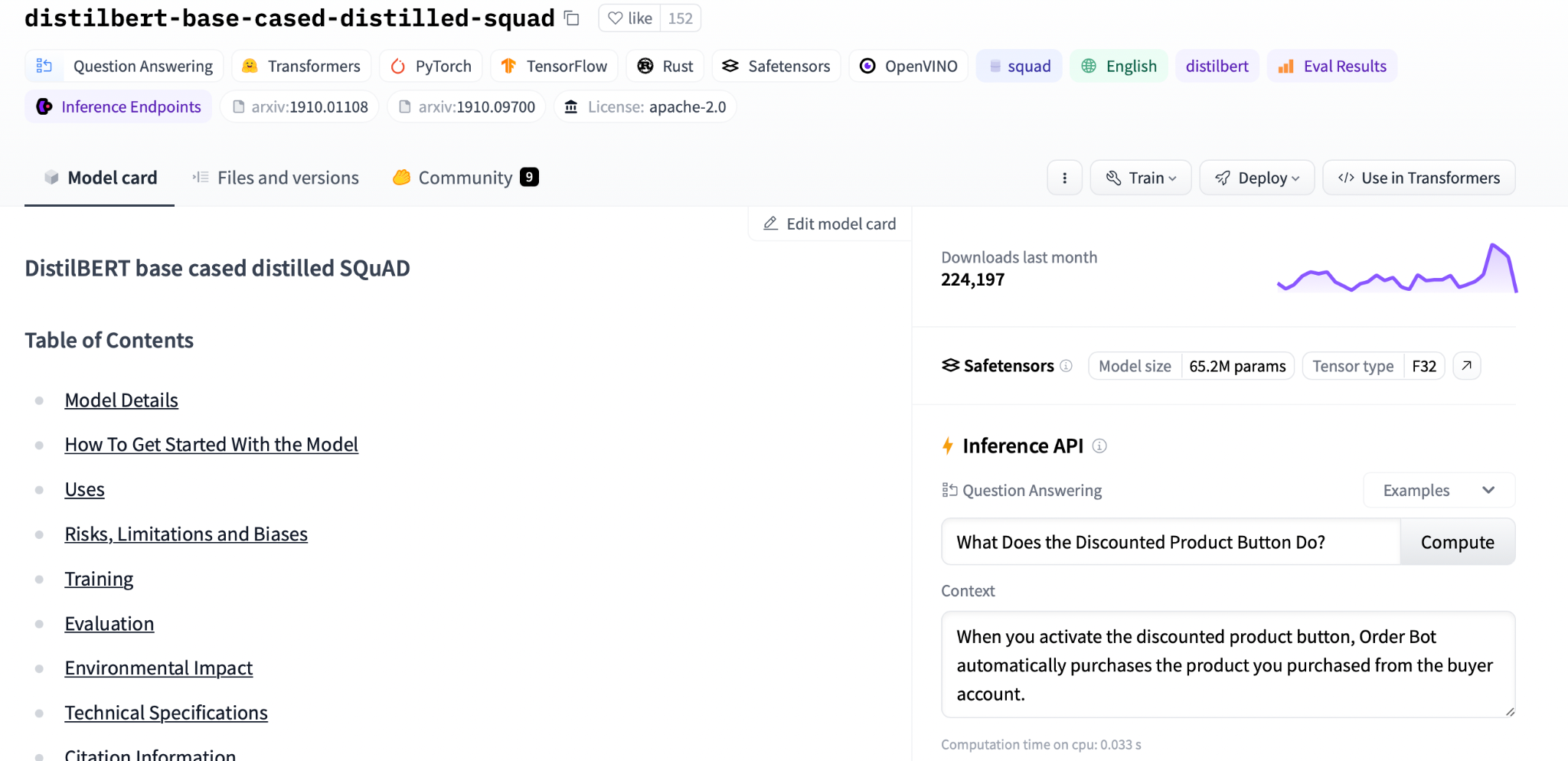
**OneAmz Chatbot**



**By Group 2**

***Verimiz ile ilgili genel bilgilendirme:***

Firma tarafından bize verilen 104 satirlik soru-cevap formatinda bir veri setimiz vardir. Mevcut Soru-Cevap modelleri için uygun içeriğe sahip olmadığı gözlenmiştir. Yaptigimiz calismalarda bu veri once İngilizceye cevrilmis, daha sonra bazi sorular eklenmistir. Mevcut model calismalari degerlendirildiginde, modellerimiz ‘Cevaplar’ sutununda geçen bir kelime ile üretilen bir soruya daha iyi cevap verdigi gözlenmiştir. Bunun da baslica sebebi, “Sorular” sütundaki verilerin soru formatinda olmadigi ve orada kullanılan kelimelerin cevaplarda geçmediği gorulmustur. Bundan dolayi modelimiz oradaki tum sorulara dogru cevabi dondurememistir.



Hugging Face platformunda bircok Ingilizce model ile denemeler yapildi. En cok kullanilan *deepset/roberta-base-squad2* ve *distilbert-base-cased-distilled-squad* modellerinin cevaplarinin daha mantikli ve iyi oldugu goruldu. Bu nedenle bu iki model uzerinde calismalar yapildi.

Streamlit için ikinci model tercih edilmistir.

BERT Modellerini Tercih Etmemizin Sebepleri

* BERT, gelen metni çift taraflı olarak incelediği için, herhangi bir kelimenin sağındaki ve solundaki kelimelerle olan ilişkisini çok iyi kavrıyor ve içeriği iyi bir şekilde öğreniyor.
* BERT, çift-yönlü olması dışında *Masked Language Modeling (MLM)* ve *Next Sentence Prediction (NSP)* adı verilen iki teknikle eğitiliyor.

Distil-BERT i tercih etmemizin amacı, BERT'in büyük boyutlu parametrelerini küçülterek model boyutunu azaltmaktır. Bu, daha hafif bir model elde edilmesini sağlar, böylece eğitim ve çıkarım süreçlerinde daha verimli hale gelir.DistilBERT, öğrenci-öğretmen (student-teacher) öğrenme yöntemini kullanarak eğitilir. Büyük ve karmaşık modellerden elde edilen öğrenmeyi, daha küçük bir öğrenci modeline aktarmak için bu yöntem kullanılır.

Bahsettiğimiz tekniklerle, hazır-eğitilmiş olarak modeller açık olarak paylaşılıyor.Bu yüzden, bu oluşturulan hazır-eğitilmiş modelleri fine-tuning adı verilen teknikle yeni problemlerin çözümünde kullanabiliyoruz.

BERT için gerekli olan 2 ana objeyi, tokenizer’ı ve model’i Hugging Face’ten indiriyoruz.

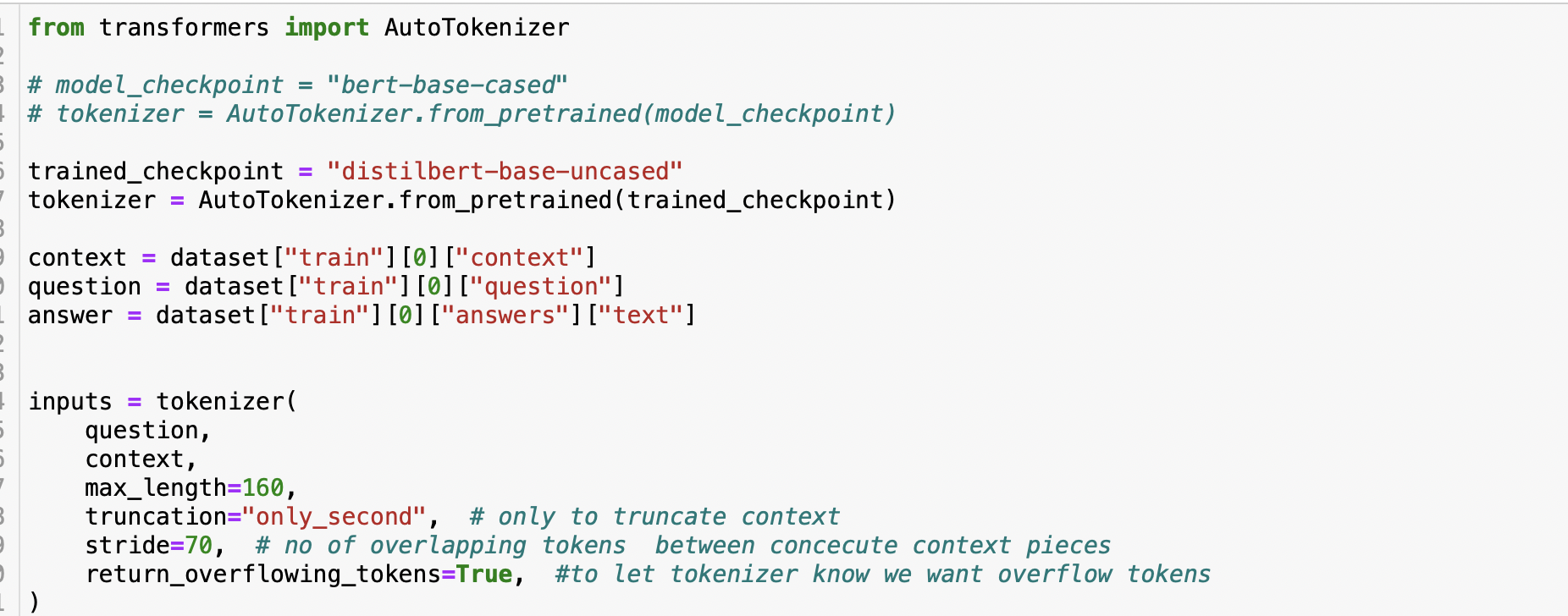
Tokenizer’ı, önceden sahip olunan kelime haznesini kullanarak metni öğelerine ayırma işleminde kullanılan araç olarak tanımlayabiliriz.

Metinlerin kelime sayısı 160'dan fazla olmasına rağmen girdi uzunluğunu maksimum 160 olarak belirtiyoruz. Daha büyük değerlerde GPU’nun memory si yetmediği için hata alıyoruz.

Turkce Model Secimi

Yukarida bahsettigimiz tum adimlar Turkce iyi çalışan modeller icinde yapildi. Fakat iyi performans gösteren bir model bulunamadi.Elimizdeki mevcut veri ile dbmdz/bert-base-turkish-128k-uncased

Modeli uzerinde calsilmistir. Burada elde edilen sonuçlar doğrultusunda modeller .json formatinda kaydedilip Streamlit e aktarilmistir. Turkce model icin colab da calistik (https://colab.research.google.com/github/ktoprakucar/fine-tuning-turkish-bert-model/blob/master/classification\_model.ipynb#scrollTo=QfNDxpE3C2bP).



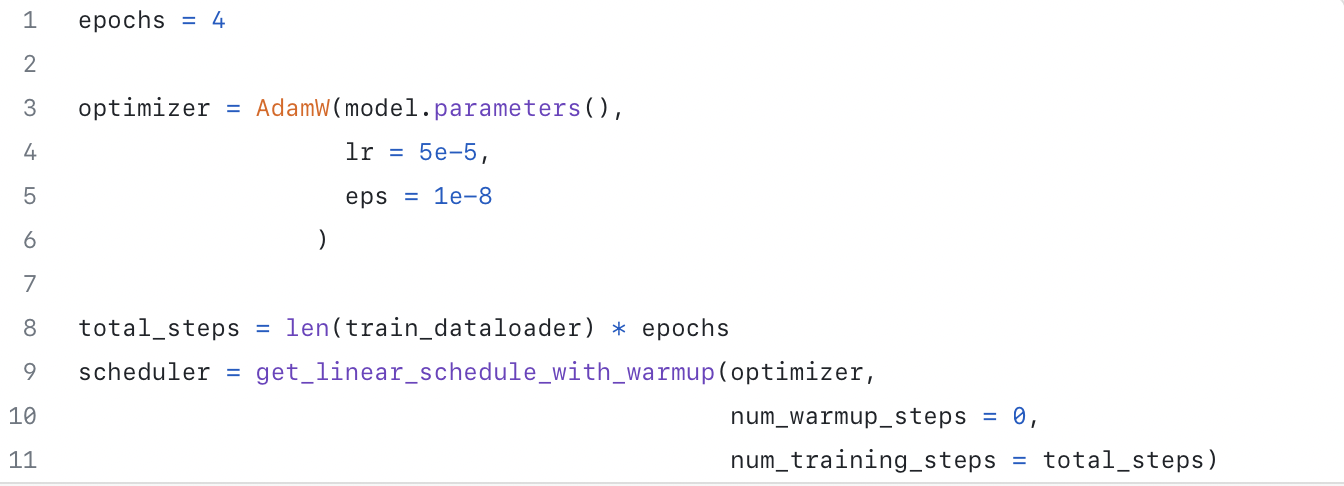
Öncelikle cümle, indirdiğimiz Tokenizer ile token larına ayrılıyor, sonrasında sınıflandırma probleminin çözülebilmesi için gerekli olan token’lar cümlenin sonuna ve başına ekleniyor. Cümle maksimum uzunluktan kısaysa, input vektörün sabit uzunlukta olduğu için boşluklar dolduruluyor, uzunsa metin limit kadar kelime ile ifade ediliyor. Attention mask’leri oluşturuluyor ve metinler işlemin sonucunda tensor objesi olarak geri dönüyor.

metindeki kelimeler tokenizer daki kelimelerin id’leri ile ifade ediliyor ve bu şekilde işleme sokuluyor.



Oluşturduğumuz tensor verisini modele vermek üzere *dataloader* değişkenine dönüştürüyoruz. Tokenizer’da olduğu gibi, önceden train edilmiş olan modeli fine tune etmek için Hugging Face den indiriyoruz.

öğrenmenin daha verimli olabilmesi ve *learning rate* optimizasyonu için bir scheduler yaratılıyor ve optimizer olarak *Adam Optimizer* kullanılıyor.

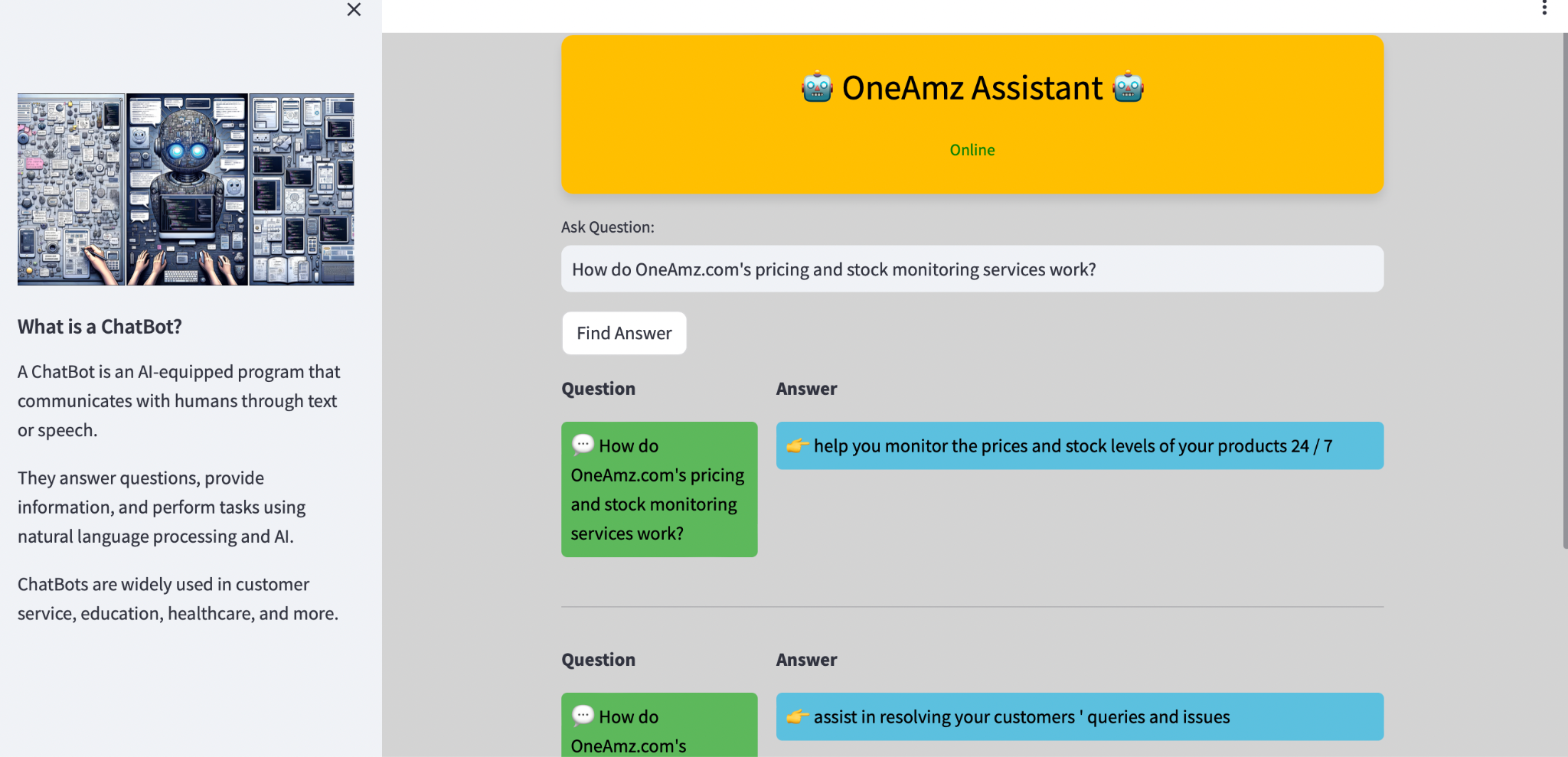


Her bölüm başlamadan önce optimize edilecek loss değeri sıfırlanıyor. Bu aşamada *train* metodu çağırılıyor. Test aşamasında ise *eval* metodu çağırılıyor. Çünkü modelin katmanları train ve eval metotlarında farklı olarak davranıyor. Dataloader’daki değerler GPU’ya aktarılıyor, gradient değerleri sıfırlanıyor ve output (logit) değerleri oluşuyor. Bu logit değerlerine bağlı olarak loss değeri hesaplanıyor. Backpropagation ile gradient ler tekrar hesaplanıyor ve son olarak da learning rate ile beraber parametreler de optimize ediliyor. Her bölümün sonunda, hesaplanan ortalama loss u inceleyebiliriz.

Training’deki model performansı incelemek için loss daki düşüşü inceliyoruz.



Bu islemler ayni sekilde test ve train datasetleri icin yapilir.



İngilizce ve Turkce icin benzer Streamlit arayuz oluşturduk. İngilizce modelimizin daha iyi cevaplar verdigi gözlenmiştir.