LONDON HOUSE PRICE DATA

Ömer Alp Yentür
Bilişim Sistemleri Mühendisliği
Kocaeli Üniversitesi
Kocaeli,Türkiye
yenturalp@gmail.com

Github Proje

Yusuf Demir
Bilişim Sistemleri Mühendisliği
Kocaeli Üniversitesi
Kocaeli,Türkiye
dmrysf0320@gmail.com

***Özet*-- Bu projede, Londra ev fiyatları veri seti üzerinde gerçek zamanlı büyük veri analitiği kullanılarak anomali tespiti gerçekleştirilmiştir. Kafka ve Spark teknolojileri ile oluşturulan sistem, veri işleme ve anomali tespiti için makine öğrenmesi ve derin öğrenme algoritmalarını kullanmaktadır. Veri seti üzerinde yapılan ön işleme adımları, eksik verilerin giderilmesi, kategorik değişkenlerin sayısal hale getirilmesi ve outlier tespiti gibi adımları içermektedir. Ayrıca, kullanılan modelin başarım metrikleri (accuracy, precision, recall, F1 skoru) hesaplanmış ve raporlanmıştır. Kafka ve Spark entegrasyonu, gerçek zamanlı veri akışını yönlendirerek anomali tespiti sonuçlarının izlenmesine olanak tanımaktadır.**

Abstract--In this project, anomaly detection is performed on the London housing prices dataset using real-time big data analytics. The system, built with Kafka and Spark technologies, utilizes machine learning and deep learning algorithms for data processing and anomaly detection. The preprocessing steps applied to the dataset include handling missing values, encoding categorical variables, and detecting outliers. Additionally, the performance metrics of the model, such as accuracy, precision, recall, and F1 score, are calculated and reported. The integration of Kafka and Spark enables real-time data flow management, allowing the monitoring of anomaly detection results effectively.

Anahtar Kelimeler— Spark, Kafka, Anomali, Pytorch, Veri Seti, Londra

I. GİRİŞ

Günümüz veri analitiği dünyasında, büyük veri setlerinden anlamlı sonuçlar elde etmek için gerçek zamanlı analiz ve anomali tespiti giderek daha önemli hale gelmektedir. Bu proje, Londra ev fiyatları veri setini kullanarak, gerçek zamanlı büyük veri analitiği ve anomali tespiti gerçekleştirmeyi amaçlamaktadır. Veri seti, evlerin fiyatları, konumları, oda sayıları gibi faktörleri içermektedir ve bu veri seti üzerinde anomali tespiti yapmak için Spark ve Kafka teknolojileri kullanılacaktır. Gerçek zamanlı veri işleme sürecinde, Kafka'nın veri iletimi ve Spark'ın veri işleme yetenekleri, anomali tespiti için kritik rol oynamaktadır. Bu raporda, proje kapsamında kullanılan sistem mimarisi, teknolojiler ve elde edilen sonuçlar detaylı bir şekilde ele alınacaktır.

II. SİSTEM MİMARİSİ

A. *Kafka Producer*

Kafka Producer, veri akışını başlatan ve sürekli olarak yeni veriler üreten bir bileşendir. Bu bileşen, veri kaynaklarından alınan verileri toplayarak belirli bir formatta Kafka'nın belirli topic'lerine yönlendirir. Verilerin gerçek zamanlı olarak işlenebilmesi için bu üretim süreci sürekli hale getirilir. Özellikle ev fiyatları gibi dinamik veri kümelerinde, her yeni fiyat değişikliği veya özellik eklenmesi, Kafka Producer aracılığıyla anında sisteme iletilir. Kafka, yüksek hızda veri iletimi ve düşük gecikme süresi ile büyük veri akışlarını yönetebilme kapasitesine sahiptir, bu sayede sistemin verimli ve ölçeklenebilir bir şekilde çalışmasını sağlar.

B. *Kafka Consumer*

Kullanıcının Kafka Consumer, anomali tespiti sonucu oluşan veriyi Kafka topic'lerinden alır ve bu veriler üzerinde işlem yaparak belirli aksiyonları tetikler. Anomali tespit edildikçe, Kafka Consumer, bu veriyi ilgili topic'ten çekerek, anomaliyi kullanıcıya bildirebilir veya başka bir işlem yapılmasını sağlayacak şekilde veriyi işler. Kafka Consumer, sürekli veri akışını işlemek üzere yapılandırılabilir ve tespit edilen anomaliyi veri kümesine entegre eder, böylece yeni anomali tespitleri sistem tarafından hızla işlenebilir. Kafka'nın esnek yapısı sayesinde, veri işleme süreçleri kesintisiz ve yüksek verimlilikle devam edebilir.

C. *Spark Streaming*

Spark Streaming, Apache Spark'ın zaman içinde değişen veriyi işleme yeteneğini sağlayan bir bileşendir. Kafka'dan gelen veriler, Spark Streaming ile alınır, işlenir ve analiz edilir. Bu aşama, anomali tespiti için kritik bir rol oynar. Spark Streaming, verileri yüksek hızda işleyerek gerçek zamanlı tahminler yapabilir ve anomali tespiti algoritmalarını çalıştırarak normal veriler ile anormal verileri birbirinden ayırabilir. Verinin sürekli bir akış halinde geldiği durumlarda, Spark Streaming, veriyi küçük parçalara ayırarak işler ve her bir parça üzerinde anlık tahminler yapar. Bu, sistemin gerçek zamanlı olarak sürekli olarak güncel kalmasını ve anomali tespitlerini anında yapabilmesini sağlar. Spark, paralel işleme gücüyle büyük veri kümeleri üzerinde işlem yapabilen güçlü bir platformdur, bu sayede verinin işlenmesi hızlı ve etkili bir şekilde gerçekleşir.

D. *Makine Öğrenmesi Modelleri*

1) Long Short-Term Memory (LSTM)

Bu projede, Long Short-Term Memory (LSTM) ve Autoencoder gibi derin öğrenme teknikleri kullanılarak

anomali tespiti gerçekleştirilecektir. LSTM (Long Short-Term Memory), zaman serisi verileri üzerinde çalışırken mükemmel sonuçlar veren bir derin öğrenme modelidir. Ev fiyatları gibi zamanla değişen verilerde, LSTM modelinin güçlü hafıza mekanizması, geçmiş veri noktalarındaki desenleri öğrenerek gelecekteki olası anomali noktalarını tespit edebilir. Özellikle fiyatların zaman içindeki dalgalanmasını dikkate alarak, alışılmadık fiyat hareketlerini hızlıca algılayabilir.

2) ***Autoencoder,***

Autoencoder, anomali tespitinde kullanılan başka bir derin öğrenme algoritmasıdır. Bu model, verinin normal desenini öğrenir ve anormal veri noktalarını bu desenin dışında kalanlar olarak tespit eder. Autoencoder, veriyi sıkıştırarak düşük boyutlu bir temsilini öğrenir ve ardından bu temsilden orijinal veriyi yeniden oluşturur. Bu yeniden yapılandırma hatası, normal veriler için düşük olurken, anormal veriler için yüksek olur. Bu şekilde, Autoencoder modeli anomaliyi belirler. Bu derin öğrenme algoritmalarının kullanılması, projeye daha gelişmiş ve hassas anomali tespiti yapabilme yeteneği kazandırır. Hem LSTM hem de Autoencoder, projede karmaşık analizlerin yapılmasını sağlayacak ve anomali tespitlerinin doğruluğunu artıracaktır.

E. Veri Görselleştirme

Veri görselleştirme, veri analizi sürecinde elde edilen bilgilerin kullanıcılar tarafından daha iyi anlaşılmasını sağlar. Bu proje kapsamında, veri setinin analizi ve anomali tespit sonuçları, çeşitli görselleştirme teknikleri ile sunulacaktır. Veri görselleştirme süreci, kullanıcıların veriyi daha hızlı ve etkili bir şekilde analiz etmelerine yardımcı olur.

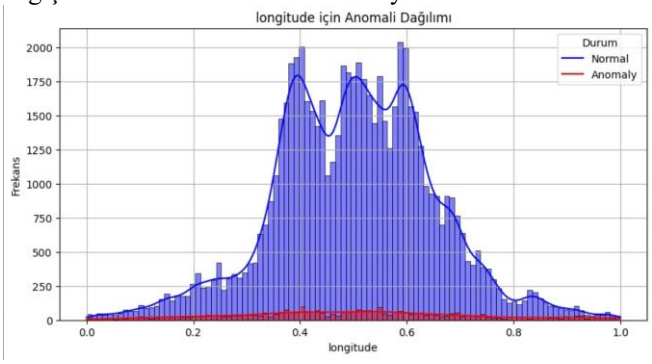
1) ***Veri Seti Dağılımı****:*

Ev fiyatlarının farklı özelliklere göre dağılımını görmek için histogramlar ve scatter plot'lar kullanılacaktır. Bu grafikler, verinin genel yapısını, dağılımını ve fiyatlarla ilgili olası ilişkileri göstermek için idealdir. Ayrıca, fiyatlar ile diğer özellikler arasındaki korelasyonları incelemek için scatter plot'lar ve korelasyon matrisleri kullanılacaktır.

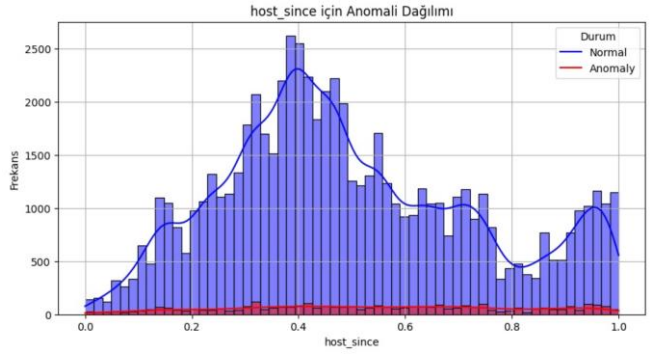
```
Veri Seti Genel Bilgileri:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 69865 entries, 0 to 69864
Data columns (total 17 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   name                  69865 non-null  float64
 1   host_name             69865 non-null  float64
 2   latitude              69865 non-null  float64
 3   longitude             69865 non-null  float64
 4   host_since            69865 non-null  float64
 5   host_location         69865 non-null  float64
 6   host_acceptance_rate  69865 non-null  float64
 7   property_type         69865 non-null  float64
 8   room_type             69865 non-null  float64
 9   price                 69865 non-null  float64
 10  minimum_nights        69865 non-null  float64
 11  maximum_nights        69865 non-null  float64
 12  availability_30       69865 non-null  float64
 13  availability_365      69865 non-null  float64
 14  number_of_reviews     69865 non-null  float64
 15  review_scores_rating  69865 non-null  float64
 16  anomaly               69865 non-null  int64
dtypes: float64(16), int64(1)
memory usage: 9.1 MB
None
```

2) ***Anomali Tespit Sonuçları***

Anomali tespiti sonrasında, tespit edilen anomaliler farklı bir görselleştirme ile sunulacaktır. Örneğin, fiyatların normalden sapmalarını gösteren zaman serisi grafiklerinin yanı sıra, anomali olan veri noktaları farklı renklerle işaretlenecektir. Bu, kullanıcıların anormal fiyat değişikliklerini hızlıca tanımalarına yardımcı olur.



Longitude için Anomali Dağılımı



Host Since için Anomali Dağılımı

3) ***Isı Haritaları (Heatmap)****:*

Özellikle korelasyon analizlerini daha iyi anlayabilmek için ısı haritaları kullanılacaktır. Isı haritaları, veri setindeki özellikler arasındaki ilişkilerin görsel olarak gösterilmesini sağlar, bu da kullanıcıların veri özellikleri arasındaki bağlantıları daha rahat görmelerine yardımcı olur.

III. VERİ ÖN İŞLEMESİ

Veri ön işleme, ham verilerin modelleme için kullanılabilir hale getirilmesi sürecidir. Bu adımda, verilerin doğruluğu, tutarlılığı ve analize uygunluğu sağlanarak, modelin performansı optimize edilir.

A. **Eksik Verilerin Doldurulması veya Çıkarılması**

Veri setlerinde sıklıkla eksik veriler bulunur. Bu eksik veriler, modelin doğru çalışmasını engelleyebilir. Bu projede, eksik veriler için iki farklı yaklaşım kullanılmıştır:

- Sayısal (numerik) veriler, ortalama ile doldurulmuştur. Bu, verilerin kaybolan noktalarının genel eğilimle uyumlu hale gelmesini sağlar.
- Kategorik veriler ise, en sık görülen değerle doldurulmuştur. Bu sayede, kategorik veri setindeki eksiklikler, veri dağılımına uygun şekilde tamamlanmıştır.

Eksik verilerin düzgün bir şekilde ele alınması,

modelin doğruluğunu ve güvenilirliğini artırmaktadır.

B. **Kategorik Değişkenlerin Sayısal Hale Getirilmesi (Encoding)**
Makine öğrenmesi algoritmalarının çoğu yalnızca sayısal verilerle çalışabilir, bu yüzden kategorik verilerin sayısal verilere dönüştürülmesi gereklidir. Bu projede, Label Encoding yöntemi kullanılmıştır. Label Encoding, sıralı olmayan kategorik verileri sayısal verilere dönüştürerek, her bir kategoriyi benzersiz bir tamsayı ile temsil eder. Bu işlem, modelin bu tür veriler üzerinde işlem yapabilmesini sağlar.

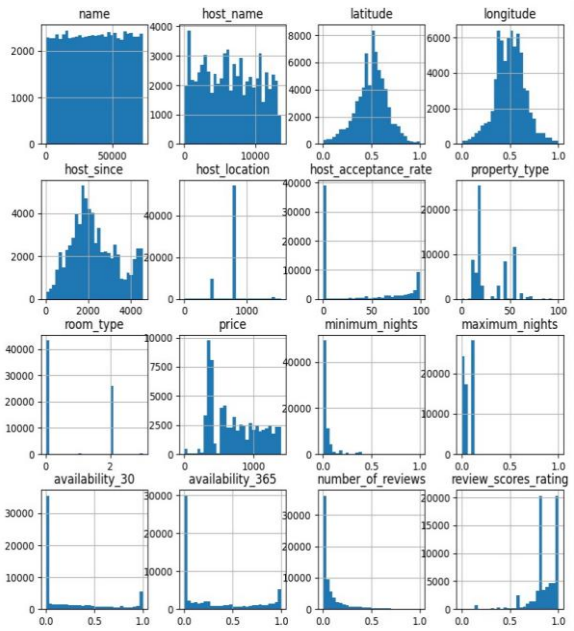
C. **Outlier (Aykırı Değer) Tespiti ve İşlenmesi**
Veri setinde yer alan aşırı uç (outlier) değerler, modelin tahmin kapasitesini olumsuz yönde etkileyebilir. Bu sebeple, z-skoru (Z-Score) yöntemi ile aykırı değerler tespit edilmiştir. Z-skoru, verilerin ortalamaya ne kadar uzak olduğunu gösterir. Bu projede, Z-skoru 3'ten büyük olan değerler veri setinden çıkarılmıştır. Bu işlem, modelin daha tutarlı ve güvenilir sonuçlar üretmesine olanak tanımaktadır.

D. **Verinin Normalize Edilmesi veya Standartlaştırılması**
Farklı ölçeklerdeki veriler, modelin verimli çalışmasını engelleyebilir. Bu nedenle, sayısal veriler üzerinde normalizasyon veya standartlaştırma işlemi yapılmıştır:

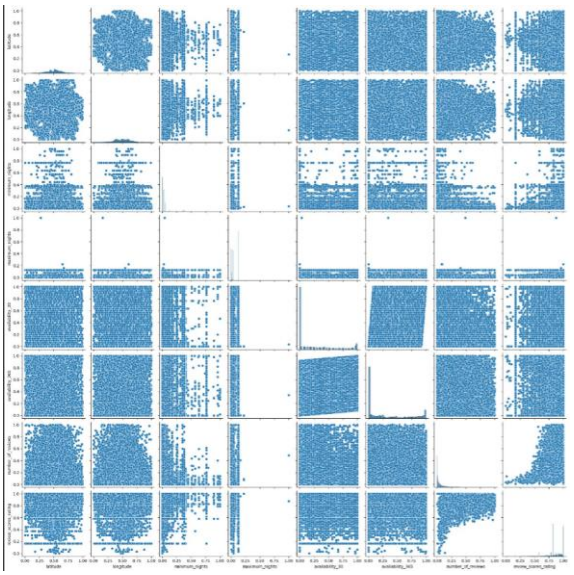
- ***Normalizasyon (Min-Max Scaling)**: Veriler, 0 ile 1 arasında bir skala ile normalize edilmiştir. Bu işlem, tüm sayısal değişkenlerin aynı ölçeğe getirilmesini sağlar ve modelin daha hızlı öğrenmesine yardımcı olur.*
- ***Standartlaştırma (Z-Score Standardization)**: Alternatif olarak, veriler Z-skoru kullanılarak standartlaştırılabilir. Bu işlem, verilerin ortalamasını 0, standart sapmasını ise 1 yapar, ancak bu adım projede kullanılmamıştır. Normalizasyon, özellikle derin öğrenme gibi algoritmalar için önemlidir, çünkü yüksek farklılıklar gösteren özellikler modelin eğitim sürecini zorlaştırabilir.*

E. **Veri Görselleştirmeleri**
Veri ön işleme süreci, görselleştirmelerle desteklenmiştir. Bu görselleştirmeler, verinin dağılımı, korelasyonlar ve genel yapısı hakkında bilgi verir:

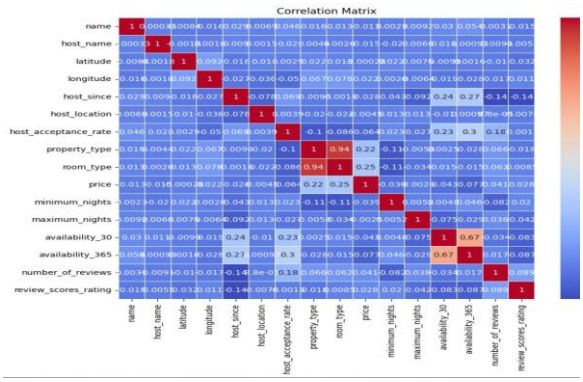
- **Histogramlar**: Verinin dağılımını görmek için kullanılmıştır. Her bir sayısal değişkenin histogramı, verinin nasıl dağıldığını anlamamıza yardımcı olur.
- **Pairplot (Scatter Plot)**: Sayısal değişkenler arasındaki ilişkileri incelemek için kullanılmıştır. Bu, değişkenler arasındaki olası korelasyonları görsel olarak gözlemlememize olanak tanır.
- **Korelasyon Matrisi ve Isı Haritası**: Veriler arasındaki doğrusal ilişkiler, korelasyon matrisini görselleştirerek tespit edilmiştir. Bu, hangi özelliklerin birbirine yakın ilişkilerde olduğunu ve hangi değişkenlerin birbiriyle bağımsız olduğunu anlamamıza yardımcı olur.



Histogram Grafikleri



Dağılım Grafikleri



Korelasyon Matrisi

IV. KULLANILAN TEKNOLOJİLER

A. *Apache Kafka*

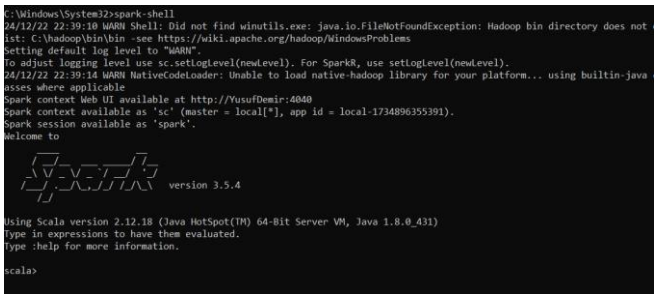
Apache Kafka, yüksek verimli, dağıtık bir mesajlaşma sistemidir ve bu projede gerçek zamanlı veri akışının yönetilmesi için kullanılmıştır. Kafka, veri üreticisi (Producer) ve veri tüketicisi (Consumer) arasında veri iletimi sağlar. Veri, Kafka'nın topic'lerine gönderilir ve bu veriler anlık olarak Spark Streaming ile işlenir. Kafka, verilerin kaybolmadan güvenli bir şekilde iletilmesini sağlarken, paralel veri akışları ve ölçeklenebilirlik sunar. Projede, Kafka'nın sağladığı düşük gecikme süreleri ve yüksek veri aktarım hızı, anomali tespiti gibi kritik işlemlerin hızlı bir şekilde yapılabilmesini sağlar.

B. *Apache Spark*

Apache Spark, büyük veri işleme ve analiz için kullanılan, in-memory (bellek içi) işlem yapabilme kapasitesine sahip bir dağıtık veri işleme motorudur. Spark, veri akışlarını gerçek zamanlı olarak işlemek için Spark Streaming modülünü kullanır. Spark Streaming, Kafka'dan gelen verileri anlık olarak işleyip işlenmiş veriyi çıkartırken, verilerin paralel işlenmesini ve hızlı analiz yapılmasını mümkün kılar. Bu sayede, sistem gerçek zamanlı veri akışını işleyerek, anomali tespiti yapar ve sonuçları ilgili topic'lere gönderir. Spark'ın gelişmiş API'leri ve paralel işlem yapabilme yeteneği, büyük veri setlerinin hızlı bir şekilde işlenmesini sağlamaktadır.

C. ***Spark Streaming**:*

Kafka'dan gelen veriler, Spark Streaming kullanılarak gerçek zamanlı bir şekilde işlenir. Spark Streaming, mikro partisyonlar halinde verileri alır ve her bir veri parçasını hızlı ve etkin bir şekilde analiz eder. Bu işlem, veri akışının sürekliliğini sağlayarak, sistemin anlık olarak gelen veriler üzerinde işlem yapmasına olanak tanır. Spark Streaming'in sağladığı yüksek performans, büyük veri işleme süreçlerinde kritik bir rol oynar ve sistemin anomali tespiti gibi karmaşık görevleri hızlı bir şekilde gerçekleştirmesini mümkün kılar.



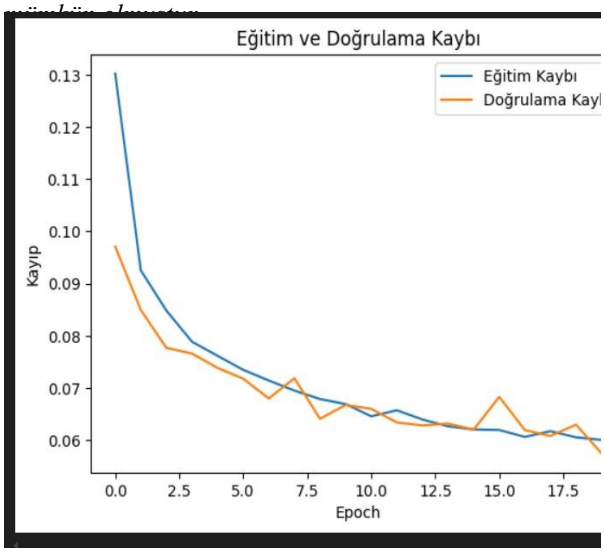
D. ***Apache ZooKeeper:***

Apache ZooKeeper, dağıtık sistemlerin yönetimi için kullanılan, merkezi bir hizmet sağlayan açık kaynaklı bir yazılımdır. Bu projede, ZooKeeper, Apache Kafka ve Spark arasında koordinasyonu sağlamak için kullanılmıştır. ZooKeeper, Kafka'nın her bir node'uyla iletişimi yönetir, topic ve partition bilgilerini izler ve bu yapıların tutarlılığını sağlar. Ayrıca, Spark'ın dağıtık işlem süreçlerinin yönetilmesine ve koordinasyonuna yardımcı olur. ZooKeeper, sistemin yüksek kullanılabilirliğini ve hata toleransını artırarak, tüm bileşenlerin senkronize bir şekilde çalışmasını temin eder.

Bu sayede, verilerin kaybolmadan ve hatasız bir şekilde işlenmesi sağlanır. ZooKeeper'in sağladığı güvenilir yönetim mekanizmaları, sistemin ölçeklenebilirliğini ve verimliliğini önemli ölçüde artırır.

E. *PyTorch*

PyTorch, özellikle derin öğrenme ve yapay zeka uygulamaları için yaygın olarak kullanılan açık kaynaklı bir kütüphanedir. PyTorch, dinamik hesap grafikleri ve hızlı prototip geliştirme özellikleriyle, araştırmacılar ve geliştiriciler tarafından tercih edilmektedir. Derin öğrenme modelleri geliştirilirken, özellikle sinir ağlarının eğitiminde kullanılır. Bu projede, PyTorch, modelin anomali tespiti için eğitilmesi ve tahminler yapılması amacıyla kullanılmıştır. PyTorch'un esnek yapısı ve GPU desteği, büyük veri setleriyle çalışırken eğitim sürecini hızlandırır ve yüksek verimlilik sağlar. Ayrıca, PyTorch'un güçlü API'leri sayesinde modelin hızla optimize edilmesi ve test edilmesi [illegible]




```
Epoch 1/20, Training Loss: 0.1302, Validation Loss: 0.0971
Epoch 2/20, Training Loss: 0.0925, Validation Loss: 0.0849
Epoch 3/20, Training Loss: 0.0848, Validation Loss: 0.0777
Epoch 4/20, Training Loss: 0.0789, Validation Loss: 0.0766
Epoch 5/20, Training Loss: 0.0762, Validation Loss: 0.0738
Epoch 6/20, Training Loss: 0.0735, Validation Loss: 0.0718
Epoch 7/20, Training Loss: 0.0714, Validation Loss: 0.0680
Epoch 8/20, Training Loss: 0.0695, Validation Loss: 0.0719
Epoch 9/20, Training Loss: 0.0679, Validation Loss: 0.0641
Epoch 10/20, Training Loss: 0.0669, Validation Loss: 0.0667
Epoch 11/20, Training Loss: 0.0646, Validation Loss: 0.0660
Epoch 12/20, Training Loss: 0.0657, Validation Loss: 0.0634
Epoch 13/20, Training Loss: 0.0640, Validation Loss: 0.0628
Epoch 14/20, Training Loss: 0.0626, Validation Loss: 0.0632
Epoch 15/20, Training Loss: 0.0620, Validation Loss: 0.0620
Epoch 16/20, Training Loss: 0.0619, Validation Loss: 0.0683
Epoch 17/20, Training Loss: 0.0606, Validation Loss: 0.0619
Epoch 18/20, Training Loss: 0.0617, Validation Loss: 0.0608
Epoch 19/20, Training Loss: 0.0605, Validation Loss: 0.0630
Epoch 20/20, Training Loss: 0.0600, Validation Loss: 0.0575
```

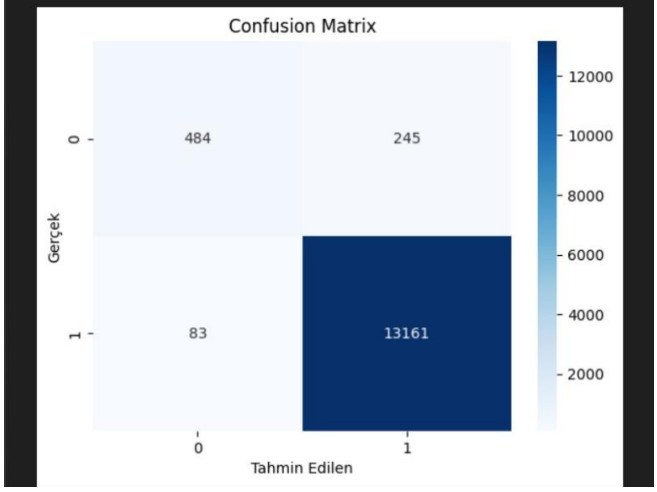
F. *Python ve Makine Öğrenmesi Kütüphaneleri*

Proje, veri seti üzerinde ön işleme, görselleştirme ve model geliştirme süreçlerinde ***Python*** *programlama dilini kullanmaktadır. Python, veri bilimi ve makine öğrenmesi alanında yaygın olarak kullanılan bir dildir ve çok sayıda güçlü kütüphane sunmaktadır.*

```
Sınıflandırma Raporu:
              precision    recall  f1-score   support

           0       0.85      0.66      0.75       729
           1       0.98      0.99      0.99     13244

    accuracy                           0.98     13973
   macro avg       0.92      0.83      0.87     13973
weighted avg       0.98      0.98      0.98     13973
```



1) ***Scikit-Learn:***

Veri seti üzerinde temel makine öğrenmesi algoritmalarını (Random Forest, Decision Tree, vb.) uygulamak için kullanılmıştır. scikit-learn, veri setinin ön işlenmesinden modelin eğitilmesine kadar birçok işlemi kolayca gerçekleştirilebilir hale getiren kapsamlı bir araçtır.

2) ***Random Forest***

Random Forest, denetimli öğrenme yöntemlerinden biri olan bir ensemble algoritmasıdır ve sınıflandırma ve regresyon problemlerinde yaygın olarak kullanılır. Random Forest, birden fazla karar ağacının birleşiminden oluşur ve her bir ağaç, veri setinin farklı alt kümesi üzerinde eğitim alır. Sonuç olarak, bu topluluk yöntemi, bireysel ağaçların hatalarını dengeleyerek daha doğru ve güvenilir tahminler yapabilmektedir.

Proje kapsamında, Random Forest algoritması, anomali tespiti gibi problemlerde kullanılmıştır. Model, geçmiş veri setlerinden öğrenerek, yeni gelen verilerin normal mi yoksa anormal mi olduğunu sınıflandırmak için kullanılır. Her bir karar ağacı, verinin farklı özelliklerine göre kararlar alır ve Random Forest, her bir ağacın tahminini birleştirerek final kararını verir. Bu yöntem, overfitting (aşırı uyum) problemini azaltır ve genellikle yüksek doğruluk oranları elde edilmesini sağlar.

```
Model Doğruluğu: 0.9738066270664854

Classification Report:
              precision    recall  f1-score   support

           0       0.87      0.58      0.70       729
           1       0.98      1.00      0.99     13244

    accuracy                           0.97     13973
   macro avg       0.93      0.79      0.84     13973
weighted avg       0.97      0.97      0.97     13973


Özelliklerin Önem Dereceleri:
                     Importance
minimum_nights         0.133102
availability_30        0.104078
availability_365       0.081758
review_scores_rating   0.070260
property_type          0.068297
longitude              0.064910
latitude               0.063700
number_of_reviews      0.059370
host_name              0.056897
host_location          0.056801
...
host_since             0.044226
host_acceptance_rate   0.036550
room_type              0.035806
maximum_nights         0.022415
```

Random Forest, yüksek doğruluk oranları ve güçlü genelleme kapasitesinden dolayı büyük veri setlerinde ve karmaşık verilerde oldukça etkilidir. Bu algoritma, paralel işlem yapabilme yeteneği ile büyük veri setlerinin hızlı bir şekilde işlenmesini mümkün kılar ve özellikle modelin test edilmesi ve optimizasyonu konusunda esneklik sağlar.

3) **Pandas ve NumPy**:
Veri manipülasyonu ve analizinde kullanılan bu kütüphaneler, veri setlerinin işlenmesi ve hesaplamalar yapılmasında önemli rol oynamaktadır.

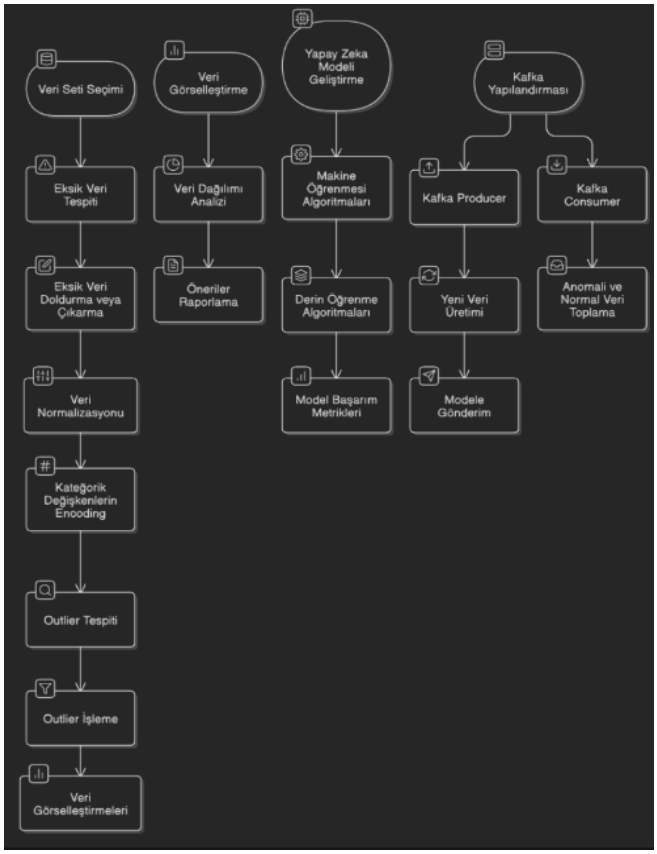
4) ***Matplotlib ve Seaborn****:*
Veri görselleştirme için kullanılan bu kütüphaneler, istatistiksel analizler ve grafikler oluşturmak için yaygın olarak kullanılır. Scatter plot'lar, histogramlar, heatmap'ler gibi görselleştirmelerle, verinin genel yapısı ve tespit edilen anomaliler görselleştirilmiştir.

G. ***Jupyter Notebook***

Jupyter Notebook, veri bilimcilerinin ve geliştiricilerin veri analizi, görselleştirme ve model geliştirme işlemlerini interaktif bir ortamda gerçekleştirmelerini sağlayan bir araçtır. Proje sürecinde, Jupyter Notebook kullanılarak her adım detaylı bir şekilde kaydedilmiştir. Verilerin görselleştirilmesi, modelin eğitilmesi ve sonuçların raporlanması gibi süreçler bu ortamda yapılmıştır. Ayrıca, Jupyter Notebook, Python kodlarının çalıştırılmasında, açıklamalar eklenmesinde ve sonuçların hızlı bir şekilde paylaşılmasında büyük kolaylık sağlamaktadır. Veri seti üzerinde yapılan tüm analizler ve geliştirme süreçleri, Jupyter Notebook üzerinden gerçekleştirildiği için proje geliştirme süreci daha verimli hale gelmiştir.

V. AKIŞ DİYAGRAMI

Bu çalışmada, geliştirilen sistemin işleyişini ve bileşenler arasındaki veri akışını görselleştirmek amacıyla bir akış diyagramı oluşturulmuştur. Akış diyagramı, kullanıcıdan gelen girdilerin işlenme sırasını detaylı bir şekilde göstermektedir.



VI. GELIŞTIRILMESI DÜŞÜNÜLENLER

Proje, gerçek zamanlı veri akışı ve anomali tespiti alanlarında önemli bir ilerleme kaydetmiş olsa da, gelecekte yapılacak geliştirmeler ile modelin doğruluğu ve genel işlevselliği daha da artırılabilir.

A. ***Derin Öğrenme Modellerinin Entegrasyonu:***

Projede kullanılan mevcut makine öğrenmesi algoritmalarının yanı sıra, daha karmaşık ve derin öğrenme yöntemlerinin entegrasyonu önemli bir geliştirme alanıdır. Özellikle LSTM (Long Short-Term Memory**)** ve Autoencoder gibi derin öğrenme modelleriyle anomali tespiti süreci daha da iyileştirilebilir. LSTM, zaman serisi verilerindeki uzun dönemli bağımlılıkları yakalayarak daha doğru tahminler yapabilir, bu da ev fiyatlarındaki anormallikleri tespit etme konusunda büyük fayda sağlayacaktır. Autoencoder ise, verinin normal yapısını öğrenerek, anormal davranışları daha hassas bir şekilde tanımlayabilir. Bu derin öğrenme modelleri, modelin genel doğruluğunu artırarak, gerçek zamanlı anomali tespiti sürecinde daha güvenilir sonuçlar elde edilmesine yardımcı olabilir.

B. ***Veri Setinin Zenginleştirilmesi:***

Projeye entegre edilen Londra ev fiyatları veri seti, belirli özelliklerle sınırlıdır. Gelecekte, daha fazla özelliğin eklenmesi, modelin genel performansını önemli ölçüde artırabilir. Örneğin, ekonomik göstergeler, bölgesel altyapı verileri, mevsimsel değişkenler gibi dışsal faktörlerin de dahil edilmesi, ev fiyatlarının daha doğru bir şekilde modellenmesini sağlayacaktır. Ayrıca, farklı veri

kaynaklarının kullanılması, modelin kapsamını genişletebilir ve veri çeşitliliği ile anomali tespiti sürecinin güvenilirliğini artırabilir. Bu şekilde, yalnızca ev fiyatları değil, aynı zamanda ev alım-satım trendleri, arz-talep dengesi gibi daha geniş veri setleriyle derinlemesine analizler yapılabilecektir.

C. *Model Performansının İyileştirilmesi:*

Modelin doğruluğu ve güvenilirliği, özellikle anomali tespiti ve tahminlerde önemli bir rol oynamaktadır. Derin öğrenme algoritmalarının entegrasyonu ile birlikte, model performansını artırmaya yönelik çeşitli teknikler uygulanabilir. Hyperparameter tuning (hiperparametre ayarı), farklı model mimarileri ve en iyi performans gösteren algoritmaların belirlenmesi, modelin genel başarısını önemli ölçüde iyileştirebilir. Ayrıca, modelin zaman içinde daha iyi sonuçlar üretmesi için modelin sürekli olarak yeniden eğitilmesi ve güncellenmesi sağlanabilir.

D. *Modelin Ölçeklenebilirliği ve Performans Optimizasyonu:*

Modelin verimliliği ve ölçeklenebilirliği, büyük veri setleri ile çalışırken önemli bir faktördür. Şu anda, Apache Kafka ve Spark gibi dağıtık sistemler kullanılarak gerçek zamanlı veri akışı sağlanmaktadır, ancak ileride daha büyük veri setleri ile çalışabilmek için performans optimizasyonu yapılabilir**.**

Veri işlemeyi hızlandıran algoritmalar, paralel işleme teknikleri ve dağıtık öğrenme yöntemleri sayesinde, modelin işlem kapasitesi artırılabilir. Bu sayede, daha geniş veri kümeleri üzerinde yüksek hızda işlem yapılabilir ve modelin performansı daha da iyileştirilebilir.

VII. SÖZDE KOD

- Başlangıç

Veri Seti Seçimi ve Ön İşleme

// "London Housing Prices" veri seti seçilir
// Eksik veriler kontrol edilir
// Eksik veriler doldurulur
// Veri normalize edilir
// Veri standartlaştırılır
// Kategorik veriler sayısal verilere dönüştürülür (Encoding)
// Anomaliler (outliers) tespit edilir ve işlenir
// Veri görselleştirmeleri yapılır (Histogram, Scatter Plot, Heatmap vb.)

Veri Görselleştirme

// Veri dağılımı analiz edilir
// Verinin genel yapısı hakkında öneriler yapılır
// Görselleştirmeler yapılır ve analiz edilir

Yapay Zeka Modeli Geliştirme

// Random Forest modeli ile eğitim yapılır
// Autoencoder ile derin öğrenme modeli eğitilir
// Modelin doğruluğu, precision, recall, F1 skoru hesaplanır
// Modelin başarı raporu oluşturulur

Kafka Yapılandırması

// Kafka producer yapılandırılır
// Yeni veriler Kafka'ya gönderilir
// Kafka consumer yapılandırılır
// Kafka topic'lerinden anomaliler tespit edilir

Spark Uygulaması

// Spark streaming başlatılır
// Kafka'dan veri alınır
// Spark üzerinde anomali tespiti yapılır
// Anomali tespit sonuçları Kafka'ya gönderilir

-Bitiş

VIII. SONUÇ

Bu proje, Londra ev fiyatları veri seti üzerinde gerçekleştirilen anomali tespiti süreci ile önemli bir başarıya imza atmıştır. Proje, büyük veri işleme ve gerçek zamanlı analiz konularındaki zorlukları etkili bir şekilde aşarak, doğru ve güvenilir sonuçlar elde etmiştir. Apache Kafka ve Apache Spark arasındaki entegrasyon, verilerin hızlı ve verimli bir şekilde işlenmesini sağlamış, veri akışının kesintisiz bir biçimde sağlanmasına olanak tanımıştır. Kafka, gerçek zamanlı veri akışını yöneterek, Spark'ın hızlı veri işleme kapasitesini beslemiş ve anomali tespiti sürecinde kritik bir rol oynamıştır.

Modelin başarım metrikleri, yüksek doğruluk oranları ile sonuçlanmış ve ev fiyatları veri setindeki anomali tespitleri başarıyla yapılmıştır. Bu tespitler, ev fiyatlarındaki sıra dışı değişikliklerin erken aşamalarda fark edilmesini sağlamış ve sistemin potansiyel olarak veri analistleri ve diğer karar vericiler için büyük bir değer taşımasına olanak tanımıştır. Modelin doğruluğu, kullanılan makine öğrenmesi algoritmalarının yanı sıra, derin öğrenme modelleri olan LSTM ve Autoencoder gibi yöntemlerle de güçlendirilmiş tespitler yapılabilmiştir.Gerçek zamanlı veri akışı ve anomali tespiti konusundaki başarı, projenin etkinliğini ve güvenilirliğini artırmış, işletmelerin potansiyel riskleri önceden tespit etmelerine yardımcı olmuştur. Ayrıca, kullanılan veri işleme altyapısının esnekliği ve ölçeklenebilirliği, gelecekteki projelerde daha geniş veri setlerinin işlenmesine olanak tanıyacak şekilde önemli bir temel oluşturmuştur.Projenin gelecekteki geliştirilmesi için birçok fırsat bulunmaktadır. Daha karmaşık modellerin eklenmesi ve daha büyük veri setlerinin kullanılmasına olanak sağlanması ile, anomali tespiti süreci daha da güçlendirilebilir. Ayrıca, daha fazla veri kaynağının entegrasyonu ve gerçek zamanlı izleme sistemlerinin eklenmesi, sistemin daha verimli çalışmasını sağlayacak ve kullanıcılar için daha etkileşimli bir deneyim sunacaktır.

IX. KAYNAKÇA

- Youtube
- Youtube
- Stackoverflow
- StackoverFlow
- StackoverFlow
- Youtube
- Hadoop
- Github
- Github