**Makine ve Derin Öğrenmede Veri Normalizasyonu ve Standardizasyonu**

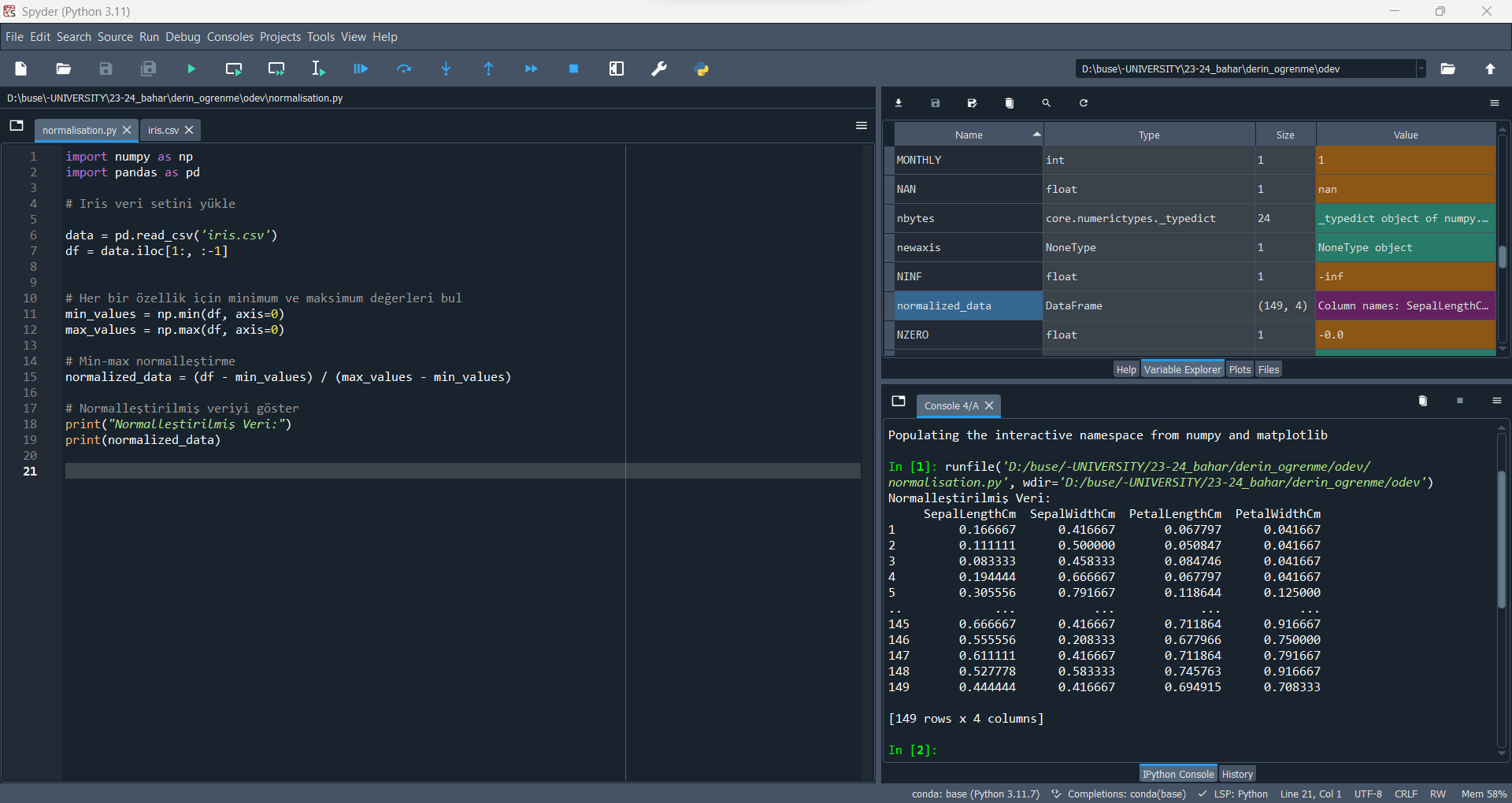
**Normalizasyon**

Verilerin 0 ve 1 arasında ölçeklendirilmesi işlemidir. En sık kullanılan yöntem aşağıdaki formül ile bulunur;



**Formülün Koda Aktarımı**

Gerekli olan kütüphanelerin eklenmesi;



Veri setinin okunması için pandas kütüphanesinin read\_csv özelliğini kullanmaktayız.

metin, ekran görüntüsü, yazılım, multimedya yazılımı içeren bir resim

Açıklama otomatik olarak oluşturuldu Verilerin içerisindeki özellikleri gösteren ilk 4 sütunu almak için pandas kütüphanesinin iloc özelliğini kullanmaktayız.

Formüldeki minimum ve maksimum olan değerleri bulmak için numpy kütüphanesinin min ve max özelliğini kullanmaktayız.

metin, ekran görüntüsü, yazılım, multimedya yazılımı içeren bir resim

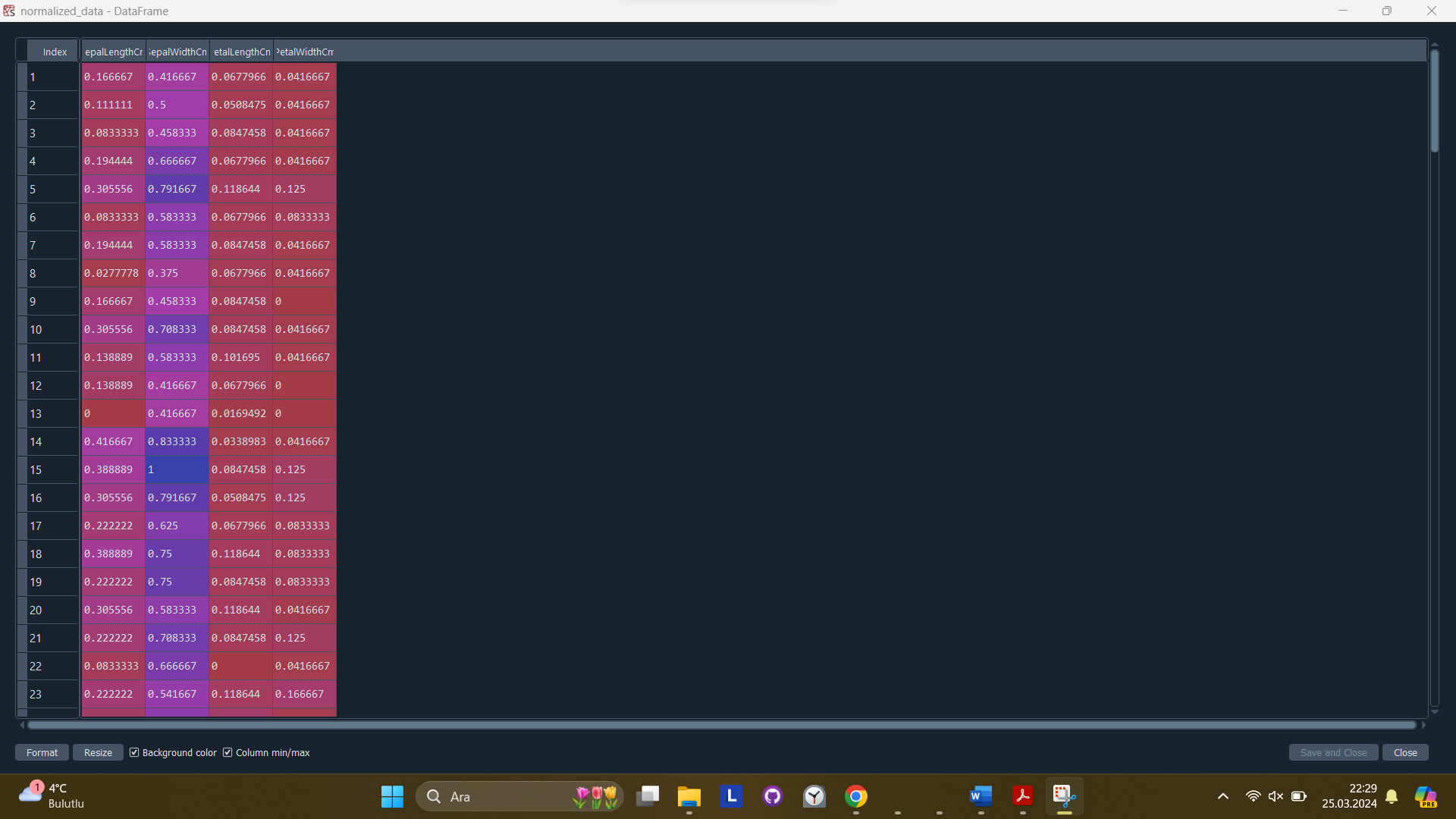
Açıklama otomatik olarak oluşturulduAxis = 0 diyerek her sütundaki değeri saptamaktayız.

Formülü yazıya dökerek normalized\_data değişkenine atıyoruz.

metin, ekran görüntüsü, yazılım, multimedya yazılımı içeren bir resim

Açıklama otomatik olarak oluşturuldu

**Sonuç**:

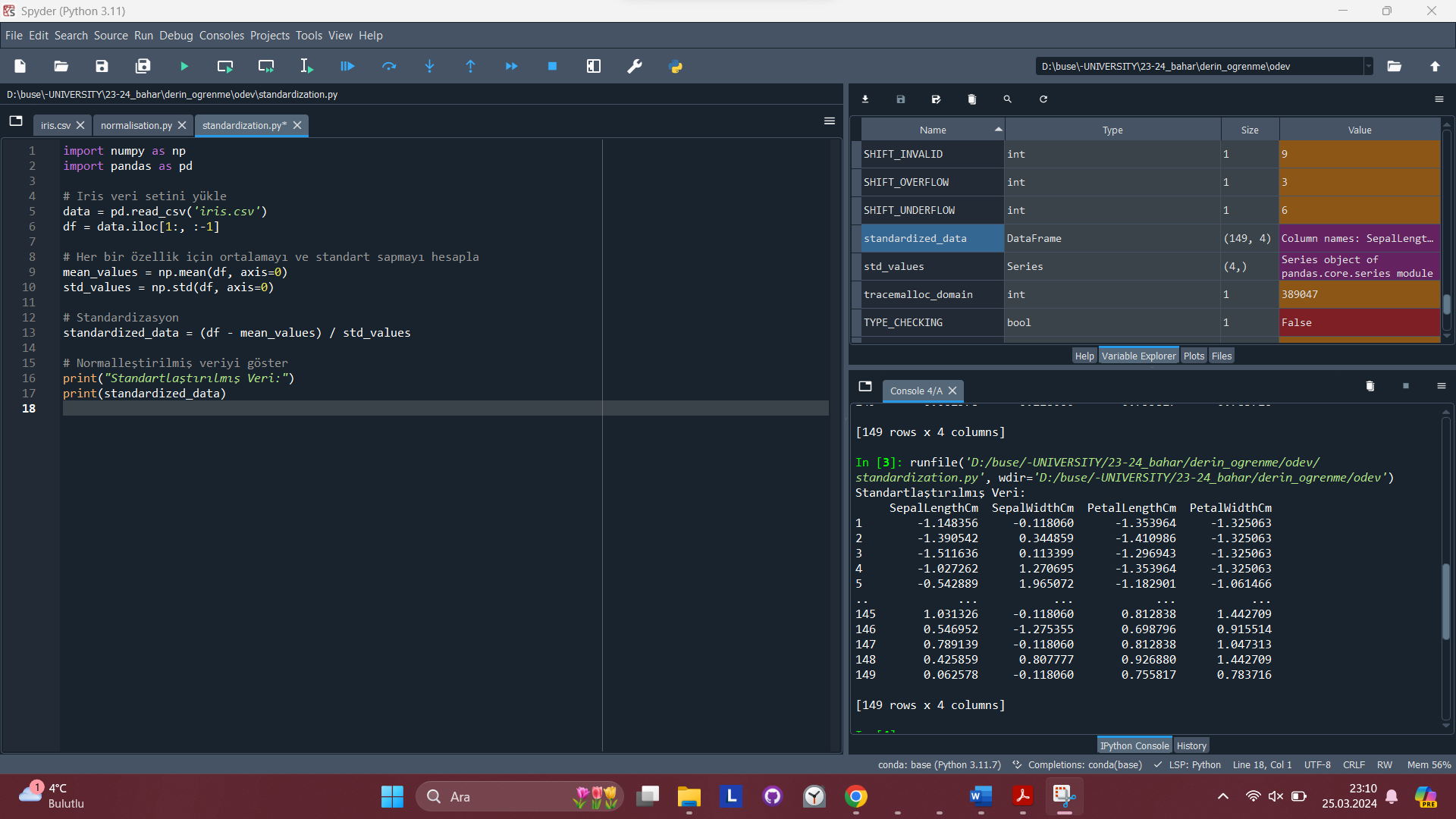


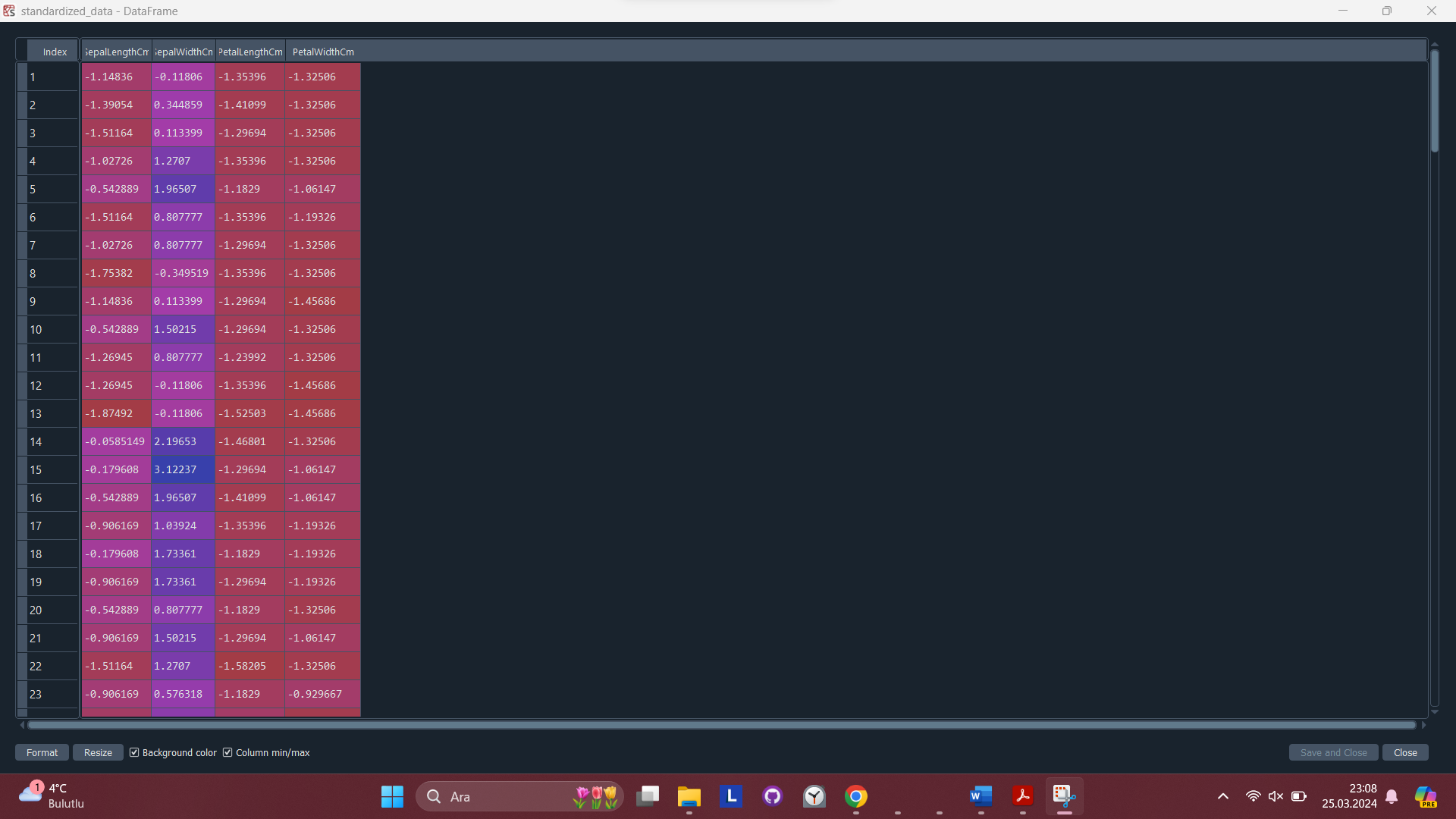
**Standardizasyon**

Veriyi, 0 ortalama ve 1 standart sapmaya sahip olması için yeniden ölçeklendirmedir. En sık kullanılan yöntem aşağıdaki formül ile bulunur;



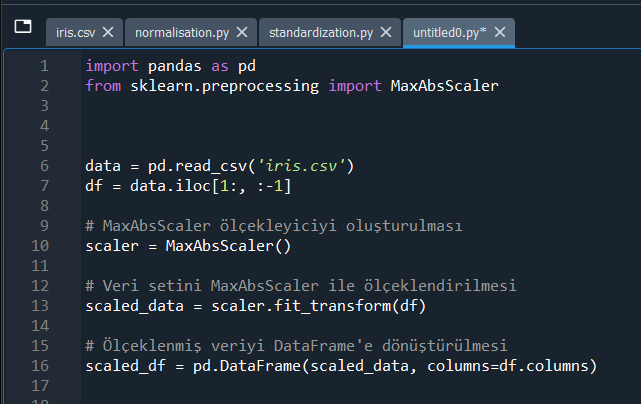
**Formülün Koda Aktarımı**

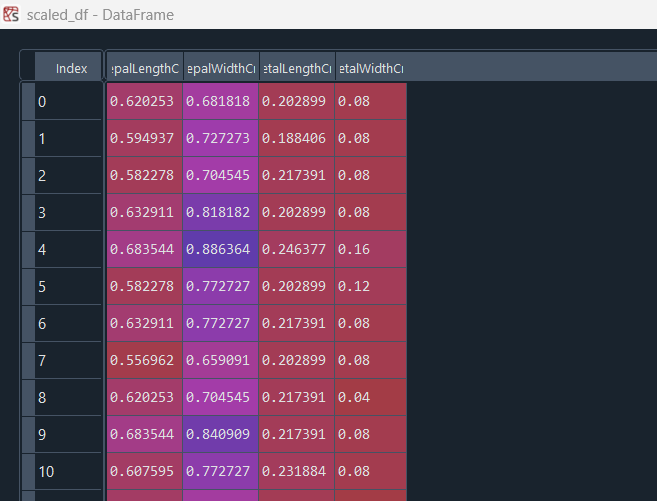


**Sonuç:**

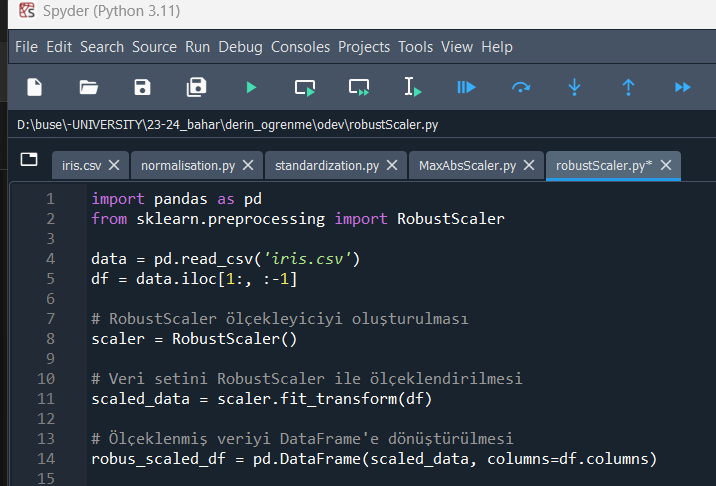
**MaxAbsScaler:** Verileri her bir özelliğin maksimum mutlak değerine göre ölçekler.

**Kod Üzerinde Gösterimi:**

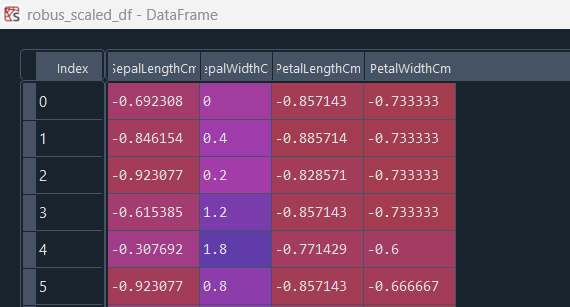
****

**Sonuç:**

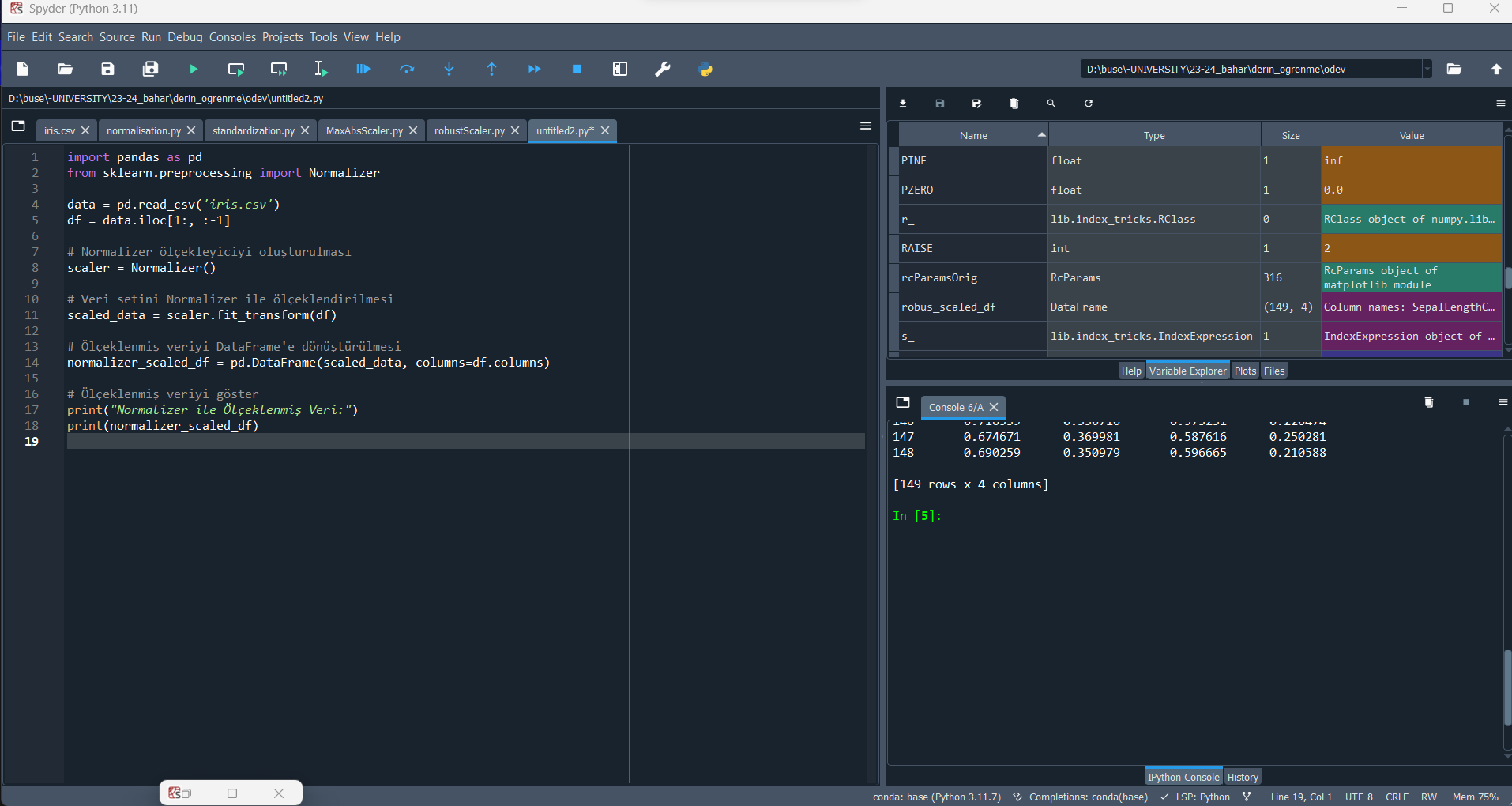
**RobustScaler:** Verilerin medyanı ve IQR (Interquartile Range) kullanılarak ölçeklenir.

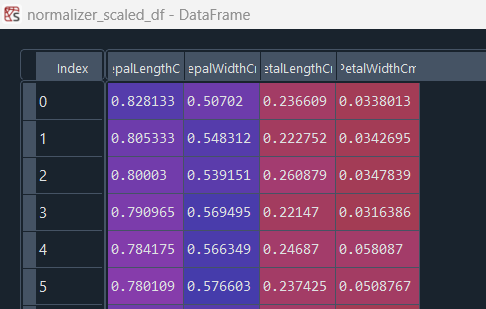
**Kod Üzerinde Gösterimi:**

**Sonuç:**

****

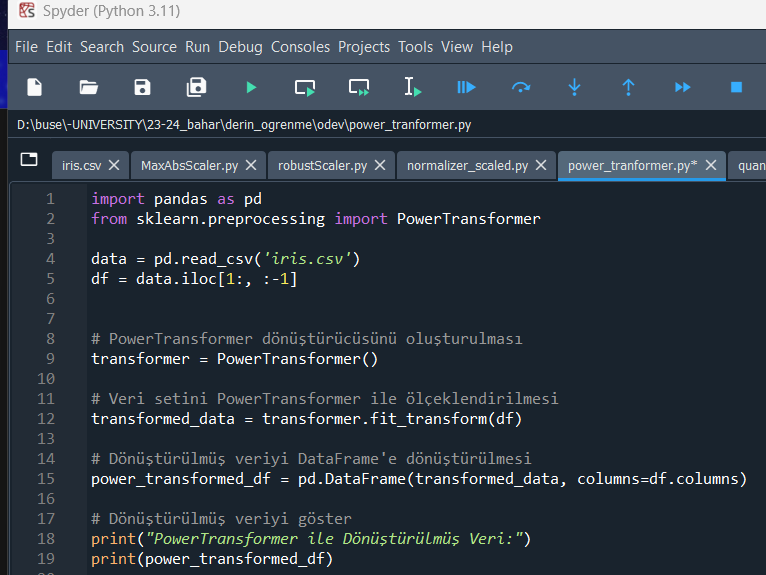
**Normalizer**: Her bir örnek veya gözlem için ölçekleme yapar, her bir satırı normlara göre dönüştürür.

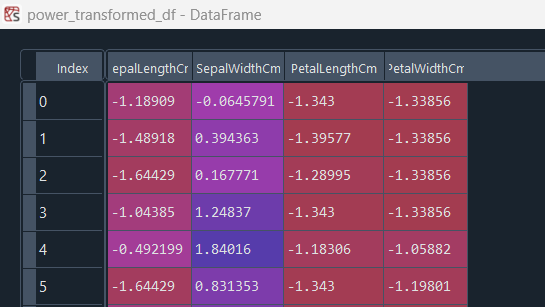
**Kod Üzerinde Gösterimi:**

**Sonuç:**

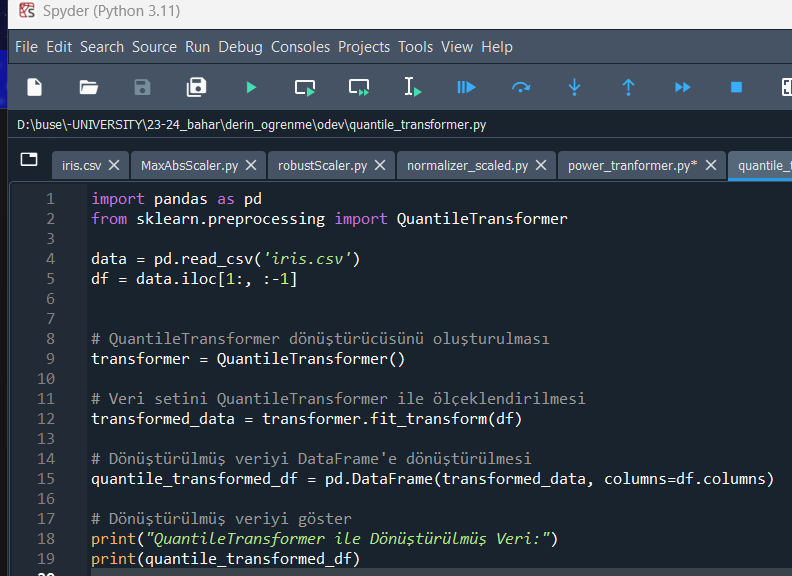
**PowerTransformer**: PowerTransformer, verileri bir güç dönüşümüne tabi tutarak normal dağılıma yakın hale getirir.

**Kod Üzerinde Gösterimi:**

****

**Sonuç:**

**QuantileTransformer:** Verileri bir veri kümesinin belirli bir yüzdelik diliminde yer alacak şekilde dönüştürür.

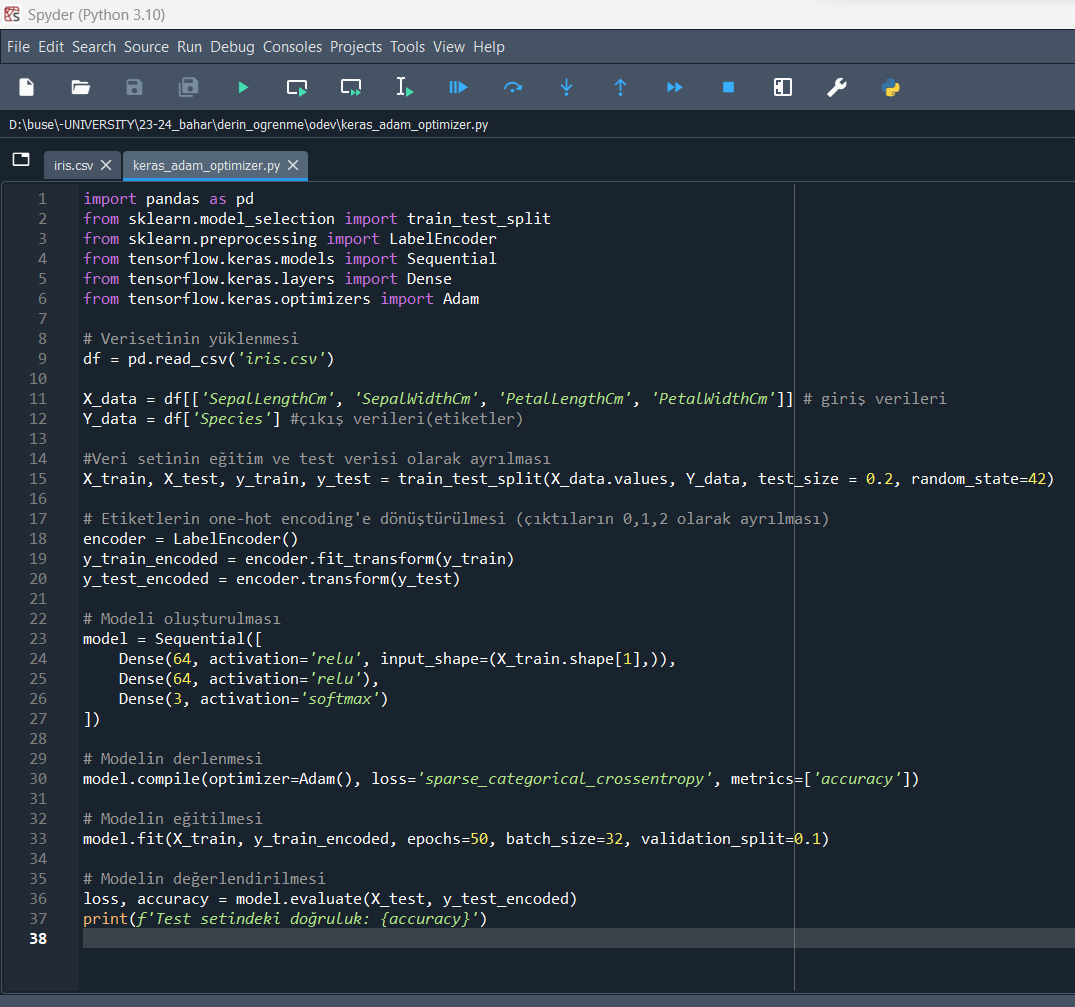
**Kod Üzerinde Gösterimi:**

**metin, ekran görüntüsü, yazılım, multimedya içeren bir resim

Açıklama otomatik olarak oluşturulduSonuç:**

**Keras Optimizer Uygulamaları**

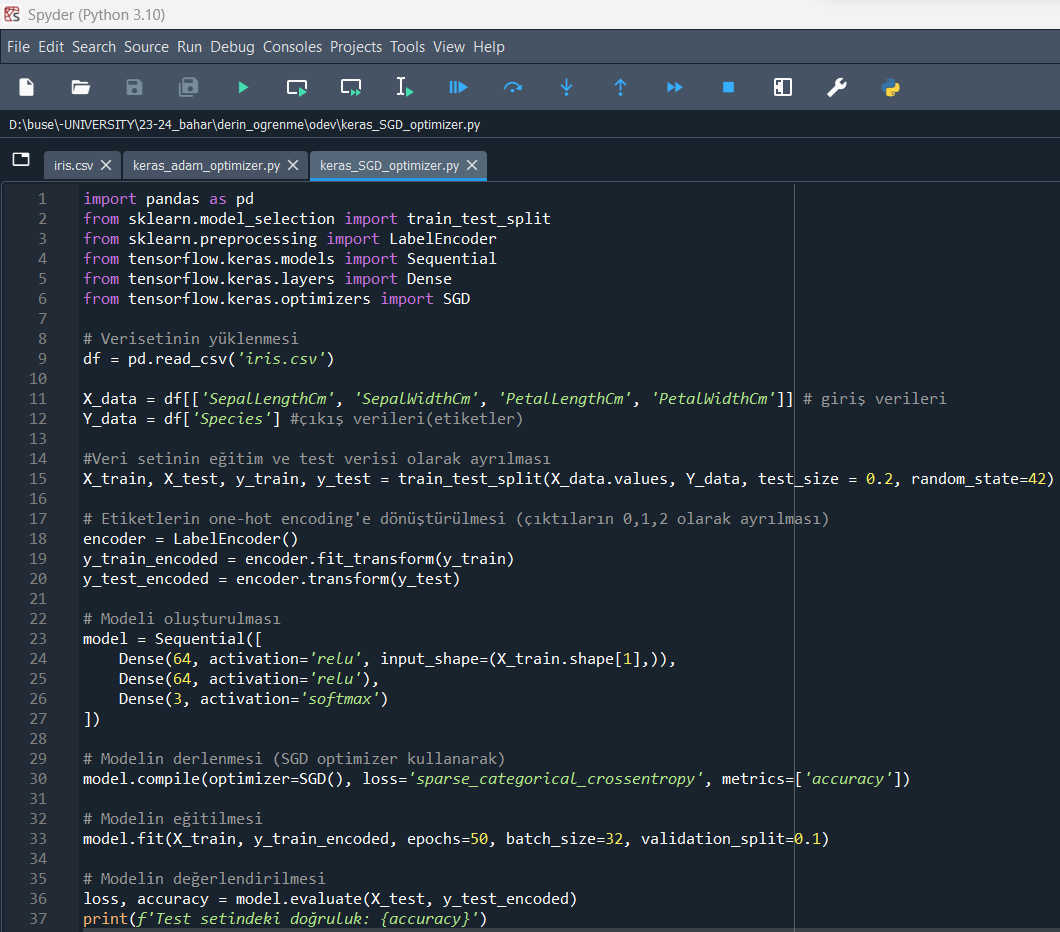
**Adam:** Adam optimizasyonu, birinci dereceden ve ikinci dereceden momentlerin uyarlanabilir tahminine dayanan stokastik bir gradyan iniş yöntemidir.

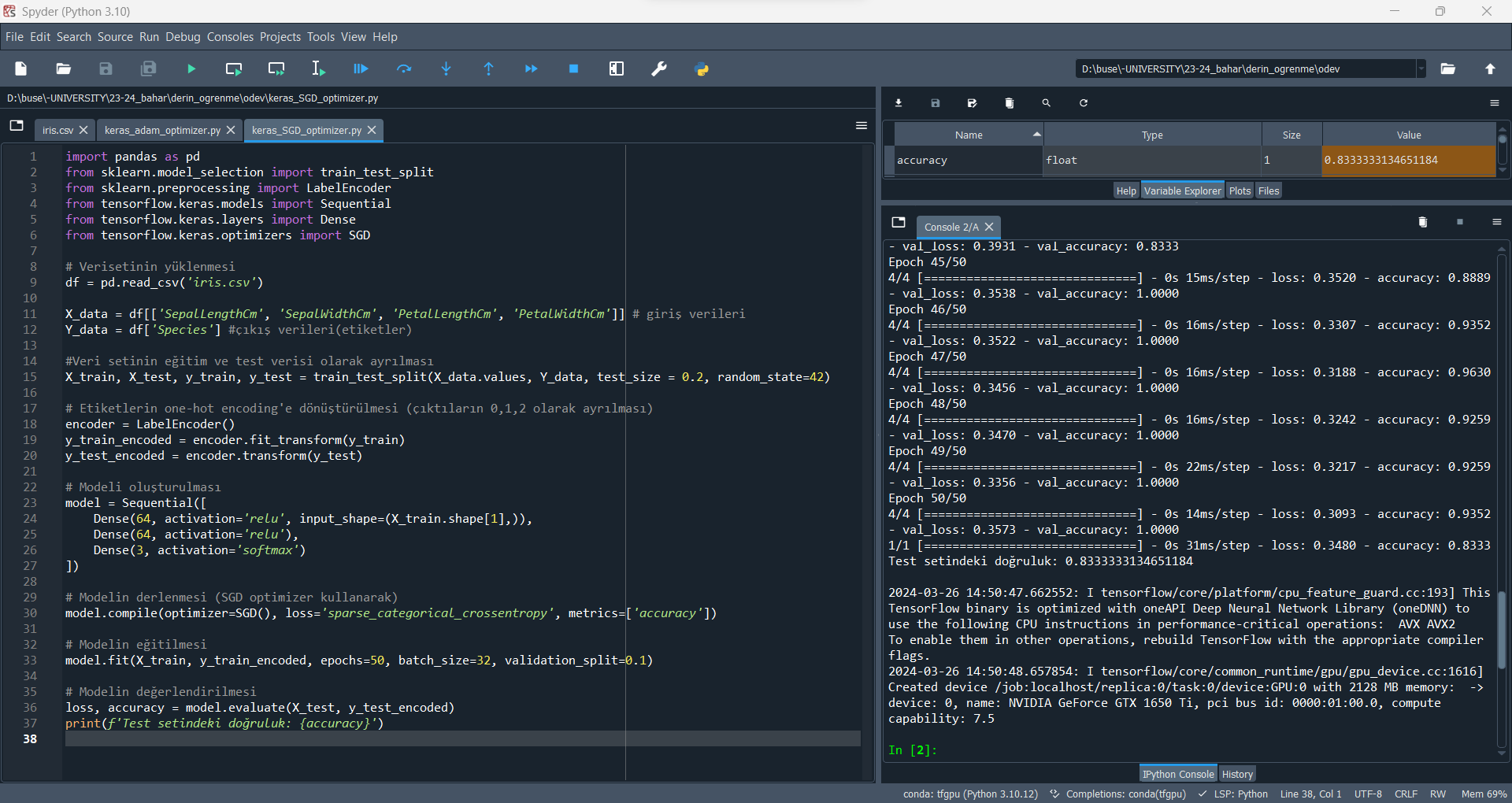
**Kod Üzerinde Gösterimi:**

**metin, ekran görüntüsü, yazılım, multimedya yazılımı içeren bir resim

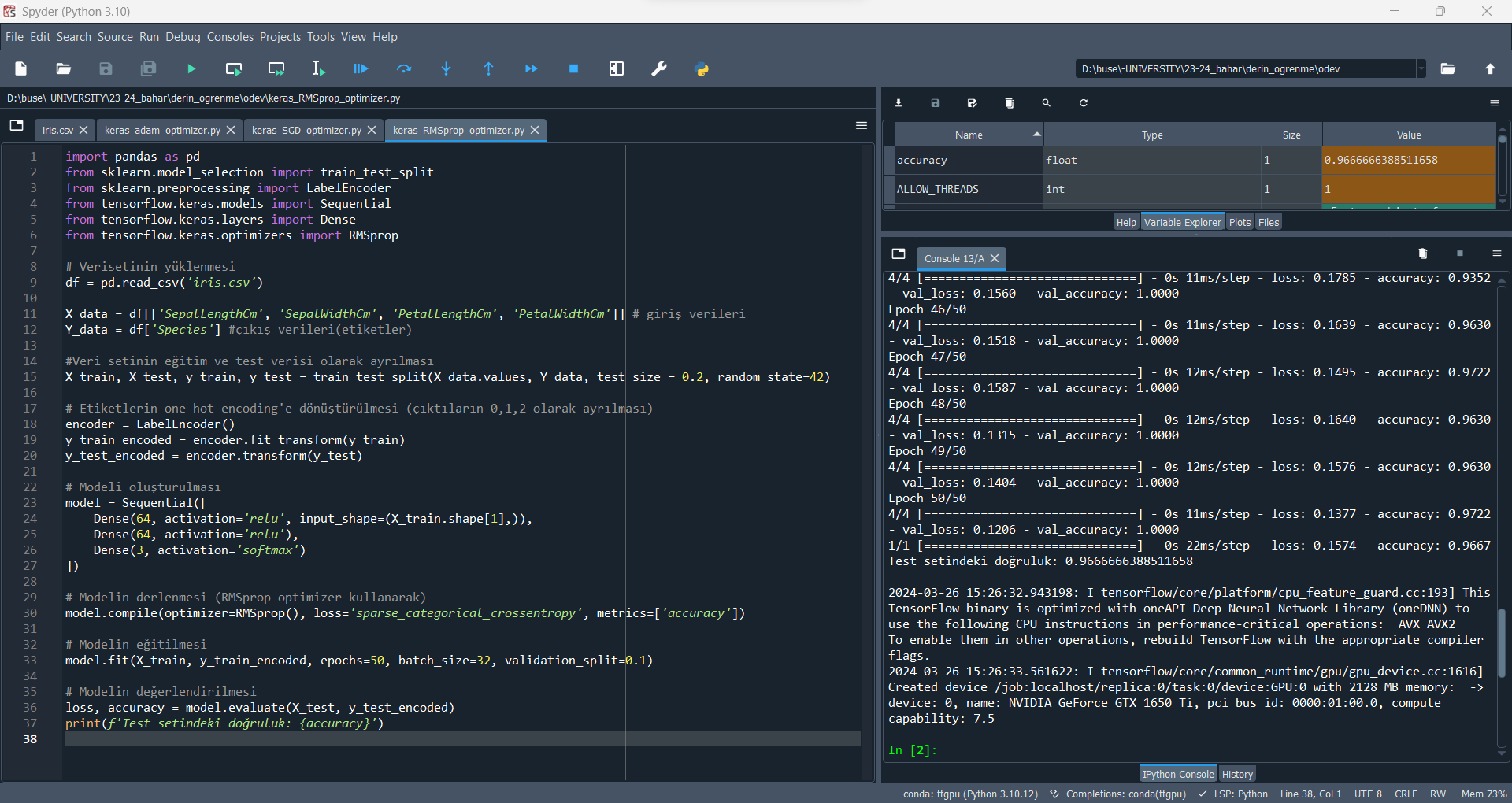
Açıklama otomatik olarak oluşturulduSonuç**

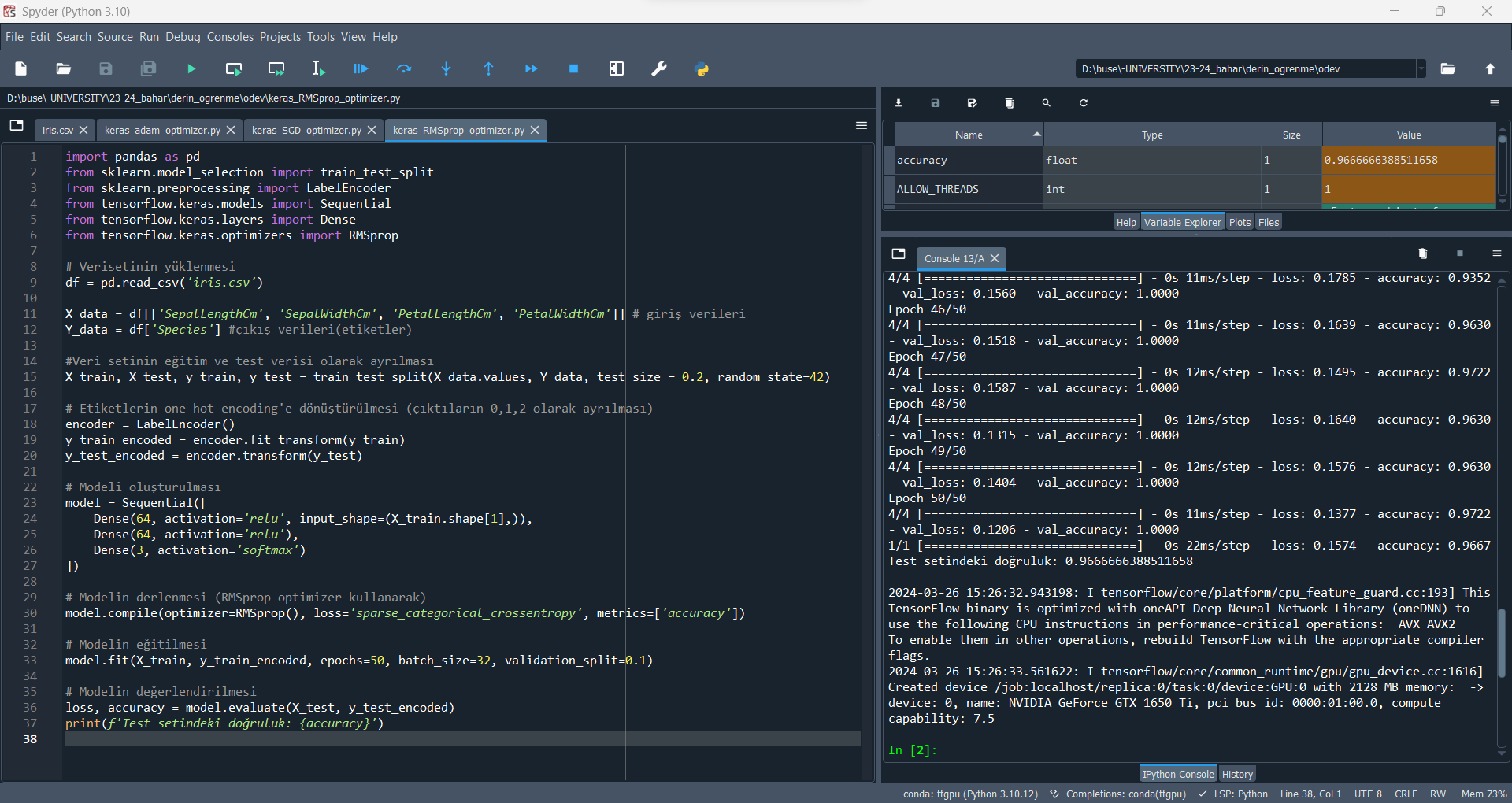
**SGD (Stochastic Gradient Descent):** Büyük veri setleri veya yüksek boyutlu özellik uzayları gibi durumlarda kullanılır. SGD'nin ana amacı, modelin parametrelerini (ağırlıklarını) optimize etmek için gradyan iniş yöntemini kullanmaktır.

**Kod Üzerinde Gösterimi:**

**Sonuç:**

**RMSprop:** Ortalanmış versiyon ayrıca gradyanların hareketli ortalamasını korur ve varyansı tahmin etmek için bu ortalamayı kullanır.

**Kod Üzerinde Gösterimi:**

**Sonuç:**

**Aktivasyon Fonksiyonları:**

**ReLU (Rectified Linear Unit)**: En yaygın kullanılan aktivasyon fonksiyonudur. Negatif değerler için 0 döndürürken, pozitif değerler için aynı değeri döndürür. Basit ve etkilidir.

model = Sequential([

Dense(64, activation='relu', input\_shape=(X\_train.shape[1],)),

Dense(64, activation='relu'),

Dense(3, activation='softmax')

])

Test setindeki doğruluk: 1.0

**Sigmoid**: Çıktıyı 0 ile 1 arasında bir değere sıkıştırır. Özellikle ikili sınıflandırma problemlerinde ve çıkışın olasılık değeri olarak yorumlandığı durumlarda kullanılır.

model = Sequential([

Dense(64, activation='sigmoid', input\_shape=(X\_train.shape[1],)),

Dense(64, activation='sigmoid'),

Dense(3, activation='softmax')

])

Test setindeki doğruluk: 1.0

**Softmax**: Çoklu sınıflandırma problemlerinde kullanılır. Çıktıları olasılık dağılımı olarak yorumlamak için kullanılır ve toplam çıktıların 1'e eşit olduğundan emin olur.

**Tanh (Hyperbolic Tangent)**: Sigmoid fonksiyonuna benzer, ancak çıktıları -1 ile 1 arasında bir değere sıkıştırır. Genellikle gizli katmanlarda kullanılır.

model = Sequential([

Dense(64, activation='tanh', input\_shape=(X\_train.shape[1],)),

Dense(64, activation='tanh'),

Dense(3, activation='softmax')

])

Test setindeki doğruluk: 1.0

**Loss Fonksiyonları:**

**Mean Squared Error (MSE)**: Gerçek ve tahmin edilen değerler arasındaki ortalama kare farkıdır. Regresyon problemleri için kullanılır.

model.compile(optimizer=Adam(), loss='mean\_squared\_error', metrics=['accuracy'])

Loss değeri: 1.18889319896698

**Binary Crossentropy**: İkili sınıflandırma problemleri için kullanılır. Gerçek ve tahmin edilen etiketler arasındaki farkı ölçer.

**Sparse Categorical Crossentropy**: Çoklu sınıflandırma problemleri için kullanılır. Etiketlerin doğrudan tamsayı değerleri olduğu durumlarda kullanılır.

model.compile(optimizer=Adam(), loss='sparse\_categorical\_crossentropy', metrics=['accuracy'])

Loss değeri: 0.10350186377763748

**Categorical Crossentropy**: Çoklu sınıflandırma problemleri için kullanılır. Etiketler one-hot encoding olarak temsil edildiğinde kullanılır.

**Huber Loss**: Regresyon problemleri için kullanılır ve ortalama kare hata (MSE) ile mutlak hata arasında bir yer tutar.

**Karşılaştırma Metrikleri:**

**Doğruluk (Accuracy)**:

model.compile(optimizer=Adam(), loss='sparse\_categorical\_crossentropy', metrics=['accuracy'])

Accuracy Değeri: 1.0

**Hassasiyet (Precision)**

**Duyarlılık (Recall veya Sensitivity)**

**F1 Skoru**

**Logaritmik Kayıp (Log Loss)**

**Ortalama Kare Hata (Mean Squared Error - MSE)**

metin, ekran görüntüsü, yazılım, multimedya yazılımı içeren bir resim

Açıklama otomatik olarak oluşturuldu