Kullanıcı Arayüzü Oluşturma:

- Kod, Streamlit kütüphanesini kullanarak bir web tabanlı kullanıcı arayüzü oluşturur. Bu arayüz, kullanıcıdan bir hisse senedi simgesi girmesini ve çeşitli analizleri ve tahminleri gerçekleştirmesini sağlar.

Hisse Senedi Verilerinin Alınması:

- Kullanıcıdan alınan hisse senedi simgesi, Yahoo Finance API'sini kullanarak ilgili hisse senedinin günlük fiyat verilerini getirir.
- Bu veriler, 'open' (açılış), 'close' (kapanış), 'volume' (hacim), 'high' (en yüksek) ve 'low' (en düşük) fiyatları içerir.

Teknik Göstergelerin Hesaplanması:

- Kod, hisse senedinin fiyat hareketlerini analiz etmek için çeşitli teknik göstergeleri hesaplar.
- Kullanılan göstergeler şunlardır:
    - RSI (Göreceli Güç Endeksi): Aşırı alım ve aşırı satım bölgelerini gösteren bir gösterge.
    - Bollinger Bantları: Fiyat oynaklığını ve destek/direnç seviyelerini gösteren bir gösterge.
    - OBV (On-Balance Volume): Ticaret hacmini izleyerek bir hisse senedinin trendini değerlendiren bir gösterge.
    - MACD (Hareketli Ortalama Yakınsama Farklılığı): Trendi ve momentumu gösteren bir gösterge.
    - Momentum: Hız ve ivme ölçümleri ile hisse senedinin gücünü gösteren bir gösterge.

  Her gösterge için ilgili verileri hesaplar ve görselleştirir.

Gelecek Fiyat Tahminleri:

- Kullanıcıya hisse senedinin gelecekteki fiyatlarını tahmin etme olanağı sunar.
- Bu tahmin, LSTM algoritması kullanılarak son fiyat verileri üzerinden yapılır. Kullanıcı, tahmin edilecek gün sayısını belirleyebilir.

Rapor Oluşturma ve İndirme:

- Kullanıcı, tüm analizleri ve tahminleri içeren bir PDF rapor oluşturabilir.
- Oluşturulan raporu indirebilir ve kaydedebilir. Raporda, hisse senedinin genel analizi, kullanılan göstergelerin açıklamaları ve son tahminler yer alır.

Bu kod örneği, finansal analiz yapmak ve bir hisse senedinin fiyat hareketlerini anlamak isteyen kullanıcılar için kullanışlı bir araç sağlar. Ayrıca, kullanıcı dostu bir arayüz sunar ve görselleştirmeler kullanarak karmaşık analizleri basitleştirir. Özellikle hisse senedi yatırımı yapmayı planlayan veya ilgilenen kişiler için değerli bir araç olabilir.

```python
from tensorflow import keras

model_rnn = keras.Sequential([
    keras.layers.Bidirectional(keras.layers.LSTM(units=75,
return_sequences=True, input_shape=(x_train.shape[1], 1))),
    keras.layers.Bidirectional(keras.layers.LSTM(units=64)),
    keras.layers.Dense(300),
    keras.layers.Dense(32),
    keras.layers.Dense(1),
])



# Erken durdurma geri çağrısı
early_stopping = tf.keras.callbacks.EarlyStopping(
    monitor="loss",    # İzlenen metrik (loss)
    min_delta=0.0008,  # Minimum değişim
    patience=3,        # Sabır seviyesi (kaç epoch sabretmeli)
)




# Modeli derle
model_rnn.compile(optimizer="adam", loss="mse")



# Modeli eğit
history = model_rnn.fit(
    x_train,
    y_train,
    epochs=20,
    batch_size=5,
    callbacks=[early_stopping]  # Erken durdurma geri çağrısı ekleDİM
)
```

```
## Transformer Yaklaşımı

Küçük zaman dilimlerinde gerçekleşen fiyat hareketlerine dikkat ederek
oynaklık sorununu çözmeye çalıştım


def transformer_encoder(inputs, head_size, num_heads, ff_dim, dropout=0):
    x = keras.layers.LayerNormalization(epsilon=1e-6)(inputs)
    x = keras.layers.MultiHeadAttention(
        key_dim=head_size, num_heads=num_heads, dropout=dropout
    )(x, x)
    x = keras.layers.Dropout(dropout)(x)
    res = x + inputs
    x = keras.layers.LayerNormalization(epsilon=1e-6)(res)
    x = keras.layers.Conv1D(filters=ff_dim, kernel_size=1,
activation="relu")(x)
    x = keras.layers.Dropout(dropout)(x)
    x = keras.layers.Conv1D(filters=inputs.shape[-1], kernel_size=1)(x)
    return x + res




def build_model(
    input_shape,
    head_size,
    num_heads,
    ff_dim,
    num_transformer_blocks,
    mlp_units,
    dropout=0,
    mlp_dropout=0,
):
    inputs = keras.Input(shape=input_shape)
    x = inputs

    for _ in range(num_transformer_blocks):
        x = transformer_encoder(x, head_size, num_heads, ff_dim, dropout)

    x = keras.layers.GlobalAveragePooling1D(data_format="channels_first")(x)
    for dim in mlp_units:
        x = keras.layers.Dense(dim, activation="elu")(x)
        x = keras.layers.Dropout(mlp_dropout)(x)
    outputs = keras.layers.Dense(1, activation="linear")(x)
    return keras.Model(inputs, outputs)
```

```python
# Giriş verisinin şekli
input_shape = x_train.shape[1:]


# Geri çağrılar
callbacks = [keras.callbacks.EarlyStopping(patience=3)]


# Model
model = build_model(
    input_shape,
    head_size=46,
    num_heads=60,
    ff_dim=55,
    num_transformer_blocks=7,
    mlp_units=[256],
    mlp_dropout=0.4,
    dropout=0,
)


model.compile(
    loss="mean_squared_error",
    optimizer=keras.optimizers.Adam(learning_rate=1e-3),
    metrics=["mean_squared_error"],
)


history = model.fit(
    x_train,
    y_train,
    validation_split=0.01,
    epochs=10,
    batch_size=32,
    callbacks=callbacks,
)
```

LSTM Modeli:

- Model ve Eğitim: İlk kod örneği, LSTM tabanlı bir sinir ağı oluşturuyor ve zaman serisi verilerinin tahmininde kullanılıyor. LSTM'ler, özellikle sıralı verilerle iyi çalışan bir türdür ve zaman serileri için kullanışlıdır.
- Erken Durdurma: Erken durdurma geri çağrısı, aşırı uydurmayı önlemek ve eğitim sürecini optimize etmek için kullanılır.

Bu model, belirli bir hisse senedinin fiyat hareketlerini tahmin etmek için uygun bir seçenektir, çünkü zaman serileri genellikle sıralı verilerdir ve geçmiş verilere dayalı gelecekteki hareketleri tahmin etmek için LSTM gibi algoritmalar kullanmak mantıklıdır.

Dikkat Mekanizmalı Model:

- Model ve Eğitim: İkinci kod örneği, dikkat mekanizması içeren bir metin sınıflandırma modeli oluşturuyor. Dikkat mekanizması, metinlerdeki önemli özelliklere odaklanmaya yardımcı olur.
- Erken Durdurma: Erken durdurma, modelin aşırı uydurmayı önlemesine yardımcı olabilir.

Bu model, metin verileri gibi yapısız verileri analiz etmek ve metin verileri ile ilgili görevleri gerçekleştirmek için uygun bir seçenektir. Dikkat mekanizmaları, metinlerdeki farklı özelliklere odaklanma yeteneği sayesinde metin sınıflandırma görevlerinde yaygın olarak kullanılır.

LSTM ve dikkat mekanizması gibi daha karmaşık modeller, verilerin iyi yakalayabilir. Bu nedenle, verilerin doğası ve karmaşıklığı, daha gelişmiş modellerin kullanılmasını gerektirebilir.

Hangi modelin kullanılacağı, verilerin türü ve tahmin ihtiyaçlarına bağlıdır. Daha karmaşık veriler ve tahminler için LSTM veya dikkat mekanizmalı modeller daha uygun olabilir.