KÜTAHYA SAĞLIK BİLİMLERİ ÜNİVERSİTESİ
MÜHENDİSLİK VE DOĞA BİLİMLERİ FAKÜLTESİ

YAPAY ZEKA DERSİ

Recurrent Neural Networks(RNN) Kullanarak Hava Durumu Tahmini

Yusuf KIZILGEDİK **2118121003**

Friday 14th June, 2024

Abstract

Bu çalışmada, hava durumu tahmininde kullanılan yapay zeka teknikleri ve özellikle tekrarlayan sinir ağlarının (RNN) rolü üzerinde durulmuştur. Çalışmanın amacı, geçmiş hava durumu verilerini kullanarak gelecekteki hava durumunu tahmin etmek için RNN tabanlı bir yapay zeka uygulaması geliştirmektir. Hava durumu tahmini için RNN'nin neden tercih edildiği, kullanılan algoritmalar ve uygulama performansı detaylı bir şekilde açıklanmıştır. Elde edilen sonuçlar, bu uygulamanın hava durumu tahmininde başarılı olduğunu göstermiştir. Ayrıca, diğer algoritmalarla yapılan karşılaştırmalar da değerlendirilmiştir.

Anahtar kelimeler: Yapay Zeka,RNN,LSTM,Hava Durumu Tahmini, Keras

1. **Giriş**

Bu çalışma, hava durumu tahmini için yapay zeka ve özellikle tekrarlayan sinir ağlarının kullanımını incelemektedir. Hava durumu tahmini, zamansal verilerin analizi gerektiren karmaşık bir alandır. Tekrarlayan sinir ağlarının, zamansal ilişkileri daha iyi anlamak için kullanılabilecek çeşitli türleri vardır. Hava durumu tahmini konusunda RNN'nin avantajları ve bu çalışmanın neyi başarmayı hedeflediği burada açıklanmaktadır.

Bu çalışmada, hava durumu tahmininde yapay zeka uygulamalarının önemine dair önceden yapılan çalışmalar ve bu çalışmanın neden önemli olduğu da vurgulanmaktadır. Önceki çalışmalardaki eksiklikler ve bu çalışmanın çözmeyi hedeflediği problemler üzerinde durulmaktadır. Bu çalışmanın amacı, hava durumu tahmininde derin öğrenme tekniklerini kullanarak daha doğru tahminler elde etmektir.

2. **Literatür Araştırması**

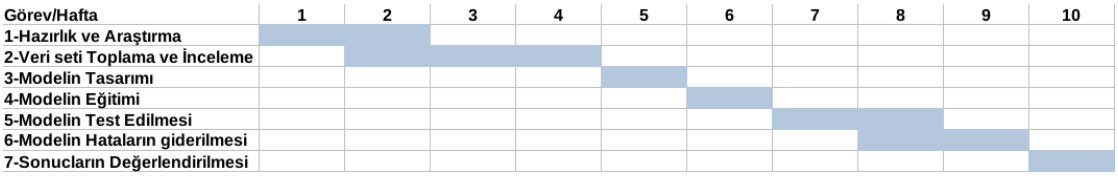
- Weather Prediction Using Recurrent Neural Networks Bu çalışmada, RNN'lerin sıcaklık, nem, rüzgar hızı gibi hava durumu özelliklerini tahmin etmedeki etkinliği araştırılmıştır. Yazarlar, RNN modellerinin özellikle zamansal verilerdeki bağımlılıkları yakalamada başarılı olduğunu ve bu nedenle hava durumu tahmini için uygun olduğunu vurgulamaktadır. RNN'ler, ardışık veriler arasındaki bağımlılıkları modelleyerek tahmin doğruluğunu artırabilirler[2].
- Long Short-Term Memory Recurrent Neural Network for Weather Prediction Bu çalışmada, hava durumu tahmini için LSTM (Long Short-Term Memory) adı verilen bir RNN türevi kullanılmıştır. LSTM'ler, uzun vadeli bağımlılık sorununu çözme yeteneğine sahiptir ve bu nedenle daha doğru tahminler yapabilir. Yazarlar, LSTM modellerinin sıcaklık, nem ve rüzgar hızı gibi parametrelerde daha düşük hata oranları gösterdiğini belirtmiştir. LSTM'ler, özellikle eksik verilerin doldurulması ve verilerin normalizasyonu gibi ön işleme teknikleriyle birleştiğinde yüksek performans sergiler[4].
- Forecasting Weather with Recurrent Neural Networks Bu araştırmada, RNN'lerin hava durumu tahmini için kullanılmasıyla ilgili bir derleme sunulmuştur. Farklı RNN mimarilerinin hava durumu tahmini üzerindeki etkileri incelenmiş ve gelecekteki çalışmalar için önerilerde bulunulmuştur. Çalışma, LSTM ve GRU (Gated Recurrent Unit) gibi farklı RNN

türlerinin performansını karşılaştırmış ve her birinin avantajlarını ve dezavantajlarını analiz etmiştir. Ayrıca, mevsimsel ve kısa vadeli tahminlerde bu modellerin nasıl uygulanabileceği tartışılmıştır[3].

3. Proje Yaklaşımı

- Veri Toplama: İlk adım, hava durumu verilerini toplamak olacaktır. Bu veriler, tarih, saat, sıcaklık, nem, rüzgar hızı, basınç gibi hava koşullarını içermelidir. Hava durumu verileri, kamu hizmeti sağlayıcılarından veya çeşitli hava durumu istasyonlarından alınabilir.
- Veri Ön İşleme: Toplanan verilerin doğruluğunu ve tutarlılığını sağlamak için ön işleme adımları uygulanmalıdır. Bu adımlar arasında eksik veya hatalı verilerin işlenmesi, veri normalizasyonu ve özellik mühendisliği yer alabilir.
- Model Seçimi: Ardından, hava durumu tahmini için bir RNN modeli seçilir. LSTM gibi uzun vadeli bağımlılığı ele alan RNN türevleri sıklıkla tercih edilir. Modelin mimarisi, girdi ve çıktı katmanları, saklı katmanlar ve hiperparametreler bu aşamada belirlenir.
- Model Eğitimi: Seçilen RNN modeli, toplanan ve önceden işlenen veriler kullanılarak eğitilir. Eğitim aşamasında, modelin hava durumu verilerini doğru bir şekilde tahmin etmesi için girdi-veri çiftleri üzerinde optimize edilir.
- Model Değerlendirmesi: Eğitilen model, belirli bir test veri kümesi üzerinde değerlendirilir. Modelin doğruluğu, ortalama karesel hata (MSE), ortalama mutlak hata (MAE) gibi performans ölçütleri kullanılarak değerlendirilir.
- Sonuçların Yorumlanması: Elde edilen sonuçlar analiz edilir ve yorumlanır. Modelin ne kadar başarılı olduğu, hangi hava koşullarını daha iyi tahmin ettiği ve potansiyel iyileştirme alanları gibi konular incelenir.

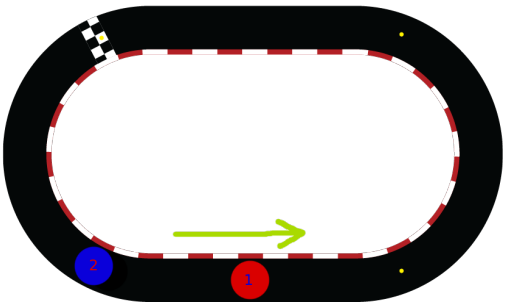
Şekil 1: GANTT Şeması



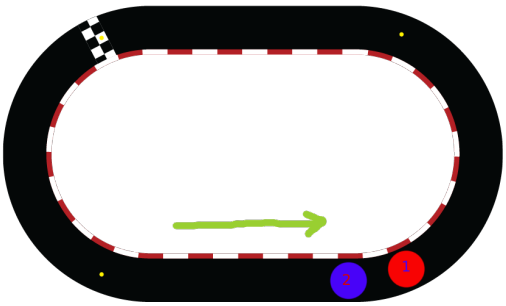
- **Hazırlık ve Araştırma:** RNN'ler hakkında temel bilgi edinme ve araştırma yapma, RNN'lerin gerçek hayattaki uygulamalarını inceleme,Proje kapsamı ve hedeflerinin netleştirilmesi.
- **Veri Seti Toplama ve İnceleme:** Hava durumu tahmini için uygun veri setlerini bulma ve toplama, Veri setlerini temizleme, işleme ve ön inceleme yapma[15][14].
- **Modelin Tasarımı ve Eğitimi:** RNN modelinin mimarisini belirleme ve tasarlama, Veri setlerini model için uygun formata dönüştürme, Modelin eğitimi için uygun algoritmaları seçme ve uygulama.
- **Modelin Test Edilmesi ve Ayarlanması:** Eğitilen modelin performansını değerlendirme ve test etme,Modelin hata analizi yapma ve iyileştirmeler için ayarlamalar yapma, Modelin doğruluğunu artırmak için optimizasyon yöntemlerini uygulama,
- **Sonuçların Değerlendirilmesi ve Raporlama:** Projenin sonuçlarını derleme ve değerlendirme, Elde edilen sonuçları raporlama ve sunum hazırlama, Projenin başarıları, sınırlamaları ve gelecekteki çalışmalar için önerileri tartışma.

4. **Neden RNN?**

Hava durumu tahmini gibi zamansal verilerin analizi için RNN (Recurrent Neural Networks), standart sinir ağlarına kıyasla daha uygun bir seçenek olabilir. Standart sinir ağları, veriler arasındaki ilişkileri analiz etmek için o anki verilere dayanır ve geçmiş verilerin zaman bağlamını göz ardı eder. Örneğin, bir yarış pistindeki iki aracın yarışını ele alalım: birinci görselde kırmızı araç, mavi araca kıyasla oldukça önde ve birincil konumda ilerlemektedir. Ancak, ikinci görselde kırmızı araç hala birinci sıradadır, ancak mavi araçla arasındaki mesafe oldukça azalmıştır. Bu noktada, standart sinir ağları, her iki görseli ayrı ayrı değerlendirir ve kırmızı aracın kazanacağını tahmin eder. Ancak, RNN, geçmiş verilerin zaman bağlamını dikkate alarak analiz yapar. Dolayısıyla, ikinci görseldeki durumda, mavi aracın mesafeyi kapatma eğiliminde olduğunu ve sonucun değişebileceğini tahmin eder. Bu nedenle, zaman serileri gibi zamansal verilerle çalışırken RNN, zaman bağlamını dikkate alarak daha doğru tahminlerde bulunabilir[6].



(a) 1.Göresel



(b) 2.Görsel

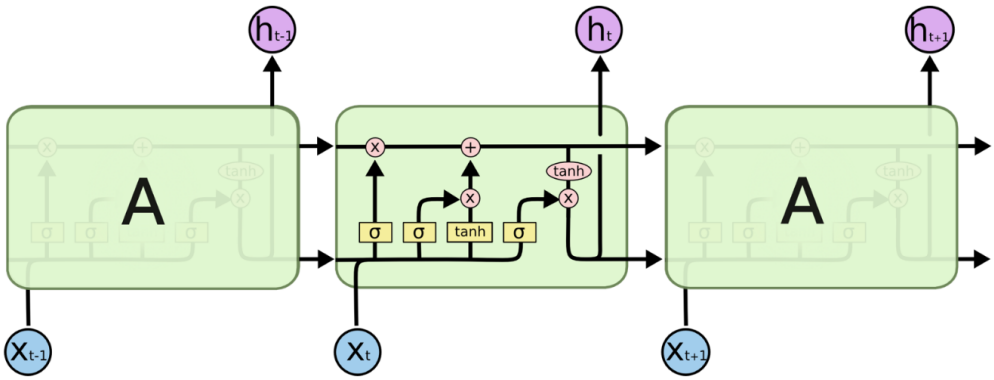
5. **RNN Türleri**

- **Vanilla RNN (Temel RNN):** Basitçe geçmiş zaman adımlarını dikkate alarak bir sonraki adımın tahminini yapar. Ancak, zamanla kaybolan (vanishing gradients) sorunu nedeniyle uzun vadeli bağlantıları yakalamakta zorlanabilir.
- **Long Short-Term Memory (LSTM):** LSTM, zaman serisi verilerinde uzun vadeli bağımlılıkları daha etkili bir şekilde modellemek için tasarlanmıştır. Hava durumu tahmini gibi zaman serisi problemleri için çok kullanışlıdır.
- **Gated Recurrent Unit (GRU):** GRU, LSTM'e benzer şekilde uzun vadeli bağımlılıkları ele almak için tasarlanmıştır. Ancak, LSTM'e kıyasla daha az parametreye sahiptir, bu da eğitim sürecini hızlandırabilir.
- **Bidirectional RNN (BRNN):** BRNN, geçmiş ve gelecek zaman adımlarını ayrı ağlarda işleyerek, her bir zaman adımında hem önceki hem de sonraki verileri dikkate alır. Bu şekilde, mevcut zamandaki tahminler için daha kapsamlı bir bağlam sağlar.
- **Deep RNN:** Birden fazla katmana (layer) sahip RNN'lerdir. Daha derin yapılar, daha karmaşık zaman serisi ilişkilerini yakalamak için kullanılabilir.
- **CNN-RNN Hibrit Modeller:** Bazı durumlarda, zaman serisi verilerini işlemek için Convolutional Neural Networks (CNN) ile RNN'leri birleştiren modeller de etkilidir. Özellikle uzaysal yapıyı (örneğin, hava durumu haritaları) dikkate almak istediğinizde, bu tür hibrit modeller kullanışlı olabilir.

Hava durumu tahmini gibi zaman serisi problemlerinde sık kullanılan iki RNN türü LSTM ve GRU'dur. Kendi projemde hangisini kullanacağıma karar vermek amacıyla, koduma ikisini de entegre edip sonuçları karşılaştıracağım. Bu karşılaştırmaya göre hangisini kullanacağıma karar vereceğim.

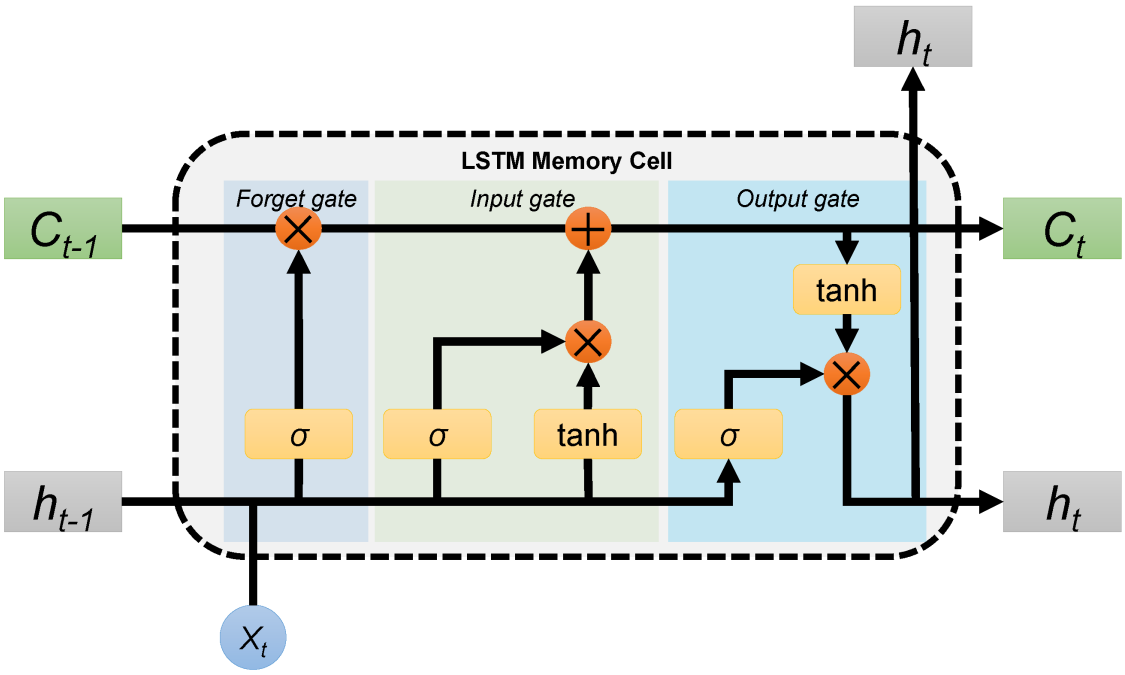
6. LSTM

Uzun Kısa-Term Hafıza (LSTM), hava durumu tahmini gibi zaman serisi problemlerinde oldukça etkili olan bir tür Recurrent Neural Network (RNN)'dir. LSTM, sıralı verilerin analizi için kullanılır ve özellikle zaman serileri gibi uzun süreli bağımlılıklar içeren verilerin işlenmesinde etkilidir[5].



Şekil 3: LSTM yapısı

- Şekil 3'de, LSTM mimarisin anlatımı kolaylığı icin bir görsel verilmiştir. Görselde 3 adet LSTM hücresi bulunmaktadır. Bu LSTM hücreleri, önceki hücrenin çıktılarını, yani h ve C değerlerini, x girdisi ile birleştirerek yeni h ve C çıktılarını oluşturur. Son LSTM hücresinin çıktısı da sonucu verir.



Şekil 4: LSTM[11]

- LSTM nin doğru gösterimi şekil 4 de verilmiştir.Bu modelimizde aslında bir hücre kendi kendine t zamanında bir girdiyi kullanarak t zamanında c ve h cıktılarını oluşturuyor.C ve H yi bir sonraki adımda kullanmak için kendisiyle paylaşıyor.Böylelikle bir sonraki adımda yeni bir c,h ve yeni bir x girdisiyle calışıyor.
- Bir LSTM ağı oluşturmanın ilk adımı, gerekli olmayan ve hücreden çıkarılacak olan bilgileri belirlemektir. Bu veri tanımlama ve hariç tutma işlemine, $t-1$ zamanında son LSTM biriminin (h_{t-1}) çıktısını ve t zamanında mevcut girdiyi (X_t) alan sigmoid fonksiyonu karar verir. Ek olarak, sigmoid fonksiyonu eski çıktının hangi kısmının elenmesi gerektiğini belirler. Bu kapıya unutma kapısı (veya f_t) denir[16].

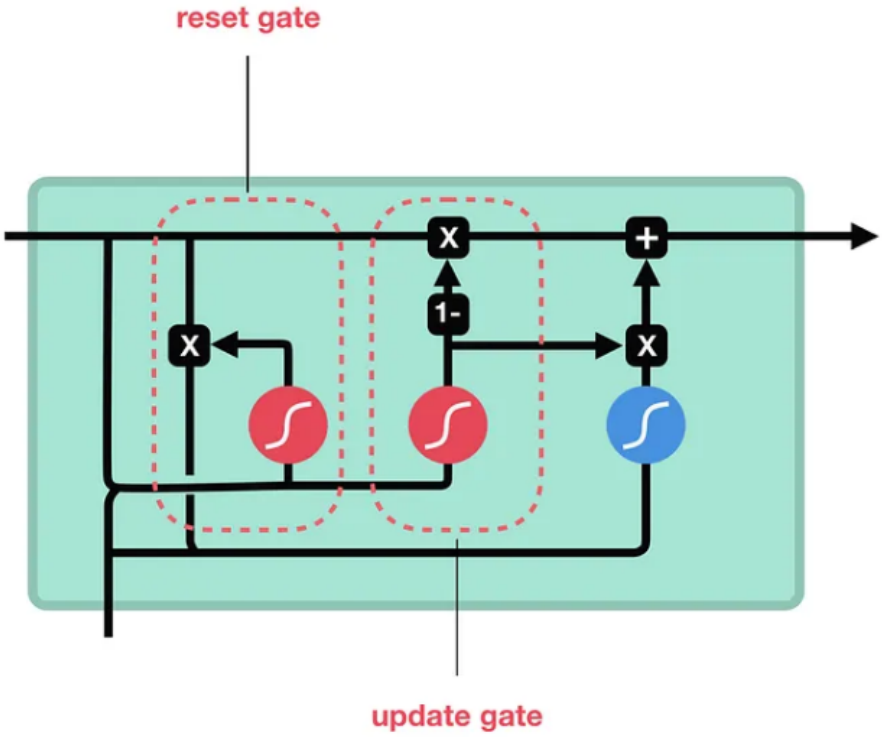
- Giriş Kapısı yeni girişten (X_t) gelen bilgileri kararlaştırır, depolar ve ayrıca hücre durumunu günceller. Bu adım, sigmoid katman ve ikinci tanh katmanı olmak üzere iki bölümden oluşur. İlk olarak, sigmoid katmanı, yeni bilginin güncellenip güncellenmeyeceğine (0 veya 1) karar verir ve ikinci olarak, tanh fonksiyonu, önem düzeylerine (-1 ile 1 arasında) karar vererek, geçen değerlere ağırlık verir. Yeni hücre durumunu güncellemek için bu iki değer çarpılır. Bu yeni bellek daha sonra eski belleğe (C_{t-1}) eklenir ve sonuç olarak C_t elde edilir[16].
- Sigmoid katmanına sahip çıkış kapısı, hücre durumunun hangi kısımlarının çıktıya iletileceğine karar verir. Ardından, sigmoid kapısından gelen çıktı (O_t), hücre durumu (C_t) üzerinden tanh katmanı tarafından oluşturulan yeni değerle çarpılır[16].

```python
# Bu fonksiyon bir LSTM hücresini tanimlar.
def LSTMCELL(prev_ct, prev_ht, Xt):



    combine = prev_ht + Xt #1-İlk olarak, önceki gizli durum ile mevcut giriş birleştirilir.

    # Unutma kapisini kontrol eder.
    ft = forget_layer(combine)

    # Aday hücre durumunu hesaplar.
    candidate = candidate_layer(combine) #Aday, hücre durumuna eklemek için olasi değerleri tutar.


    # Giriş kapisini kontrol eder.
    it = input_layer(combine)

    # Mevcut zaman dilimindeki bellek durumunu hesaplar.
    Ct = prev_ct * ft + candidate * it

    # Çikiş kapisini kontrol eder.
    Ot = output_layer(combine)

    # Mevcut zaman dilimindeki gizli durumunu hesaplar.
    ht = Ot * tanh(Ct)

    # `Ct` ve `ht`'yi fonksiyonun çiktisi olarak döndürür.
    return ht, Ct

# Başlangiç bellek ve gizli durumlari.
ct = [0, 0, 0]
ht = [0, 0, 0]

# Her zaman diliminde fonksiyonu çağirir.
for Xt in inputs:
    Ct, ht = LSTMCELL(ct, ht, Xt)
```

Şekil 5: LSTM python pseudo code [13]

7. GRU

GRU modeli, "Gated Recurrent Unit" anlamına gelir ve nöral ağlar içinde popüler bir tür tekrarlayan sinir ağı (RNN) birimidir. GRU, özellikle doğal dil işleme, zaman serisi analizi ve benzeri ardışık veri tabanlı görevlerde kullanılır.GRU, LSTM'nin daha basit ve hesaplama açısından daha verimli bir alternatifi olarak geliştirildi.GRU, LSTM'nin sahip olduğu bazı karmaşık yapıları ve kapıları sadeleştirerek benzer bir performans sunar



Şekil 6: GRU[17]

Güncelleme Kapısı: Güncelleme kapısı, hücre durumunun ne kadarının bir sonraki durum için korunacağına ve ne kadarının güncelleneceğine karar verir. Güncelleme kapısı değeri, önceki gizli durum ve mevcut girişe bağlı olarak hesaplanır. Bu kapı, hücre durumunun ne kadarının korunacağını kontrol eder.

Reset Kapısı Reset kapısı, hücre içindeki geçmiş bilginin ne kadarının unutulacağına veya sıfırlanacağına karar verir. Reset kapısı da önceki gizli durum ve mevcut girişe bağlı olarak hesaplanır. Bu kapı, eski gizli durumun yeni bilgiye ne ölçüde karışacağını belirler.

```
# GRU Pseudocode

# Ağırlık ve biasları başlat
Wz, Uz, bz = initialize_weights_and_biases()
Wr, Ur, br = initialize_weights_and_biases()
Wh, Uh, bh = initialize_weights_and_biases()

# Her zaman adımı için ileri geçiş
for t in range(T):
    z_t = sigmoid(Wz * x_t + Uz * h_(t-1) + bz)  # Güncelleme kapısı
    r_t = sigmoid(Wr * x_t + Ur * h_(t-1) + br)  # Sıfırlama kapısı
    h_hat_t = tanh(Wh * x_t + Uh * (r_t * h_(t-1)) + bh)  # Aday gizli durum
    h_t = (1 - z_t) * h_(t-1) + z_t * h_hat_t  # Nihai gizli durum

    # Bu zaman adımı için gizli durumu çıktı olarak ver
    output_h_t(h_t)
```

Şekil 7: GRU Kod

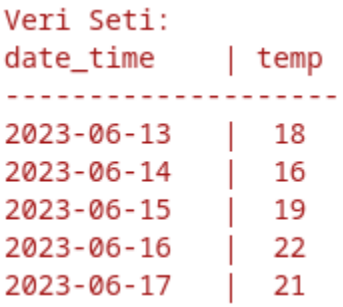
- GRU, genellikle LSTM'e göre daha hızlı ve hafif çalışır çünkü daha az karmaşıktır.
- GRU'nun daha düşük bellek tüketimi vardır, bu nedenle veri ve model boyutları arttığında daha iyi çalışabilir.
- LSTM, karmaşık zaman serisi modellerinde daha iyi genelleme yapabilir, ancak genellikle GRU da benzer performans sergiler.
- GRU, LSTM'e göre daha basittir. Unutma ve giriş kapıları yerine güncelleme ve sıfırlama kapıları vardır. Bu da daha az parametre ve daha hızlı çalışma anlamına gelir.
- GRU genellikle daha hızlı ve daha hafif olması nedeniyle tercih edilir, ancak LSTM'nin uzun vadeli bağımlılıkları daha iyi modelleyebilme yeteneği vardır.

- **Hava Durumu Tahmini Kod**

```python
import pandas as pd
import numpy as np
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense

# Excel dosyasını okuyun
df = pd.read_excel('sakarya2018-2024.xlsx')

# Veri setini işleyin
df['date_time'] = pd.to_datetime(df['date_time'])
df.set_index('date_time', inplace=True)

# Veriyi ölçeklendirin
scaler = MinMaxScaler()
scaled_data = scaler.fit_transform(df)

# Eğitim ve test veri setlerini ayırın
X = scaled_data[:-1]
y = scaled_data[1:]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, shuffle=False)

# Eğitim ve test veri setlerini yeniden şekillendirin
X_train = X_train.reshape((X_train.shape[0], X_train.shape[1], 1))
X_test = X_test.reshape((X_test.shape[0], X_test.shape[1], 1))

# LSTM modelini oluşturun
model = Sequential([
    LSTM(50, activation='relu', input_shape=(X_train.shape[1], X_train.shape[2])),
    Dense(1)
])

model.compile(optimizer='adam', loss='mse')

# Modeli eğitin
model.fit(X_train, y_train, epochs=100, batch_size=1, verbose=1)

# Önümüzdeki 4 günün tahminini yapın
future_days = 4
future_dates = pd.date_range(start=df.index[-1], periods=future_days + 1)[1:]  # Bugünden itibaren 4 gün ileri

# Son gözlemi alın ve ölçeklendirin
last_data = scaled_data[-1]

# Tahminleri depolamak için bir dizi oluşturun
predicted_values = []

# Gelecekteki her bir gün için tahmini yapın
for _ in range(future_days):
    # Ölçeklendirilmiş veriyi yeniden şekillendirin
    X_new = np.array(last_data.reshape(1, -1, 1))
    # Tahmini yapın
    prediction = model.predict(X_new)
    # Ölçeklendirmeyi tersine çevirin
    prediction = scaler.inverse_transform(prediction)
    # Tahmini değeri saklayın
    predicted_values.append(prediction[0][0])
    # Tahmin edilen değeri son veriye ekleyin ve ilk girdiyi kaldırın
    last_data = np.append(last_data[1:], prediction[0][0])

# Tahmin edilen sıcaklık değerlerini ekrana yazdırın
for date, temp in zip(future_dates, predicted_values):
    print(f"{date.date()}: {temp:.2f} °C")
```

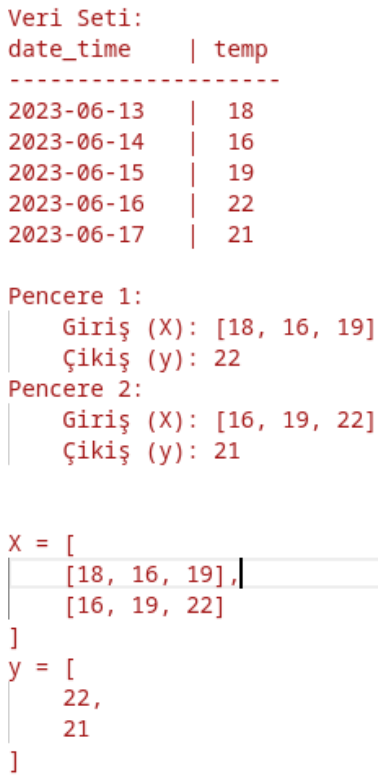
Şekil 8: Hava Durumu Tahmini Temel Kod [1]

  - Yapay sinir ağlarının türlerinden biri olan Uzun Kısa Süreli Bellek (LSTM) modelini kullanan bir Python kodu yazdım. Başlangıçta, bu kod doğru tahminler yapmıyordu ve sonuçlar beklenenin oldukça dışında, tuhaf değerler üretiyordu. Bu durumu düzeltmek ve algoritmanın performansını artırmak için bir dizi iyileştirme yaptım.



Şekil 9: Veri Seti

– **Veri Ön İşleme** Herhangi bir makine öğrenimi projesinde olduğu gibi, veri ön işleme adımı hava durumu tahmini için de kritiktir. İlk adım olarak, hava durumu veri setimizi alırız. Örneğimizde, hava durumu ölçümleri Sakarya şehrimizin 2018-2024 yılları arasındaki tarih bilgilerini içeren bir veri seti kullanıyoruz. Veri setimizi Pandas kütüphanesi aracılığıyla yükleriz ve tarih saat sütununu indeks olarak ayarlarız. Daha sonra, veriyi ölçeklendirme işlemine tabi tutarız. Bu, tüm değerleri belirli bir aralığa (genellikle [0, 1] aralığına) yeniden ölçeklendirir ve modelin daha iyi performans göstermesine yardımcı olur[15].



Şekil 10: Pencere Veri

- **Veriyi Window Haline Getirme** LSTM gibi zaman serisi modelleriyle çalışırken, veriyi pencere (window) haline getirmek yaygın bir uygulamadır. Bu, belirli bir zaman dilimindeki verileri bir araya getirerek, modelin zamansal ilişkileri daha iyi öğrenmesine olanak tanır. Örneğin, son üç günün hava durumu verilerini kullanarak, bir sonraki günün sıcaklık tahminini yapabiliriz. Bu işlem, bir veri penceresi boyunca belirli bir adımda gezinerek gerçekleştirilir. Yukarıdaki görselde, verinin pencere haline getirilmesini gösteren bir örnek bulunmaktadır. Her adımda, belirli bir pencere boyutu alınır ve bu pencere boyunca sıcaklık verileri toplanır. Bu, X ve y verilerini oluşturur ve modelin eğitimi için kullanılır.
- **Dropout** Aşırı öğrenmeyi önlemek için yaygın olarak kullanılan bir tekniktir. Bu teknik, rastgele belirlenen bir oranda ağdaki nöronların çıkartılmasını sağlar. Bu, ağın genelleştirme yeteneğini artırır ve aşırı uyumun önüne geçer. Modelimizde Dropout katmanları kullanarak, her eğitim döneminde belirli bir oranda nöronları rastgele devre dışı bırakırız.

- **EarlyStopping** Modelin eğitimini belirli bir kriter karşılandığında durduran bir geri çağrıdır. Bu, modelin aşırı uyuma meyilli olduğu durumlarda ağın eğitimini zamanında durdurarak, ağın genelleme yeteneğini artırır. Örneğin, modelin kaybı artık azalmıyorsa veya doğrulama kaybı artmaya başlarsa, eğitimi durdurabiliriz.
- **Model Karmaşıklığının Artırılmas** Modelin karmaşıklığını artırmak için daha fazla LSTM katmanı eklendi. LSTM katmanları, zaman serisi verilerindeki karmaşık ilişkileri yakalamak için kullanılır. Modelin karmaşıklığını artırmak, daha fazla öğrenme kapasitesi sağlayabilir ve daha karmaşık veri yapılarını daha iyi modelleyebilir.
- **Daha Fazla Epoch Sayısının Kullanılması** Modelin eğitim sürecinde daha fazla epoch (iterasyon) sayısı kullanıldı. Daha fazla epoch, modelin daha fazla öğrenme fırsatı elde etmesini sağlar. Bu, modelin daha iyi uyum sağlamasını ve daha doğru tahminler üretmesini sağlayabilir.

– **MinMAxScaler**

Makine öğrenmesinde, veri ölçekleme önemli bir adımdır. MinMaxScaler, verileri belirli bir aralığa dönüştürmek için kullanılır, genellikle 0 ile 1 arasına.Bu yöntem, özellikle verilerin belirli bir aralıkta olmasının önemli olduğu durumlarda kullanışlıdır[7].

$$X_{\text{scaled}} = \frac{X - X_{\min}}{X_{\max} - X_{\min}} \times (\max - \min) + \min[7]$$

```python
import pandas as pd
import numpy as np
from sklearn.preprocessing import MinMaxScaler
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, Dropout
from tensorflow.keras.callbacks import EarlyStopping

# Excel dosyasını okuyun
df = pd.read_excel('sakarya2018-2024.xlsx')

# Veri setini işleyin
df['date_time'] = pd.to_datetime(df['date_time'])
df.set_index('date_time', inplace=True)

# Veriyi ölçeklendirin
scaler = MinMaxScaler()
scaled_data = scaler.fit_transform(df)

# Öğrenme ve test veri setlerini ayırın
train_size = int(len(scaled_data) * 0.8)
train_data, test_data = scaled_data[:train_size], scaled_data[train_size:]

# Veriyi pencerele (window) haline getirin
def create_dataset(data, window_size):
    X, y = [], []
    for i in range(len(data) - window_size):
        X.append(data[i:i+window_size])
        y.append(data[i+window_size])
    return np.array(X), np.array(y)

window_size = 3  # Pencere boyutu
X_train, y_train = create_dataset(train_data, window_size)
X_test, y_test = create_dataset(test_data, window_size)

# LSTM modelini oluşturun
model = Sequential([
    LSTM(100, activation='relu', return_sequences=True, input_shape=(X_train.shape[1], X_train.shape[2])),
    Dropout(0.2),
    LSTM(100, activation='relu', return_sequences=True),
    Dropout(0.2),
    LSTM(100, activation='relu', return_sequences=True),
    Dropout(0.2),
    LSTM(100, activation='relu'),
    Dropout(0.2),
    Dense(1)
])

model.compile(optimizer='adam', loss='mse')
# Erken durdurma callback'i ekleyerek modelin eğitimini iyileştirin
early_stopping = EarlyStopping(monitor='val_loss', patience=10, restore_best_weights=True)

# Modeli eğitin
history = model.fit(X_train, y_train, epochs=100, batch_size=32, verbose=1, validation_data=(X_test, y_test), callbacks=[early_stopping])

# Önümüzdeki 7 günün tahminini yapın
future_days = 7
future_dates = pd.date_range(start=df.index[-1], periods=future_days + 1)[1:]  # Bugünden itibaren 4 gün ileri

# Son pencereyi alın
last_window = scaled_data[-window_size:].reshape(1, window_size, 1)

# Tahminleri yapın
predicted_values = []
for _ in range(future_days):
    prediction = model.predict(last_window)[0]
    predicted_values.append(prediction)
    last_window = np.append(last_window[:, 1:, :], prediction.reshape(1, 1, 1), axis=1)

# Tahmin edilen sıcaklık değerlerini ölçekten çıkarın
predicted_values = scaler.inverse_transform(predicted_values)

# Tahmin edilen sıcaklık değerlerini ekrana yazdırın
for date, temp in zip(future_dates, predicted_values):
    print(f"{date.date()}: {temp[0]:.2f} °C")
```

Şekil 11: Hava Durumu Tahmini geliştirilmiş Kod [1]

– Modelde gerçekleştirdiğim iyileştirmelerden sonra, sonuçların ne kadar geliştiğini belirlemek için eski ve yeni sonuçları karşılaştırdım.

Table 1: İyileştirme sonunda çıktı karşılaştırılması

TARİH	**Gerçek değer**	**kod1**	**kod2**
2023-11-24	9	8.97	8.64
2023-11-25	9	565.90	9.01
2023-11-26	10	27567.37	9.44
2023-11-27	2	1292014.62	9.65
2023-11-28	2	60554036.00	9.90

Son geliştirmelerimiz sonucunda, modelimizdeki çıktılarda önemli bir doğruluk artışı elde ettik.

– LSTM ile GRU karşılaştırmıştık Geliştirdiğimiz bu koda ikisinide entegre edip sonuçlarını karşılaştırdım

Table 2: LSTM ve GRU çıktıları

TARİH	**Gerçek değer**	**GRU**	**LSTM**
2023-11-24	9	8.62	8.64
2023-11-25	9	8.49	9.01
2023-11-26	10	8.44	9.44
2023-11-27	2	8.36	9.65

– İleriki geliştirmelerimde projemin daha iyi sonuç vermesi için LSTM ile devam etmeye karar verdim.

9. **RNN ile Küresel Isınma**

İklimin giderek değişmesi RNN tabanlı hava durum tahmin uygulamasının öngörü yapmasını daha zor hale getirebilir.Bu nedenle , hava durumu tahmin projelerinde küresel ısınmanın etkilerini dikkate almak ve modelleri daha esnek ve uyarlanabilir hale getirmek, tahminlerin doğruluğunu artırmak için kritik öneme sahiptir[10].

10. **Veri Çeşitliliği**

Modelimizin daha iyi bir performans göstermesi için veri çeşitliliğini artırmak istedik ve bu şekilde daha iyi bir sonuç almayı amaçladık. Ancak bunu yaparken, boyut uyumsuzluğundan dolayı bir hatayla karşılaştık ve hatayı çözemeyince tüm veri çeşitlerini hesaplattırdık.

Table 3: Ortalama Hava Koşulları

Tarih	**Veri Türü**	**Sıcaklık**	**Rüzgar Hızı**	**Rüzgar Yönü**	**Nem**
2023-11-26	Gerçek	6.1	2	168	60
2023-11-26	Tahmin	6.7	1.3	172	60.5
2023-11-27	Gerçek	1.2	1.5	191.5	51
2023-11-27	Tahmin	8.65	1.3	175	57
2023-11-28	Gerçek	5.5	1.3	150	51
2023-11-28	Tahmin	6.5	1.3	172	61
2023-11-29	Gerçek	9.7	1.6	145.6	64.4
2023-11-29	Tahmin	6.6	1.3	170	59

11. **Boyut uyumsuzluğu nedir?**

    Veri analizi veya makine öğrenimi gibi alanlarda, veri işleme adımlarında boyut uyumsuzlukları sıkça karşılaşılan sorunlardan biridir.Bir modelin başarılı bir şekilde çalışabilmesi için, modelin giriş verisiyle çıkış verisi arasında uygun boyut uyumu sağlanmalıdır. Giriş verisi, modelin beklediği şekle (shape) sahip olmalıdır; aynı şekilde, modelin çıkışı da beklenen bir şekilde oluşturulmalıdır. Ancak, giriş ve çıkış boyutları arasında bir uyumsuzluk varsa, modelin doğru şekilde çalışması engellenebilir ve genellikle 'boyut uyumsuzluğu' hatası (shape mismatch error) ile karşılaşılır.

12. **Lstm Boyut Uyumsuzluğu**

    Modelde giriş ve cıkış verilerinin boyutlarının uymamasından dolayı boyut uyuşmazlığı hatası alındı.

    - İlk başta, modelin sonuçlarını karşılaştırmak için boyut uyumsuzluğunu gidermeden önce, tüm verilerin tahmin edilmesini istedim. Böylece herhangi bir boyut uyumsuzluğu olmadan sonuçları görebilirdim.
    - Bu projede sadece sıcaklık değerlerine ihtiyacımız vardı. Bu gereksinimi karşılamak için boyut uyumsuzluğunu çözmek için adımlara başladım.
    - İlk olarak `model.predict(last_window)` çağrısından dönen tahminin şekli ve değeri kontrol edildi. Python'un shape özelliği kullanılarak ekrana yazdırıldı.

```
# Tahminin şeklini ve değerini kontrol etme
print("Prediction shape:", prediction.shape)  # Prediction'ın şeklini yazdırın
print("Prediction value:", prediction)        # Prediction değerini yazdırın
```

Şekil 12: kod

- Kodda np.append() işlemi, `last_window`'ın sonuna prediction'ı eklemek için kullanılıyor. Zaman birimi olarak son örneği ve tahminlenen değerler arasında bir ilişki olduğundan, tahminlenen değeri `last_window`'ın sonundaki bir zaman birimi olarak yerine koyarak hatayı cözme hedeflendi ama yeni bir hata çıktı

```
# Last window'ı güncelleme
last_window[:, -1, 0] = prediction  # Son zaman birimini tahmin edilen değerle güncelle
```

Şekil 13: kod

- `predict(scaler.inverse_transform())` işlemi, giriş olarak bir matris bekler, bu nedenle prediction'ın şeklini (1, 1) değil (1, 4) yapmalıyız. Bunun için prediction'ın şeklini değiştirmek gerekti.

```
# Tahmin edilen sıcaklık değerlerini ölçekten çıkarın
predicted_values = scaler.inverse_transform(prediction.reshape(1, -1))  # prediction'ın şeklini (1, 4) yap
```

Şekil 14: kod

Bu adımlardan sonra Hata alınmadan istediğimiz değer alındı

13. Lstm Boyut Uyumsuzluğu Bir Diğer Yöntem

- İlk başta `predicted_values` listesi oluşturuluyor ve her tahmin edilen değer bu listeye ekleniyor. Sonra, `last_window` güncellenmeden her tahminden sonra ilerliyoruz. Tüm tahminler toplandıktan sonra, bu değerler geri ölçeklendirilip (`inverse_transform`) gerçek dünyadaki değerlere dönüştürülüyor. Ancak, bu işlem sonrasında tahmin edilen sıcaklık değerleri her gün aynı sonucu veriyordu. Bu durumda bir hata olduğunu düşünerek yeni bir yöntem aramaya koyuldum. `last_window`'u güncelleme kararı aldım ve sıcaklık değerini alırken bunu uyguladım. Ancak, bu yöntemi uyguladığımda, veri setimdeki diğer veri türlerinin eğitime katkısı olmadığı için veri setimin çeşitliliği bir anlam ifade etmiyordu.
- Bir diğer yöntem ise tahmin yapılırken sadece sıcaklık (ilk sütun) değerini doldurduk ve diğer sütunları 0 olarak bıraktık.
- Veriyi ölceklendirmesinde yeni yöntemde, tahmin edilen sıcaklık değerlerini geri dönüştürmek için scaler.inversetransform() yerine sadece hedef değişkenin minimum ve maksimum değerlerini kullanıyoruz. Bu, işlemi daha basit ve kontrollü hale getiriyor, boyut uyuşmazlıklarını ve hataları önlüyor. Özellikle tek bir değişkeni geri dönüştürmek gerektiğinde, bu yöntem daha esnek ve güvenilir bir çözüm sunuyor.

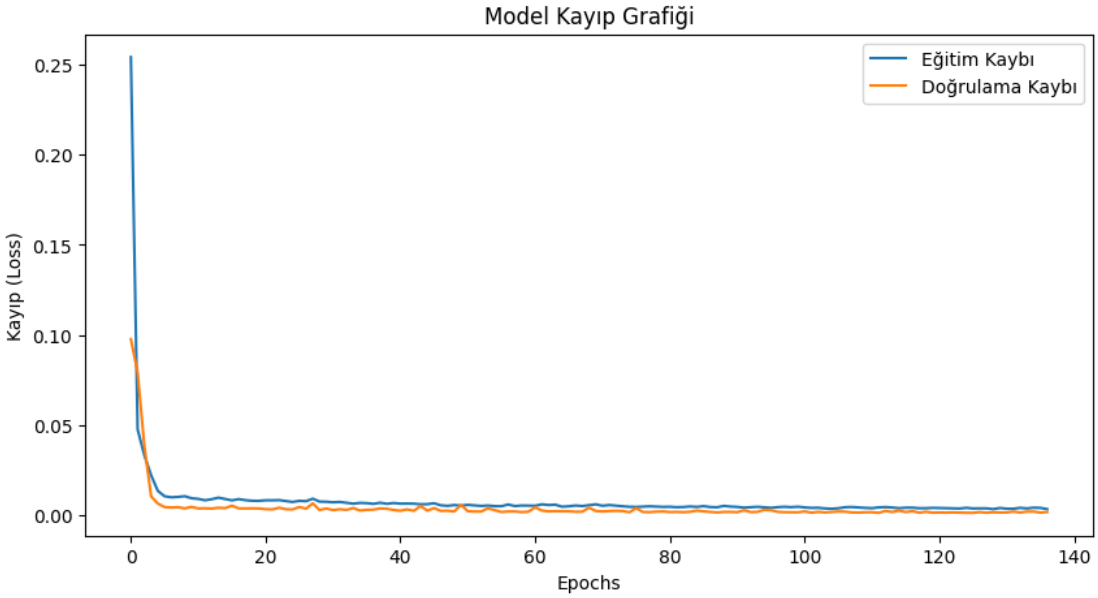
```
# scaler'ın hedef değişken için min ve max değerlerini sakla
temp_min = scaler.data_min_[0]
temp_max = scaler.data_max_[0]

# Tahmin edilen sıcaklık değerlerini ölçekten çıkarın
predicted_values = prediction * (temp_max - temp_min) + temp_min
```

Şekil 15: MAXMİN KOD

$$X_{\text{scaled}} = \frac{X - X_{\min}}{X_{\max} - X_{\min}} \times (\max - \min) + \min[7]$$

14. **Model Kayıp Grafiği**

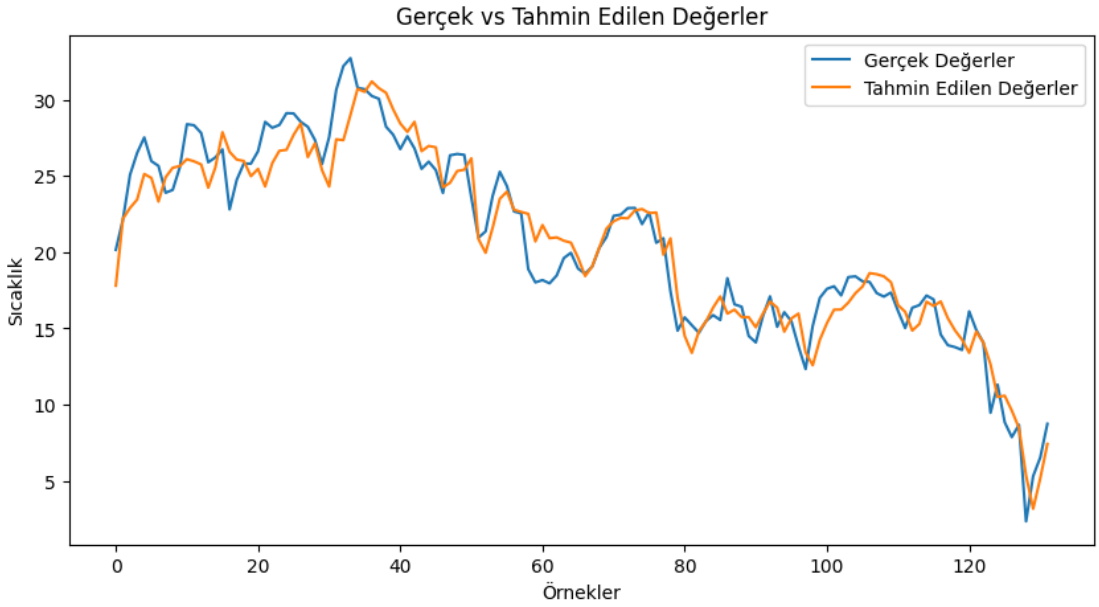
Model kayıp grafiği, modelin eğitim ve doğrulama kayıplarını gösterir.Model kayıp grafiği, modelin eğitim sürecinde nasıl performans gösterdiğini ve aşırı öğrenme (overfitting) veya eksik öğrenme (underfitting) olup olmadığını anlamak için önemlidir. Eğitim kaybının zamanla azaldığı ve doğrulama kaybının nasıl bir eğilim gösterdiği analiz edilir[8].



Şekil 16: Model Kayıp Grafiği

15. **Gerçek ve Tahmin Edilen Değerler Grafiği**

Bu grafik, gerçek hava durumu verileri ile model tarafından tahmin edilen değerleri karşılaştırır.Gerçek ve tahmin edilen değerler grafiği, modelin tahmin doğruluğunu görselleştirir ve modelin ne kadar başarılı olduğunu gösterir. Bu grafik, modelin kısa vadeli hava durumu tahminlerinde ne kadar hassas olduğunu anlamaya yardımcı olur[8].



Şekil 17: Gerçek ve Tahmin Edilen Değerler Grafiği

16. **Tahmin Çıktısı**

Modelin önümüzdeki dört gün için tahmin ettiği hava durumu verileri tablo şeklinde sunulur[12].

Date	Predicted Temperature (°C)
2023-11-23	8.712389945983887
2023-11-24	10.04599666595459
2023-11-25	11.872902870178223
2023-11-26	13.081089973449707

Şekil 18: Çıktı [9]

Table 4: Gerkcek Veri

DATE	**TEMP**
2023-11-23	8.7
2023-11-24	11.7
2023-11-25	10
2023-11-26	6.1

References

[1] Weather forecasting with recurrent neural networks. https://medium.com/analytics-vidhya/weather-forecasting-with-recurrent-neural-networks-1eaa057d70c3. Erişim Tarihi: 20 Mart 2024.

[2] James D. Annan and J. C. Hargreaves. Using causal discovery algorithms to investigate solar and enso influences on temperature. https://www.mdpi.com/2073-4433/10/11/668, 2019. Erişim Tarihi: 2024-06-12.

[3] Ana I. J. Aparecido, Joseph B. Fisher, Ji Chen, Jared Leffler, Christopher T. Martius, and Stefan Schnitzer. Tree community structural changes in response to logging in amazonia. https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0285713, 2023. Erişim Tarihi: 2024-06-12.

[4] Mustafa S. Celen. Operational excellence in business processes: Monitoring, diagnosis, and improvement. https://ieomsociety.org/detroit2020/papers/540.pdf, 2020. Erişim Tarihi: 2024-06-12.

[5] DataKapital. Yinelenen sinir ağları nedir? https://datakapital.com/blog/yinelenen-sinir-aglari-nedir/, Tarih bilinmiyor. Erişim Tarihi: 20 Mart 2024.

[6] Deep Learning Türkiye. Rnn nedir? nasıl Çalışır? https://medium.com/deep-learning-turkiye/rnn-nedir-nas%C4%B1l-%C3%A7al%C4%B1%C5%9F%C4%B1r-9e5d572689e1, 2018. Erişim Tarihi: 20 Mart 2024.

[7] Mehmet Can Duru. Özellik Ölçeklendirme: Standardscaler, robustscaler ve minmaxscaler karşılaştırması. https://medium.com/@mehmetcanduru/Ã¶zellik-Ã¶lÃ§eklendirme-standardscaler-robustscaler-ve-minmaxscaler-karÅ, 2020. Erişim Tarihi: 20 Mart 2024.

[8] Keras. Model training apis - fit method. https://keras.io/api/models/model_training_apis/#fit-method. Erişim Tarihi: 20 Mart 2024.

[9] Yusuf Kizilgedik. Github gist. https://gist.github.com/YusufKizilgedik/1edae4ff8630a49a29cb86fbc4e4c2de, 2023. Accessed: 2024-06-07.

[10] Türkiye Meteoroloji Genel Müdürlüğü. Hava ve İklim. https://mgm.gov.tr/FILES/genel/makale/4_Havaiklim.pdf, Tarih bilinmiyor. Erişim Tarihi: 20 Mart 2024.

[11] Author(s) Name. Article title. https://www.mdpi.com/2073-4441/12/1/175, 2020. Erişim Tarihi: 20 Mart 2024.

[12] Plotly. Table in python. https://plotly.com/python/table/. Erişim Tarihi: 20 Mart 2024.

[13] Towards Data Science. Illustrated guide to lstms and grus: A step-by-step explanation. https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85b Erişim Tarihi: 20 Mart 2024.

[14] UlaşAV. Hava durumu verileri. https://ulasav.csb.gov.tr/dataset/06-hava-durumu-verileri, 2023. Erişim Tarihi: 20 Mart 2024.

[15] UlaşAV. Hava durumu Ölçüm değerleri. https://ulasav.csb.gov.tr/dataset/42-hava-durumu-olcum-degerleri, 2023. Erişim Tarihi: 20 Mart 2024.

[16] Yazar Adı veya Yazarlar. Makalenin başlığı. https://dergipark.org.tr/en/download/article-file/1371423, Yayın Yılı. Erişim Tarihi: 20 Mart 2024.

[17] Mustafa Can Vurarer. Rnn, lstm ve gru modellerinin İncelemesi. https://medium.com/@mcvarer/rnn-lstm-ve-gru-modellerinin-incelemesi-f59a73499edb, 2022. Erişim Tarihi: 20 Mart 2024.