Bilgisayar Mühendisliği Gelişmeler

Final Raporu

Borsa Tahmini Uygulaması

Borsa Tahmini Uygulaması, seçilen borsa endeksleri için geçmiş verilere dayalı olarak fiyat tahmini yapan bir yazılımdır. Kullanıcı dostu bir arayüzle geliştirildi. Kullanıcıdan hangi borsaya dair veri ve grafiklere erişmek istediğini soruyor, sonra o günün tarihini girmesini istiyor. Bu girdilere göre yedi farklı çeşit çıktı ile faydalı veri ve grafikler oluşturuyor. Geliştirdiğimiz yazılım; verilerin işlenmesi, analiz edilmesi ve görselleştirilmesini içeriyor. Yatırımcılara daha bilinçli kararlar alabilmeleri için destek sunuyor.

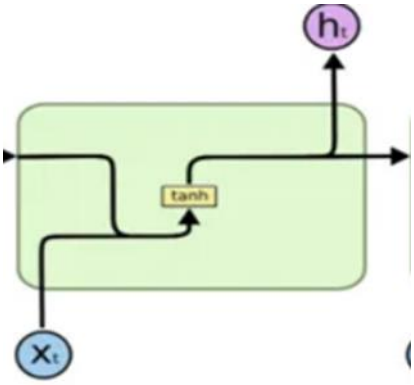
Kullanılan Teknolojiler ve Kütüphaneler

Projede aşağıdaki kütüphaneler ve teknolojiler kullanılmıştır:

- **Python**: Ana programlama dili.
- **Streamlit**: Kullanıcı arayüzü oluşturmak için.
- **TensorFlow/Keras**: LSTM modeli oluşturmak ve tahmin yapmak için.
- **yfinance**: Finansal veri toplamak için.
- **Pandas**: Veri analizi ve ön işleme için.
- **NumPy**: Matematiksel hesaplamalar için.
- **Matplotlib**: Grafikler ve veri görselleştirme için.
- **scikit-learn (MinMaxScaler)**: Verilerin normalizasyonu için.

LSTM Modeli Hakkında

LSTM, Özyinelemeli Sinir Ağı (RNN) ‘nın gelişmiş versiyonudur.



RNN

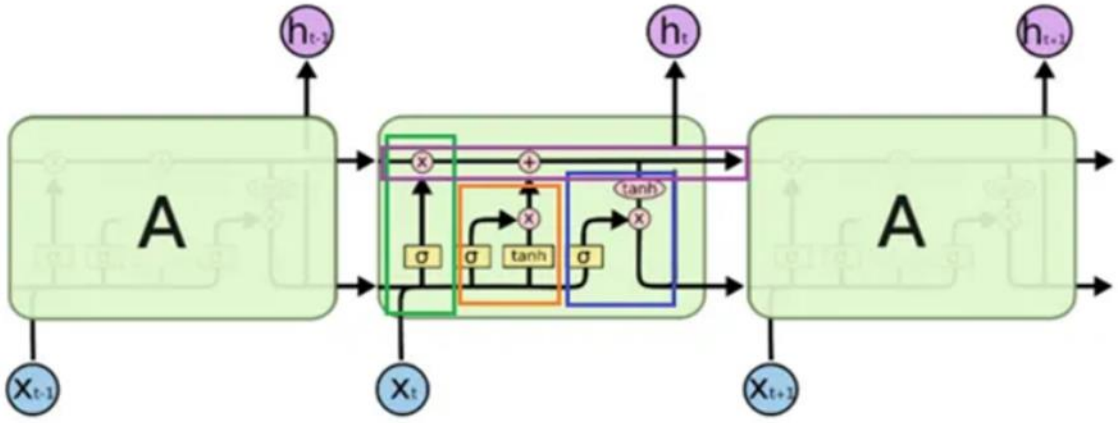
Özyinelemeli Sinir Ağı, veri işlerken, önceki veriyi de işleme dahil eder.
Sinir ağına “Merhaba” eklersin.
Sinir ağına “Dünya” eklersen kendisinden önceki “Merhaba” yı hatırlar ve onu da işleme katar.

Ama eksik yanı vardır. Çok fazla terimi / tokeni aklında tutamaz. Ve geriye yayılım sırasında gradyan bozulması / yok olması problemi de yaşar.

Bu eksikliği giderecek versiyonu olan LSTM (Uzun-Kısa Vadeli Bellek) piyasa dahil olur.

Cell State:
Genel hafızadır. “Hello” “World” gibi terimlerimizi tutmamızı sağlayan yapıdır.

Input Gate:



Forget Gate - Input Gate - Output Gate - Cell State

LSTM

Bulunan terimlere yeni terimler ekler. Ama eklemeden önce “terim değer testi” yapar.
Örnek:

“Yağmur” teriminin değeri 0,8 gibi yüksekse (bu değerler kütüphanede varmış) Cell State ‘ye ekler.
“ve de” gibi bağlaç bir terim ise değeri (muhtemelen) 0,1 gibi düşüktür. Bu durumda Cell State ‘ye eklenmez.

Forget Gate:
Input Gate ‘den terim eklenmesi sonucu Cell State ‘de değerini kaybedebilecek terimler olabilir. Bu durumda Forget Gate ile bu terimler elenir.
Örnek: Değeri 0,2 ‘den daha da düşmüş olan terimleri sil.

Output Gate:
Cell State ‘de belirli değerlerin üstüne çıkmış terimleri output olarak bir sonraki sinir ağına aktarır.
Örnek: Değeri 0,8 ‘in üstündeki terimleri çıktı olarak ver.

Uygulamanın İşleyiş Aşamaları

1. **Borsa Seçimi**: Kullanıcı, tahmin yapmak istediği borsa endeksini seçer. Örneğin, NASDAQ 100 veya BIST 100.
2. **Tarih Girişi**: Kullanıcı, tahmin için kullanılacak tarihi seçer.
3. **Model Yükleme**: Seçilen endeks için önceden eğitilmiş LSTM modeli yüklenir. **yfinance** kütüphanesi kullanılarak seçilen borsa için tarihsel fiyat verileri indirilir. Bu veriler "Kapanış Fiyatı" gibi alanları içerir ve veri analizi için uygun hale getirilir. Model, seçilen borsaya göre yüklenir ve tahmin işlemleri başlatılır.
4. **Veri İndirme ve Ölçeklendirme**:
    a. Yahoo Finance API kullanılarak geçmiş veriler indirilir.
    b. Veriler, MinMaxScaler ile normalize edilir.
5. **Tahmin İşlemi**:
    a. Geçmiş 60 günlük veriler modele giriş olarak verilir.
    b. Model, seçilen tarih itibarıyla 5 günlük fiyat tahmini üretir.
6. **Sonuçların Görselleştirilmesi**:
    Kullanıcı, tahmin sonuçlarının grafiksel bir görünümünü veya tablo halinde sunumunu görebilir. Şu görünümler sunulur:
    a. Tahmin Grafiği
    b. Son 10 Gün ve Tahmin
    c. Günlük Değişim ve Volatilite
    d. Hareketli Ortalamalar ve RSI
    e. Son 1 Ay ve Hareketli Ortalama
    f. Tahmin Tablosu

g. Tahmin Özeti

Model Eğitimindeki Kilit Noktalar (machineLearning.ipynb)

```
def create_dataset(dataset, time_step=60):
    X, y = [], []
    for i in range(len(dataset) - time_step):
        X.append(dataset[i:(i + time_step), 0])
        y.append(dataset[i + time_step, 0])
    return np.array(X), np.array(y)


time_step = 60
X_train, y_train = create_dataset(scaled_data, time_step)
X_train = X_train.reshape(X_train.shape[0], X_train.shape[1], 1)
```

Veri setlerindeki hedeflenmiş günün 60 gün öncesini girdi olarak ele alır. Çıktı olarak ise bir sonraki günü ele alır.

```
model = Sequential()
model.add(LSTM(50, return_sequences=True, input_shape=(X_train.shape[1], 1)))
model.add(LSTM(50, return_sequences=False))
model.add(Dense(25))
model.add(Dense(1))
model.compile(optimizer='adam', loss='mean_squared_error') #Adaptive Moment Estimation
model.fit(X_train, y_train, batch_size=64, epochs=50)
```

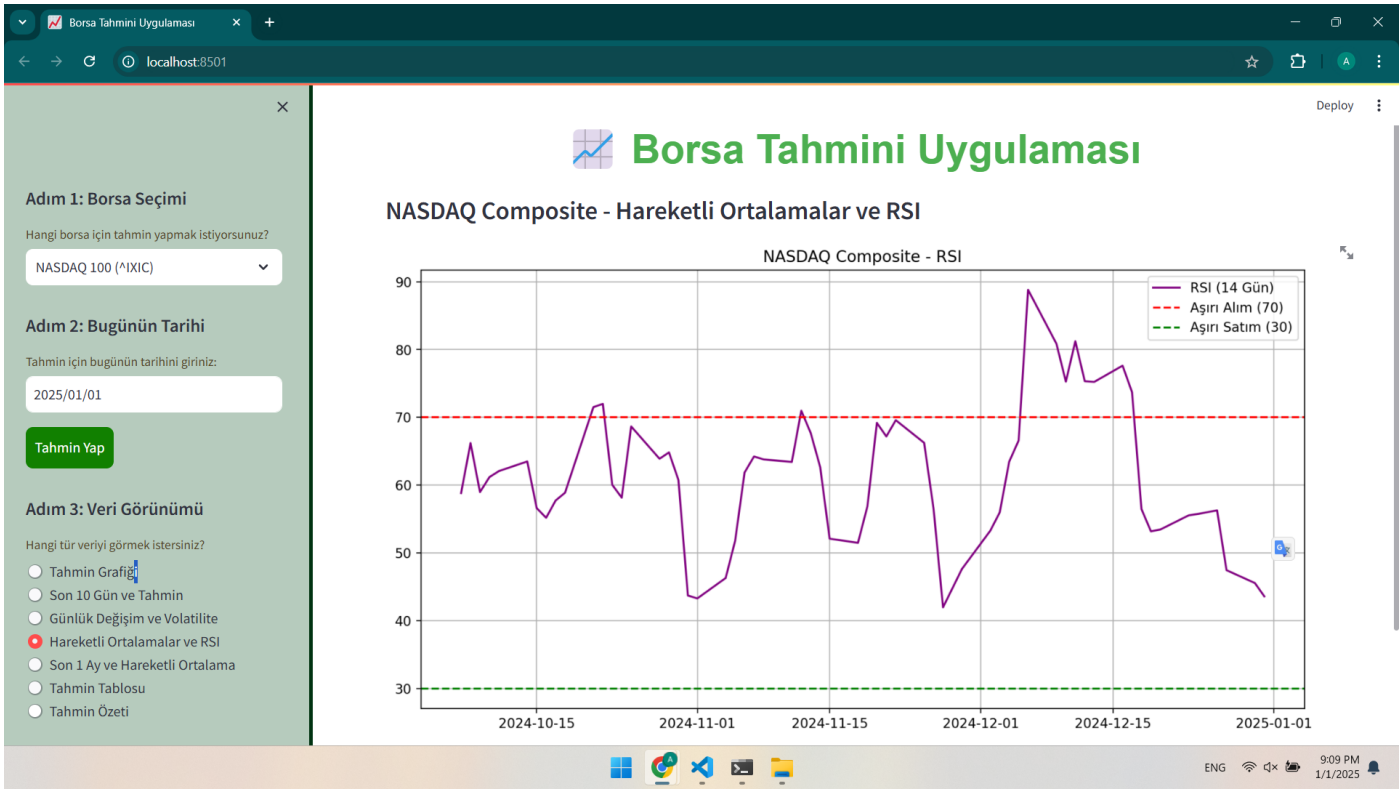
Sinir ağını oluşturduğumuz kısımdır. Sinir katmanları şu şekildedir:
(LSTM)50, (LSTM)50, 25 tam bağlı, 1 tam bağlı.

Uygulama Dosyasında Kilit Noktalar (app.py)

```
# CSS ile arayüz tasarımını iyileştirme
st.markdown("""
        <style>
        /* Sidebar arka planını değiştirmek */
        [data-testid="stSidebar"] {
            background-color: #B6CBBD !important;
            border-right: 4px solid #003311;
        }
        .stButton > button {
            background-color: #118000;
            color: white;
            border-radius: 8px;
            padding: 10px;
            border: none;
            font-size: 16px;
            font-weight: bold;
        }
        .stButton > button:hover {
            background-color: #0b5600;
            color: white;
        }
        /* Success mesajı düzenlemesi */
        [data-testid="stNotification"] {
            background-color: #84f594 !important; /* Tahmin sonrası sidebar ile uyumlu renk */
            border-left: 5px solid #118000;
            color: #003311 !important;
            font-weight: bold;
        }
        h1 {
            color: #4CAF50;
            text-align: center;
            font-family: 'Arial', sans-serif;
        }
        .stRadio label {
            font-size: 8px;
            font-weight: bold;
            color: #524528;
        }
        .stSelectbox label, .stDateInput label {
            font-weight: bold;
            color: #524528;
        }
        /* Üst margin (margin-top) kaldırma */
        .block-container {
            padding-top: 5px !important;
            margin-top: 5px !important
        }
    </style>

""", unsafe_allow_html=True)
```

Bu kısımda CSS özelliklerini kullanabilmemiz açısından streamlit'te markdown methoduyla HTML içinde "style" property'siyle CSS yazdığımız gibi UI için renk, şekillendirme, yazı boyutu vb. özellikleri ekledik.
Uygulamamızın arayüzü:



```python
# Adım 2: Bugünün Tarihi Girişi
st.sidebar.header("Adım 2: Bugünün Tarihi")
tarih_girisi = st.sidebar.date_input(
    "Tahmin için bugünün tarihini giriniz:",
    value=pd.Timestamp.now().date()
)

# Kullanıcı tarihi seçtiğinde end_date değişkenini güncelle ve değişiklik kontrolü yap
if "end_date" not in st.session_state or st.session_state.end_date != tarih_girisi:
    st.session_state.end_date = tarih_girisi
    st.session_state.data_loaded = False  # Veriler yeniden yüklenecek
```

Burada projeyi açtığımız anda varsayılan olarak bugünün tarihini getiriyor. Ve önümüzdeki beş günü tahmin ederken bu tarih referans alınıyor. Ama tarihi değiştirip geçmiş bir zaman için de bu tahmin işlemlerini gerçekleştirebiliyoruz.

```python
# Tahmin butonu
if st.sidebar.button("Tahmin Yap") or not st.session_state.data_loaded:
    try:
        # 1. Model Yükleme
        ticker = yf.Ticker(ticker_symbol)
        borsa_adi = ticker.info.get('shortName', borsa_secimi)
        st.session_state.borsa_adi = borsa_adi

        current_directory = os.getcwd()
        model_path = os.path.join(current_directory, ticker_symbol + ".h5")
        model = load_model(model_path)
        st.success("✅ Model başarıyla yüklendi!")

        # 2. Veri İndirme ve Ölçeklendirme
        data = yf.download(ticker_symbol, start="2010-01-01", end=str(st.session_state.end_date))
        st.session_state.data = data
        close_prices = data['Close'].values.reshape(-1, 1)
        scaler = MinMaxScaler(feature_range=(0, 1))
        scaled_data = scaler.fit_transform(close_prices)

        # 3. Gelecekteki Tahminler
        last_60_days = scaled_data[-60:]
        temp_input = list(last_60_days.flatten())
        n_days = 5
        predictions = []

        # Gelecekteki tahmin tarihleri, tarih_girisi baz alınarak başlatılıyor
        future_dates = [tarih_girisi + timedelta(days=i) for i in range(n_days)]

        for i in range(n_days):
            X_test = np.array(temp_input[-60:]).reshape(1, 60, 1) #(örnek sayısı, zaman adımı, özellik sayısı)
            predicted_value = model.predict(X_test, verbose=0)[0, 0] #verbose=0:Sessiz mod.Hiçbir bilgi yazdırılmaz.[0, 0] predict sonucutahminedilen ilk
            predictions.append(predicted_value)
            temp_input.append(predicted_value)
```

```python
        predictions = scaler.inverse_transform(np.array(predictions).reshape(-1, 1)) #Tahminleri eski hale geri döndürür
        # Tahmin sonuçlarını kaydet
        st.session_state.data_loaded = True
        st.session_state.close_prices = close_prices
        st.session_state.future_dates = future_dates
        st.session_state.predictions = predictions
    except Exception as e:
        st.error(f"Bir hata oluştu: {e}")
```

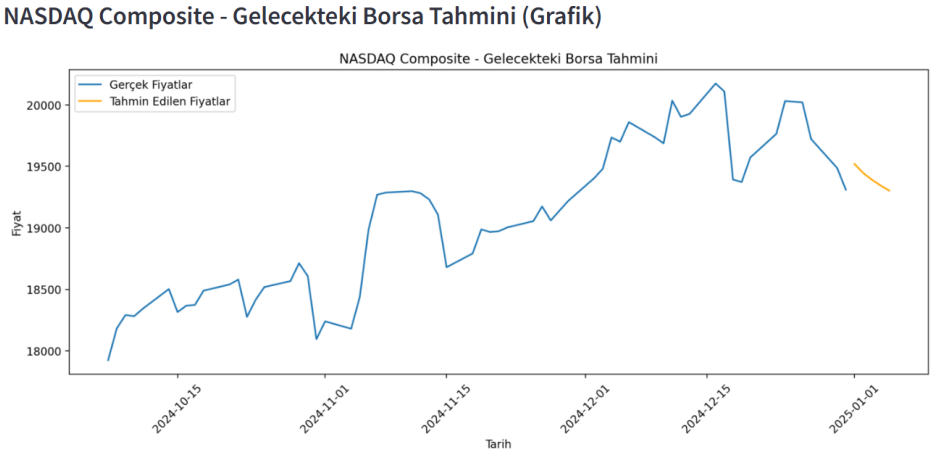
Burası tamamen projemizin makine öğrenmesi kısmında oluşturduğu .h5 uzantılı dosyalardaki modellemelerden faydalanarak gelecek günler için yapacağı tahmin işlemi öncesi sırayla “model yükleme, veri indirme ve ölçekleme, gelecekteki tahminleri oluşturmak için girilen tarihi referans alma” işlemlerini gerçekleştirip sonra tahmin işlemlerini tamamlıyor ve tahmin edilen değerleri “predictions” ismindeki listemize ekliyor. Artık elimizde üç önemli bileşen var: close_prices, future_dates_predictions.

```python
# Eğer tahmin yapıldıysa, veri türü seçimi göster
if st.session_state.data_loaded:
    # Adım 3: Veri Görünümü
    st.sidebar.header("Adım 3: Veri Görünümü")
    options = [
        "Tahmin Grafiği",
        "Son 10 Gün ve Tahmin",
        "Günlük Değişim ve Volatilite",
        "Hareketli Ortalamalar ve RSI",
        "Son 1 Ay ve Hareketli Ortalama",
        "Tahmin Tablosu",
        "Tahmin Özeti"
    ]
    st.session_state.current_view = st.sidebar.radio("Hangi tür veriyi görmek istersiniz?", options)
```

Burada yaptığımız tüm işlemlerden sonra sidebar’da hangi veri veya grafiği getirmek istediğimizi seçiyoruz.
Aşağıda 7 farklı veri veya grafiği oluşturma kodlarını açıklıyoruz:

```python
# Görünümler
if st.session_state.current_view == "Tahmin Grafiği":
    st.subheader(f"{st.session_state.borsa_adi} - Gelecekteki Borsa Tahmini (Grafik)")
    fig, ax = plt.subplots(figsize=(14, 5))
    ax.plot(st.session_state.data.index[-60:], st.session_state.close_prices[-60:], label="Gerçek Fiyatlar")
    ax.plot(st.session_state.future_dates, st.session_state.predictions, label="Tahmin Edilen Fiyatlar", color='orange')
    ax.set_title(f"{st.session_state.borsa_adi} - Gelecekteki Borsa Tahmini")
    ax.set_xlabel("Tarih")
    ax.set_ylabel("Fiyat")
    plt.xticks(rotation=45)
    ax.legend()     # Grafikteki çizgiler için bir açıklama (legend) ekler.
    st.pyplot(fig)  # Streamlit aracılığıyla grafiği kullanıcıya gösterir.
```

Bu kod bloğu, kullanıcının Tahmin Grafiği verisini çağırmak istediğinde yaptığı işlemler içindir. Aşağıdaki gibi bir grafik oluşturmaya yarar:



```python
elif st.session_state.current_view == "Tahmin Tablosu":
    st.subheader(f"{st.session_state.borsa_adi} - Tahmin Sonuçları (Tablo)")
    try:
        prediction_table = pd.DataFrame({
            'Tarih': [date.strftime('%Y-%m-%d') for date in st.session_state.future_dates],
            'Tahmin Edilen Fiyat': st.session_state.predictions.flatten() # Tahmin edilen fiyatları tek boyutlu
        })
        st.dataframe(prediction_table)
    except Exception as e:
        st.error(f"Tablo gösteriminde bir hata oluştu: {e}")
```

Bu kod bloğu, kullanıcının Tahmin Tablosu verisini çağırmak istediğinde yaptığı işlemler içindir. Aşağıdaki gibi bir tablo oluşturmaya yarar:

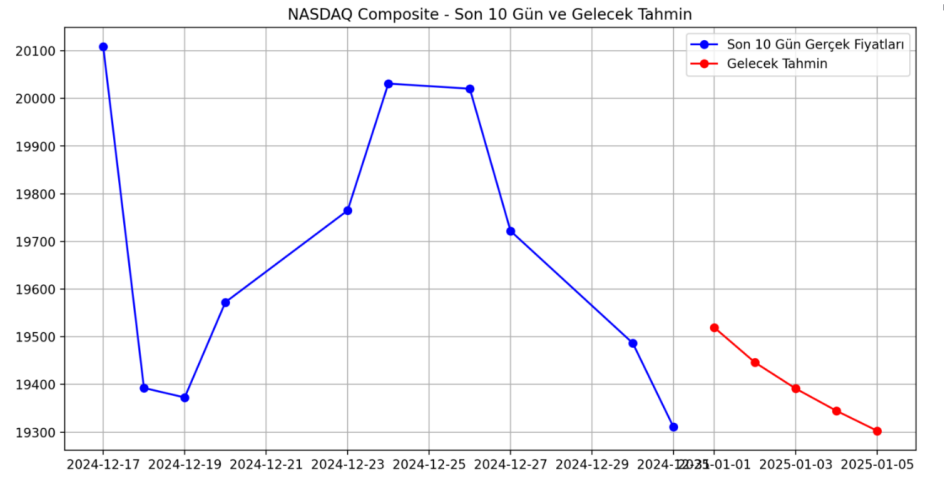
NASDAQ Composite - Tahmin Sonuçları (Tablo)

	Tarih	Tahmin Edilen Fiyat
0	2025-01-01	19,519.5859
1	2025-01-02	19,446.1875
2	2025-01-03	19,391.4219
3	2025-01-04	19,344.8281
4	2025-01-05	19,302.8984

```
elif st.session_state.current_view == "Son 10 Gün ve Tahmin":
    st.subheader(f"{st.session_state.borsa_adi} - Son 10 Gün ve Gelecek Tahmin")
    try:
        filtered_data = st.session_state.data.loc[:str(st.session_state.end_date)]
        real_last_10_days = filtered_data['Close'][-10:].values
        fig, ax = plt.subplots(figsize=(12, 6))
        ax.plot(filtered_data.index[-10:], real_last_10_days, label="Son 10 Gün Gerçek Fiyatları", marker='o', color='blue')
        ax.plot(st.session_state.future_dates, st.session_state.predictions.flatten(), label="Gelecek Tahmin", marker='o', color='red')
        ax.set_title(f"{st.session_state.borsa_adi} - Son 10 Gün ve Gelecek Tahmin")
        ax.legend()
        ax.grid()
        st.pyplot(fig)
    except Exception as e:
        st.error(f"Görselleştirme sırasında bir hata oluştu: {e}")
```

Bu kod bloğu, kullanıcının son 10 gün ve tahminin verisini çağırmak istediğinde yaptığı işlemler içindir. Aşağıdaki gibi bir grafik oluşturmaya yarar:

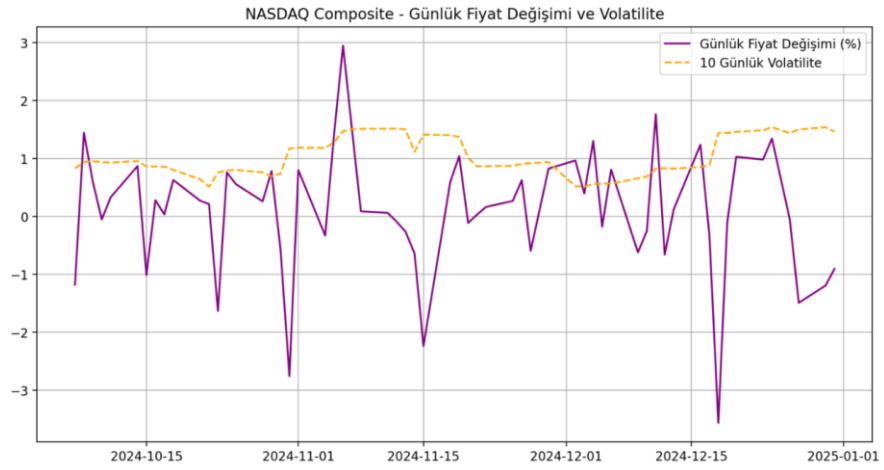
NASDAQ Composite - Son 10 Gün ve Gelecek Tahmin



```python
elif st.session_state.current_view == "Günlük Değişim ve Volatilite":
    st.subheader(f"{st.session_state.borsa_adi} - Günlük Değişim ve Volatilite")
    try:
        st.session_state.data['Daily Change (%)'] = st.session_state.data['Close'].pct_change() * 100
        st.session_state.data['Volatility'] = st.session_state.data['Daily Change (%)'].rolling(window=10).std()  # 10 günlük hareketli standart sapma (vola
        fig, ax = plt.subplots(figsize=(12, 6))
        ax.plot(st.session_state.data.index[-60:], st.session_state.data['Daily Change (%)'][-60:], label="Günlük Fiyat Değişimi (%)", color='purple')
        ax.plot(st.session_state.data.index[-60:], st.session_state.data['Volatility'][-60:], label="10 Günlük Volatilite", color='orange', linestyle='--')
        ax.set_title(f"{st.session_state.borsa_adi} - Günlük Fiyat Değişimi ve Volatilite")
        ax.legend()
        ax.grid()
        st.pyplot(fig)
    except Exception as e:
        st.error(f"Veri analizi sırasında bir hata oluştu: {e}")
```

Bu kod bloğu, kullanıcı son altmış günün günlük değişim ve volatilitesini çağırmak istediğinde yaptığı işlemler içindir. Aşağıdaki gibi bir grafik oluşturmaya yarar:

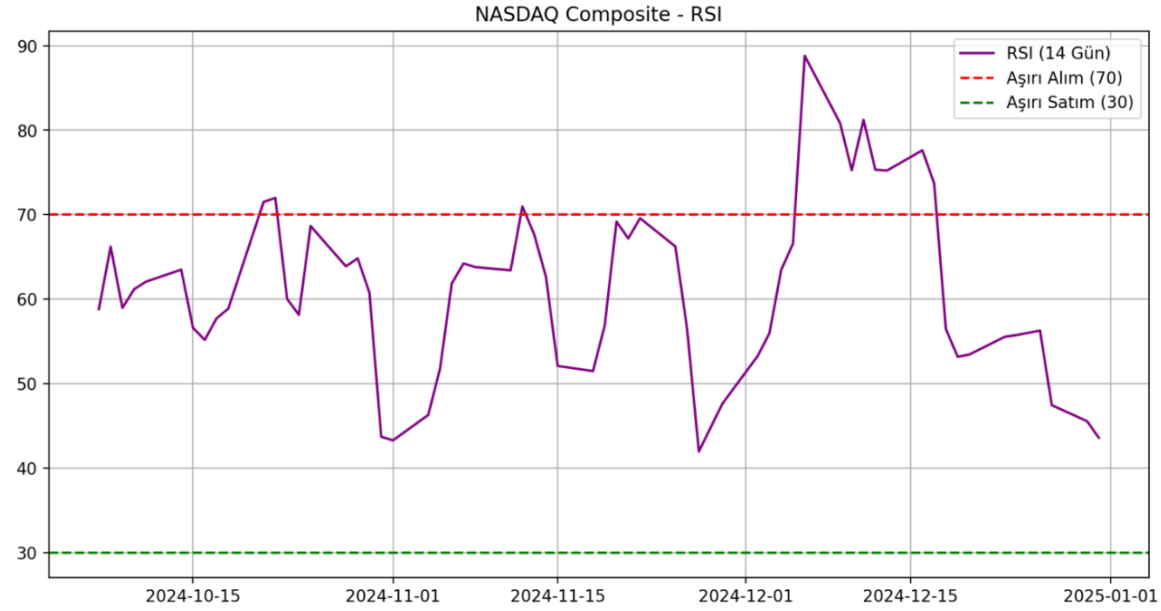
NASDAQ Composite - Günlük Değişim ve Volatilite



```python
elif st.session_state.current_view == "Hareketli Ortalamalar ve RSI":
    st.subheader(f"{st.session_state.borsa_adi} - Hareketli Ortalamalar ve RSI")
    try:
        data = st.session_state.data
        delta = data['Close'].diff(1)  #Günlük kapanış fiyatlarındaki değişiklikleri hesaplar (bugünkü fiyat - dünkü fiyat)
        gain = delta.where(delta > 0, 0)  #Pozitif değişiklikleri alır (kazanç)
        loss = -delta.where(delta < 0, 0)  #Negatif değişiklikleri alır (kayıp), diğer değerleri 0 olarak belirler.
        avg_gain = gain.rolling(window=14).mean()
        avg_loss = loss.rolling(window=14).mean()
        rs = avg_gain / avg_loss #         # RSI için gerekli olan göreli güç oranını (Relative Strength) hesaplar.
        data['RSI'] = 100 - (100 / (1 + rs)) #RSI (Relative Strength Index) değerini hesaplar ve bir sütuna kaydeder
        fig, ax = plt.subplots(figsize=(12, 6))
        ax.plot(data.index[-60:], data['RSI'][-60:], label="RSI (14 Gün)", color='purple')
        ax.axhline(70, color='red', linestyle='--', label="Aşırı Alım (70)")
        ax.axhline(30, color='green', linestyle='--', label="Aşırı Satım (30)")
        ax.set_title(f"{st.session_state.borsa_adi} - RSI")
        ax.legend()
        ax.grid()
        st.pyplot(fig)
    except Exception as e:
        st.error(f"RSI hesaplaması sırasında bir hata oluştu: {e}")
```

Bu kod bloğu, kullanıcı son altmış günün hareketli ortalamalar ve RSI değerini çağırmak istediğinde yaptığı işlemler içindir. Aşırı alım, satım durumlarını gösterir. Aşağıdaki gibi bir grafik oluşturmaya yarar:

NASDAQ Composite - Hareketli Ortalamalar ve RSI

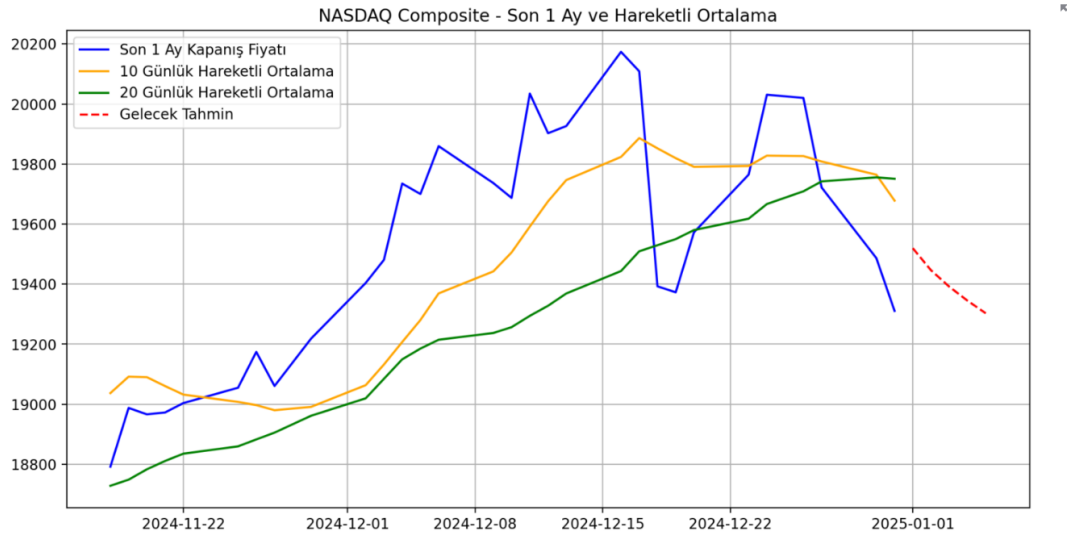


```
elif st.session_state.current_view == "Son 1 Ay ve Hareketli Ortalama":
    st.subheader(f"{st.session_state.borsa_adi} - Son 1 Ay ve Hareketli Ortalama")
    try:
        data = st.session_state.data
        data['SMA_10'] = data['Close'].rolling(window=10).mean()  # 10 günlük basit hareketli ortalamayı hesaplar ve yeni bir sütun olarak e
        data['SMA_20'] = data['Close'].rolling(window=20).mean()

        fig, ax = plt.subplots(figsize=(12, 6))
        ax.plot(data.index[-30:], data['Close'][-30:], label="Son 1 Ay Kapanış Fiyatı", color='blue')
        ax.plot(data.index[-30:], data['SMA_10'][-30:], label="10 Günlük Hareketli Ortalama", color='orange')
        ax.plot(data.index[-30:], data['SMA_20'][-30:], label="20 Günlük Hareketli Ortalama", color='green')
        ax.plot(st.session_state.future_dates, st.session_state.predictions.flatten(), label="Gelecek Tahmin", color='red', linestyle='--')
        ax.set_title(f"{st.session_state.borsa_adi} - Son 1 Ay ve Hareketli Ortalama")
        ax.legend()
        ax.grid()
        st.pyplot(fig)
    except Exception as e:
        st.error(f"Hareketli ortalama hesaplamasında bir hata oluştu: {e}")
```

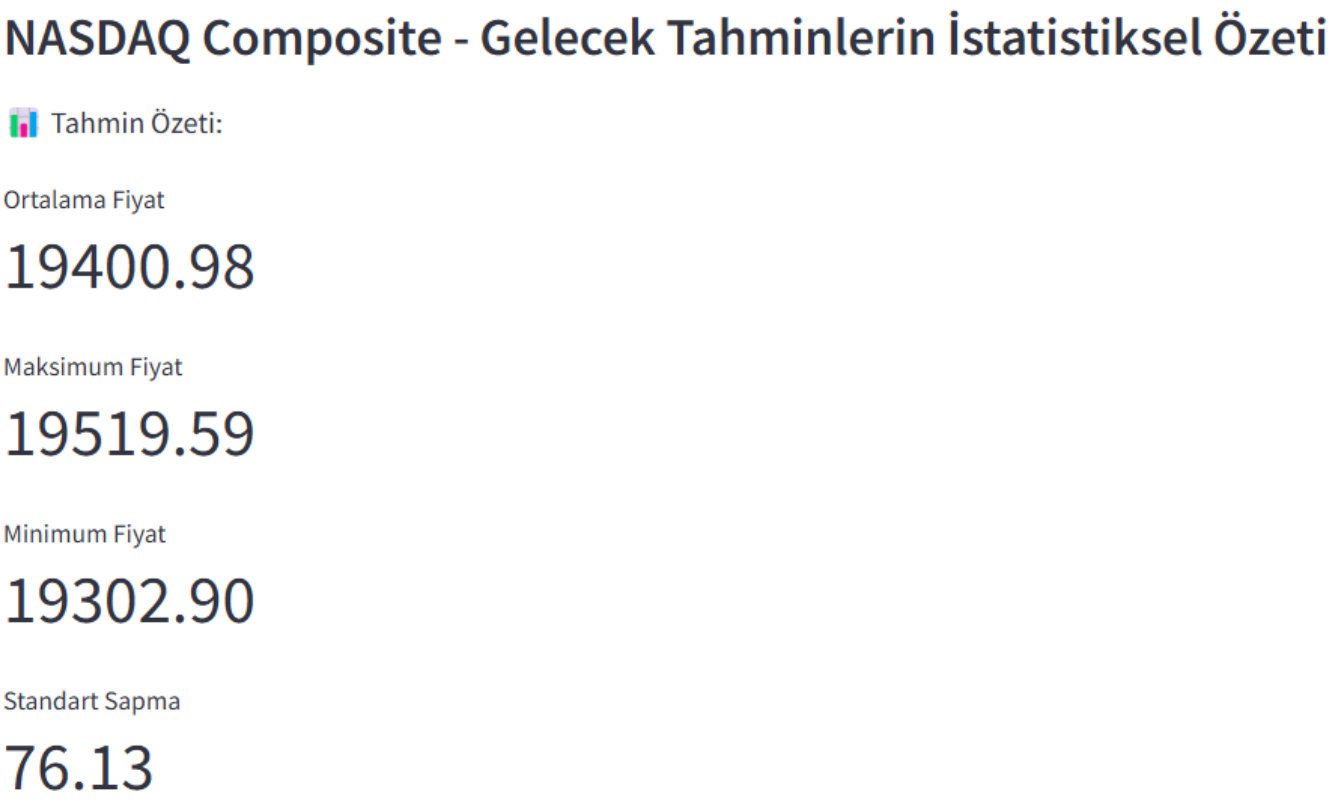
Bu kod bloğu, kullanıcı son bir ay ve hareketli ortalamalar değerini çağırmak istediğinde yaptığı işlemler içindir. 10 günlük ve 20 günlük hareketli ortalamaların verileriyle gelecek tahminin yönelimin durumlarını gösterir. Aşağıdaki gibi bir grafik oluşturmaya yarar:

NASDAQ Composite - Son 1 Ay ve Hareketli Ortalama



```
elif st.session_state.current_view == "Tahmin Özeti":
    st.subheader(f"{st.session_state.borsa_adi} - Gelecek Tahminlerin İstatistiksel Özeti")
    try:
        predictions = st.session_state.predictions.flatten()
        st.write("📊 Tahmin Özeti:")
        st.metric("Ortalama Fiyat", f"{np.mean(predictions):.2f}")
        st.metric("Maksimum Fiyat", f"{np.max(predictions):.2f}")
        st.metric("Minimum Fiyat", f"{np.min(predictions):.2f}")
        st.metric("Standart Sapma", f"{np.std(predictions):.2f}")
    except Exception as e:
        st.error(f"Tahmin özetinde bir hata oluştu: {e}")
```

Bu kod bloğu beş günlük tahmin değerleriyle ilgili, model eğitimimiz sonucu tahminlerdeki en yüksek, en düşük, ortalama değer ve standart sapma bilgilerini çağırmak istediğinde yaptığı işlemler içindir. Çıktısı şu şekildedir:



Projeyi Yeniden Geliştirme Rehberi

Projeyi sıfırdan başlatmak isteyen bir geliştirici için adımlar:

1. **Ortam Kurulumu**: Gerekli kütüphaneleri aşağıdaki komutla yükleyin:

```
pip install -r requirements.txt
```

2. **Model Eğitimi**: Tarihsel fiyat verilerini indirerek bir LSTM modeli eğitin ve modeli `.h5` formatında kaydedin.
3. **Uygulama Çalıştırma**: Uygulamayı başlatmak için aşağıdaki komutu kullanın:

```
streamlit run app.py
```

4. **Görsel ve Veri Sunumu**: Grafiklerin ve tabloların doğru çalıştığından emin olun.