**1. Gerekli Kütüphaneleri İçe Aktar**

|  |
| --- |
| import pandas as pd  import torch  import torch.nn as nn  import torch.optim as optim  from sklearn.feature\_extraction.text import CountVectorizer  from sklearn.model\_selection import train\_test\_split  from sklearn.preprocessing import LabelEncoder  from sklearn.metrics import classification\_report |

Bu bölümde, gerekli kütüphaneler içe aktarılır. `pandas` veri işleme için, `torch` derin öğrenme modelleri için, `sklearn` ise metin vektörleştirme ve model değerlendirmesi için kullanılır.

**2. Veriyi Oku ve İlgili Sütunları Seç**

|  |
| --- |
| df = pd.read\_csv('banka.csv')  df = df[['sorgu', 'label']] |

Bu bölümde `banka.csv` dosyasından veri okunur ve sadece `sorgu` ve `label` sütunları seçilir.

**3. Stopwords Listesini Tanımla**

|  |
| --- |
| stopwords = [  'fakat', 'lakin', 'ancak', 'acaba', 'ama', 'aslında', 'az', 'bazı', 'belki',  'biri', 'birkaç', 'birşey', 'biz', 'bu', 'çok', 'çünkü', 'da', 'daha', 'de',  'defa', 'diye', 'eğer', 'en', 'gibi', 'hem', 'hep', 'hepsi', 'her', 'hiç',  'için', 'ile', 'ise', 'kez', 'ki', 'kim', 'mı', 'mu', 'mü', 'nasıl', 'ne',  'neden', 'nerde', 'nerede', 'nereye', 'niçin', 'niye', 'o', 'sanki', 'şey',  'siz', 'şu', 'tüm', 've', 'veya', 'ya', 'yani'  ] |

Bu bölümde, Türkçe stopwords (önemsiz kelimeler) listesi tanımlanır. Bu kelimeler metin temizleme aşamasında metinden çıkarılacaktır.

**4. Kullanıcıdan Yeni Mesaj Al ve DataFrame'e Ekleyerek Temizleme**

|  |
| --- |
| mesaj = input("Yapmak istediğiniz işlemi giriniz: ")  mesajdf = pd.DataFrame({'sorgu': [mesaj], 'label': ['yeni\_sınıf']})  df = pd.concat([df, mesajdf], ignore\_index=True) |

Bu bölümde kullanıcıdan yeni bir mesaj alınır ve geçici olarak `label` değeri `yeni\_sınıf` olarak atanır. Bu yeni veri mevcut DataFrame'e eklenir.

**5. Stopwords'leri Kaldır ve Büyük Harfleri Küçük Harfe Çevir**

|  |
| --- |
| for word in stopwords:  df['sorgu'] = df['sorgu'].str.replace(r'\b' + word + r'\b', '', regex=True)  df['sorgu'] = df['sorgu'].str.lower() |

Bu bölümde, tanımlanan stopwords'ler metinlerden kaldırılır ve tüm metinler küçük harfe çevrilir.

**6. Metinleri Vektörize Et ve Etiketleri Kodla**

|  |
| --- |
| cv = CountVectorizer(max\_features=200)  x = cv.fit\_transform(df['sorgu']).toarray()  le = LabelEncoder()  df['label'] = le.fit\_transform(df['label'].astype(str))  y = df['label'].values |

Bu bölümde, metinler `CountVectorizer` ile vektörize edilir (en sık kullanılan 200 kelime ile). `LabelEncoder` ile etiketler sayısal değerlere dönüştürülür.

**7. Veriyi Tensorlere Dönüştür ve Eğitim-Test Verisi Olarak Böl**

|  |
| --- |
| x = torch.tensor(x, dtype=torch.float32)  y = torch.tensor(y, dtype=torch.long)  x\_train, x\_test, y\_train, y\_test = train\_test\_split(x, y, random\_state=42, train\_size=0.8) |

Bu bölümde, veriler PyTorch tensörlerine dönüştürülür ve eğitim-test verisi olarak bölünür.

**8. Yapay Sinir Ağı Modelini Oluştur**

|  |
| --- |
| class Net(nn.Module):  def \_\_init\_\_(self, input\_size, hidden\_size, output\_size):  super(Net, self).\_\_init\_\_()  self.fc1 = nn.Linear(input\_size, hidden\_size)  self.fc2 = nn.Linear(hidden\_size, hidden\_size)  self.fc3 = nn.Linear(hidden\_size, output\_size)  self.dropout = nn.Dropout(0.5) # Dropout ekle  def forward(self, x):  x = torch.relu(self.fc1(x))  x = self.dropout(x)  x = torch.relu(self.fc2(x))  x = self.dropout(x)  x = self.fc3(x)  return x  input\_size = x.shape[1]  hidden\_size = 256 # Daha geniş bir gizli katman boyutu  output\_size = len(le.classes\_) # Geçici sınıfı ekle  model = Net(input\_size, hidden\_size, output\_size) |

Bu bölümde, üç katmanlı bir sinir ağı modeli oluşturulur. `Dropout` katmanı eklenerek aşırı öğrenmenin önüne geçilir.

**9. Kayıp Fonksiyonu ve Optimizasyon Algoritmasını Tanımla**

|  |
| --- |
| criterion = nn.CrossEntropyLoss()  optimizer = optim.Adam(model.parameters(), lr=0.001) |

Bu bölümde, kayıp fonksiyonu olarak `CrossEntropyLoss` ve optimizasyon algoritması olarak `Adam` kullanılır.

**10. Modeli Eğit**

|  |
| --- |
| epochs = 50 # Eğitim süresini artır  for epoch in range(epochs):  model.train()  optimizer.zero\_grad()  outputs = model(x\_train)  loss = criterion(outputs, y\_train)  loss.backward()  optimizer.step()  if (epoch + 1) % 10 == 0:  print(f'Epoch [{epoch+1}/{epochs}], Loss: {loss.item():.4f}') |

Bu bölümde, model 50 epoch boyunca eğitilir ve her 10 epoch'ta bir kayıp değeri yazdırılır.

**11. Modelin Performansını Değerlendir**

|  |
| --- |
| model.eval()  with torch.no\_grad():  outputs = model(x\_test)  \_, predicted = torch.max(outputs, 1)  print(classification\_report(y\_test, predicted, labels=range(len(le.classes\_)), target\_names=le.classes\_)) |

Bu bölümde, model test verisi üzerinde değerlendirilir ve sınıflandırma raporu oluşturulur.

**12. Yeni Mesajın Tahminini Al**

|  |
| --- |
| y\_pred\_new\_message = model(x[-1].unsqueeze(0))  \_, predicted\_class = torch.max(y\_pred\_new\_message, 1)  predicted\_class = predicted\_class.item()  predicted\_label = le.inverse\_transform([predicted\_class])[0]  print(f'\nSonuç: {predicted\_label}') |

Bu bölümde, yeni eklenen mesajın sınıfı tahmin edilir ve sonuç yazdırılır.

Bu adımlar ile metin verisi temizlenir, vektörize edilir, yapay sinir ağı ile eğitilir ve yeni bir mesajın sınıfı tahmin edilir.

**13. Kodu Çalıştırma**

* Python 3.11.9 Kurulumunu yapın.
* Python extension kurulumunu yapın.
* Terminali açın ve ‘pip’ Python standart paket yönetim sistemi ile kütüphaneleri yükleyin

‘pip install pandas’

‘pip install torch’

‘pip install sickit-learn’

* Projeyi run edin, ister buton aracılığı ile ister projenin bulunduğu dizide terminal açarak “python <projeismi>.py” ile calistirabilirsiniz.