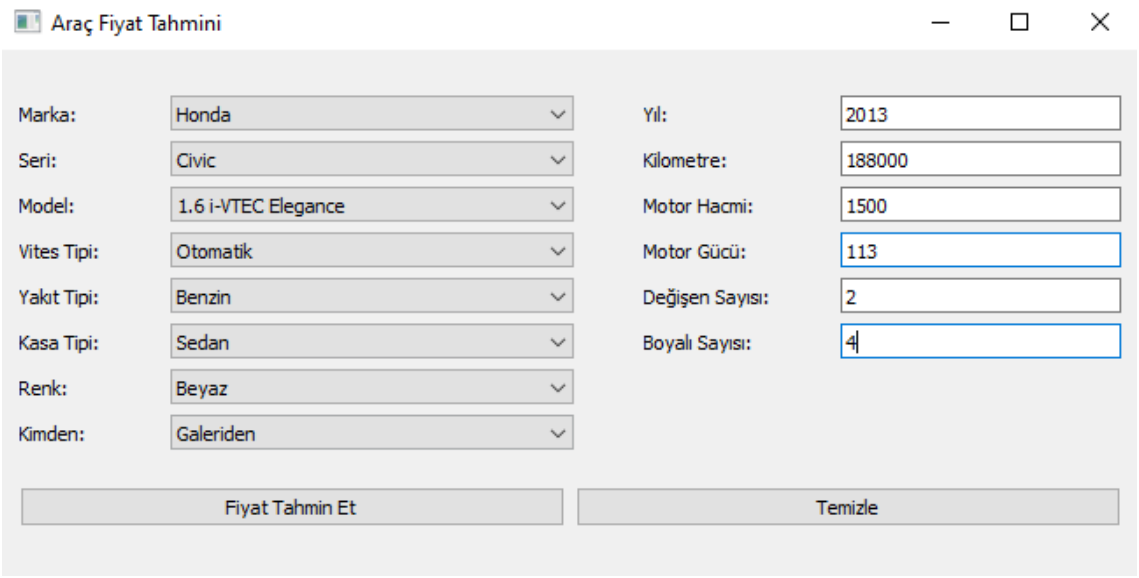
Şekil 4. 3 Marka filtreleme özelliği



Şekil 4. 4 Seri filtreleme özelliği



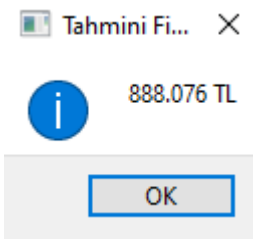
Şekil 4. 5 Model filtreleme özelliği



Şekil 4. 6 Otomobil özellikleri

	marka	seri	model	yil	kilometre	vites_tipi	yakit_tipi	kasa_tipi	renk	motor_hacmi	motor_gucu	degisen_sayisi	boyali_sayisi	kimden	fiyat
32583	Honda	Civic	1.6 i-VTEC Elegance	2013	188000	Otomatik	Benzin	Sedan	Beyaz	1500	113	2	4	Galeriden	890000

Şekil 4. 7 Tahmin edilmesi gereken otomobil özellikleri



Şekil 4. 8 Tahmin sonucu

5. TARTIŞMA VE SONUÇLAR

Bu çalışmada makine öğrenmesi modelleri ile yapılan analizler sonucunda çıktılar elde edilmiştir. Tabloda test verisinden elde edilen R^2, MAE ve RMSE değerleri bulunmaktadır.

Tablo 5 Tüm model sonuçları

	KNN	DT	RFR	XGB	LR	ANN
R^2	0,929	0,903	0,954	0,963	0,678	0,869
MAE	97.690,04	104.773,54	71.032,33	66.878,508	236.671,064	126.666,51
RMSE	207.252,906	237.906,47	163.448,58	147.386,12	432.708,82	276.435,5

Yapılan karşılaştırmalar sonucunda, XGB model, test verisi üzerinde en yüksek R^2 skorunu 0,963 R^2 ve en düşük hata metriklerini MAE 66.878, RMSE 147.386 vererek en başarılı model olmuştur. Bu modelin hem eğitim hem de test verisi üzerinde benzer başarı göstermesi, aşırı öğrenme (overfitting) sorununu minimuma indirdiğini göstermektedir.

Hiperparametre optimizasyonu sürecinde GridSearchCV ve RandomizedSearchCV yöntemleri denenmiş; RandomizedSearchCV, daha kısa sürede benzer hatta daha başarılı sonuçlar ürettiği için tercih edilmiştir.

Sonuç olarak, bu çalışmada geliştirilen model, ikinci el otomobil piyasasında kullanıcıların daha gerçekçi fiyat tahminleri yapabilmesine olanak sağlamaktadır. Ayrıca model, sektörel analizler, bayi değerlendirmeleri ve fiyat politikaları oluşturma süreçlerinde de kullanılabilecek niteliktedir.

KAYNAKLAR

Aggarwal, C. C. (2018). *Neural networks and deep learning* (Vol. 10, No. 978, p. 3). Cham: Springer.

Altman, N. S. (1992). An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician*, *46*(3), 175-185.

Asilkan, Ö. (2008). Veri madenciliği kullanılarak ikinci el otomobil pazarında fiyat tahmini.

Breiman, L. (2001). Random forests. *Machine learning*, *45*, 5-32.

Burges, C. J. (1998). A tutorial on support vector machines for pattern recognition. *Data mining and knowledge discovery*, *2*(2), 121-167.

Chen, T., & Guestrin, C. (2016, August). Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining* (pp. 785-794).

Çelik, Ö., & Osmanoğlu, U. Ö. (2019). Prediction of the prices of second-hand cars. *Avrupa Bilim ve Teknoloji Dergisi*, (16), 77-83.

Dere, E. (2023). Makine öğrenmesi yöntemleri ile ikinci el otomobillarda fiyat tahmini.

Gülmez, B., & Kulluk, S. (2023). Türkiye'de ikinci el otomobilların büyük veri ve makine öğrenme teknikleriyle analizi ve fiyat tahmini. *Gazi Üniversitesi Mühendislik Mimarlık Fakültesi Dergisi*, *38*(4), 2279-2290.

Gültekin, S. U. (2017). *Veri madenciliği: yapay sinir ağı ve doğrusal regresyon yöntemleri ile fiyat tahmini* (Master's thesis, Pamukkale Üniversitesi Sosyal Bilimler Enstitüsü).

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). *An introduction to statistical learning* (Vol. 112, No. 1). New York: springer.

Muti, S., & Yıldız, K. (2023). Using linear regression for used car price prediction. *International Journal of Computational and Experimental Science and Engineering*, *9*(1), 11-16.

Myers, R. H., Montgomery, D. C., Vining, G. G., & Robinson, T. J. (2012). *Generalized linear models: with applications in engineering and the sciences*. John Wiley & Sons.

Namlı, E., Ünlü, R., & Gül, E. (2019). Fiyat tahminlemesinde makine öğrenmesi teknikleri ve doğrusal regresyon yöntemlerinin kıyaslanması; türkiye'de satılan ikinci el otomobil fiyatlarının tahminlenmesine yönelik bir vaka çalışması. *Konya Journal of Engineering Sciences*, *7*(4), 806-821.

Özçalıcı, M. (2017). Predicting second-hand car sales price using decision trees and genetic algorithms. *Alphanumeric Journal*, *5*(1), 103-114.

Yılmaz, S. (2023). *Web kazıma ve makine öğrenmesi yöntemleri kullanılarak fiyat tahminleme: İkinci el otomobil piyasasında bir örnek= Price prediction using web scraping and machine learning methods: An example in the used car market* (Master's thesis, Sakarya Üniversitesi).

GridSearchCV (https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html alındığı tarih: 30.06.2025.

RandomizedSearchCV (https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.RandomizedSearchCV.html) alındığı tarih: 30.06.2025.