Bert Algoritması Sonuçları

# Gerekli kütüphanelerin yüklenmesi

!pip install transformers datasets evaluate scikit-learn

from transformers import AutoTokenizer, AutoModelForSequenceClassification, Trainer, TrainingArguments

from datasets import load\_dataset

from evaluate import load as load\_evaluate  # load\_metric yerine evaluate kullanımı

from sklearn.metrics import accuracy\_score, f1\_score

import numpy as np

import os

# wandb izlemeyi kapatma

os.environ["WANDB\_DISABLED"] = "true"

# 1. Veri Setini Yükleme (hızlı test için küçük bir alt küme seçiyoruz)

dataset = load\_dataset("tweet\_eval", "emotion")

# 2. Model ve Tokenizer Ayarlama

model\_name = "distilbert-base-uncased"  # Alternatif olarak "bert-base-uncased" kullanılabilir

tokenizer = AutoTokenizer.from\_pretrained(model\_name)

model = AutoModelForSequenceClassification.from\_pretrained(model\_name, num\_labels=dataset["train"].features["label"].num\_classes)

# 3. Veri Setini Tokenize Etme

def tokenize\_function(examples):

    return tokenizer(examples["text"], padding="max\_length", truncation=True)

tokenized\_datasets = dataset.map(tokenize\_function, batched=True)

# Hızlı sonuç için veri setini küçük bir alt kümeye indirgeme

small\_train\_dataset = tokenized\_datasets["train"].shuffle(seed=42).select(range(100))  # 100 örnek

small\_eval\_dataset = tokenized\_datasets["validation"].shuffle(seed=42).select(range(50))  # 50 örnek

# 4. Eğitim Parametrelerinin Ayarlanması (epoch sayısını ve batch boyutunu küçültme)

training\_args = TrainingArguments(

    output\_dir="./results",

    evaluation\_strategy="epoch",

    logging\_dir="./logs",

    per\_device\_train\_batch\_size=8,  # Batch boyutu küçültüldü

    per\_device\_eval\_batch\_size=8,

    num\_train\_epochs=1,  # Epoch sayısı küçültüldü

)

# 5. Performans Değerlendirme Fonksiyonu

def compute\_metrics(p):

    preds = np.argmax(p.predictions, axis=1)

    acc = accuracy\_score(p.label\_ids, preds)

    f1 = f1\_score(p.label\_ids, preds, average="weighted")

    return {"accuracy": acc, "f1": f1}

# 6. Model Eğitimi

trainer = Trainer(

    model=model,

    args=training\_args,

    train\_dataset=small\_train\_dataset,

    eval\_dataset=small\_eval\_dataset,

    compute\_metrics=compute\_metrics,

)

trainer.train()

# 7. Modeli Değerlendirme

eval\_results = trainer.evaluate(tokenized\_datasets["test"])

print("Test Sonuçları:", eval\_results)

Test Sonuçları: {'eval\_loss': 1.305336594581604, 'eval\_accuracy': 0.4004222378606615, 'eval\_f1': 0.25311524144652225, 'eval\_runtime': 1316.4423, 'eval\_samples\_per\_second': 1.079, 'eval\_steps\_per\_second': 0.135, 'epoch': 1.0}

# Gerekli kütüphanelerin yüklenmesi

!pip install transformers datasets evaluate scikit-learn

from transformers import AutoTokenizer, AutoModelForSequenceClassification, Trainer, TrainingArguments

from datasets import Dataset

import pandas as pd

from sklearn.metrics import accuracy\_score, f1\_score

import numpy as np

import os

# wandb izlemeyi kapatma

os.environ["WANDB\_DISABLED"] = "true"

# 1. Veriyi Yükle

url = "https://raw.githubusercontent.com/dD2405/Twitter\_Sentiment\_Analysis/master/train.csv"

df = pd.read\_csv(url)

# 'tweet' ve 'label' sütunlarını seçme

df = df[['tweet', 'label']]

# Yalnızca 100 örneklik küçük bir veri kümesi oluştur

df = df.sample(100, random\_state=42)  # Rastgele 100 örnek seç

# Veriyi Hugging Face datasets formatına dönüştür

dataset = Dataset.from\_pandas(df)

# 2. Model ve Tokenizer Ayarlama

model\_name = "distilbert-base-uncased"  # Alternatif olarak "bert-base-uncased" kullanılabilir

tokenizer = AutoTokenizer.from\_pretrained(model\_name)

model = AutoModelForSequenceClassification.from\_pretrained(model\_name, num\_labels=2)

# 3. Veri Setini Tokenize Etme

def tokenize\_function(examples):

    return tokenizer(examples['tweet'], padding="max\_length", truncation=True, max\_length=128)

tokenized\_datasets = dataset.map(tokenize\_function, batched=True)

# Eğitim ve test verisini ayır

train\_dataset, eval\_dataset = tokenized\_datasets.train\_test\_split(test\_size=0.2).values()

# 4. Eğitim Parametrelerini Ayarlama

training\_args = TrainingArguments(

    output\_dir="./results",

    evaluation\_strategy="epoch",

    logging\_dir="./logs",

    per\_device\_train\_batch\_size=8,  # Batch boyutunu küçülttük

    per\_device\_eval\_batch\_size=8,

    num\_train\_epochs=1,  # Epoch sayısını küçülttük

)

# 5. Performans Değerlendirme Fonksiyonu

def compute\_metrics(p):

    preds = np.argmax(p.predictions, axis=1)

    acc = accuracy\_score(p.label\_ids, preds)

    f1 = f1\_score(p.label\_ids, preds, average="weighted")

    return {"accuracy": acc, "f1": f1}

# 6. Model Eğitimi

trainer = Trainer(

    model=model,

    args=training\_args,

    train\_dataset=train\_dataset,

    eval\_dataset=eval\_dataset,

    compute\_metrics=compute\_metrics,

)

trainer.train()

# 7. Modeli Değerlendirme

eval\_results = trainer.evaluate()

print("Test Sonuçları:", eval\_results)

Test Sonuçları: {'eval\_loss': 0.11907094717025757, 'eval\_accuracy': 1.0, 'eval\_f1': 1.0, 'eval\_runtime': 5.1077, 'eval\_samples\_per\_second': 3.916, 'eval\_steps\_per\_second': 0.587, 'epoch': 1.0}

# Gerekli kütüphanelerin yüklenmesi

!pip install transformers datasets evaluate scikit-learn

from transformers import AutoTokenizer, AutoModelForSequenceClassification, Trainer, TrainingArguments

from datasets import Dataset

import pandas as pd

from sklearn.metrics import accuracy\_score, f1\_score

import numpy as np

import os

# wandb izlemeyi kapatma

os.environ["WANDB\_DISABLED"] = "true"

# 1. Veriyi Yükle

url = "https://raw.githubusercontent.com/dD2405/Twitter\_Sentiment\_Analysis/master/train.csv"

df = pd.read\_csv(url)

# 'tweet' ve 'label' sütunlarını seçme

df = df[['tweet', 'label']]

# 500 örneklik küçük bir veri kümesi oluştur

df = df.sample(500, random\_state=42)  # Rastgele 500 örnek seç

# Veriyi Hugging Face datasets formatına dönüştür

dataset = Dataset.from\_pandas(df)

# 2. Model ve Tokenizer Ayarlama

model\_name = "distilbert-base-uncased"  # Alternatif olarak "bert-base-uncased" kullanılabilir

tokenizer = AutoTokenizer.from\_pretrained(model\_name)

model = AutoModelForSequenceClassification.from\_pretrained(model\_name, num\_labels=2)

# 3. Veri Setini Tokenize Etme

def tokenize\_function(examples):

    return tokenizer(examples['tweet'], padding="max\_length", truncation=True, max\_length=128)

tokenized\_datasets = dataset.map(tokenize\_function, batched=True)

# Eğitim ve test verisini ayır

train\_dataset, eval\_dataset = tokenized\_datasets.train\_test\_split(test\_size=0.2).values()

# 4. Eğitim Parametrelerini Ayarlama

training\_args = TrainingArguments(

    output\_dir="./results",

    evaluation\_strategy="epoch",

    logging\_dir="./logs",

    per\_device\_train\_batch\_size=8,  # Batch boyutunu küçülttük

    per\_device\_eval\_batch\_size=8,

    num\_train\_epochs=1,  # Epoch sayısını küçülttük

)

# 5. Performans Değerlendirme Fonksiyonu

def compute\_metrics(p):

    preds = np.argmax(p.predictions, axis=1)

    acc = accuracy\_score(p.label\_ids, preds)

    f1 = f1\_score(p.label\_ids, preds, average="weighted")

    return {"accuracy": acc, "f1": f1}

# 6. Model Eğitimi

trainer = Trainer(

    model=model,

    args=training\_args,

    train\_dataset=train\_dataset,

    eval\_dataset=eval\_dataset,

    compute\_metrics=compute\_metrics,

)

trainer.train()

# 7. Modeli Değerlendirme

eval\_results = trainer.evaluate()

print("Test Sonuçları:", eval\_results)

Test Sonuçları: {'eval\_loss': 0.17771455645561218, 'eval\_accuracy': 0.96, 'eval\_f1': 0.9404081632653061, 'eval\_runtime': 18.7039, 'eval\_samples\_per\_second': 5.346, 'eval\_steps\_per\_second': 0.695, 'epoch': 1.0}

# Gerekli kütüphanelerin yüklenmesi

!pip install transformers datasets evaluate scikit-learn

from transformers import AutoTokenizer, AutoModelForSequenceClassification, Trainer, TrainingArguments

from datasets import Dataset

import pandas as pd

from sklearn.metrics import accuracy\_score, f1\_score

import numpy as np

import os

# wandb izlemeyi kapatma

os.environ["WANDB\_DISABLED"] = "true"

# 1. Veriyi Yükle

url = "https://raw.githubusercontent.com/dD2405/Twitter\_Sentiment\_Analysis/master/train.csv"

df = pd.read\_csv(url)

# 'tweet' ve 'label' sütunlarını seçme

df = df[['tweet', 'label']]

# Yalnızca 1000 örneklik küçük bir veri kümesi oluştur

df = df.sample(1000, random\_state=42)  # Rastgele 1000 örnek seç

# Veriyi Hugging Face datasets formatına dönüştür

dataset = Dataset.from\_pandas(df)

# 2. Model ve Tokenizer Ayarlama

model\_name = "distilbert-base-uncased"  # Alternatif olarak "bert-base-uncased" kullanılabilir

tokenizer = AutoTokenizer.from\_pretrained(model\_name)

model = AutoModelForSequenceClassification.from\_pretrained(model\_name, num\_labels=2)

# 3. Veri Setini Tokenize Etme

def tokenize\_function(examples):

    return tokenizer(examples['tweet'], padding="max\_length", truncation=True, max\_length=128)

tokenized\_datasets = dataset.map(tokenize\_function, batched=True)

# Eğitim ve test verisini ayır

train\_dataset, eval\_dataset = tokenized\_datasets.train\_test\_split(test\_size=0.2).values()

# 4. Eğitim Parametrelerini Ayarlama

training\_args = TrainingArguments(

    output\_dir="./results",

    evaluation\_strategy="epoch",

    logging\_dir="./logs",

    per\_device\_train\_batch\_size=8,  # Batch boyutunu küçülttük

    per\_device\_eval\_batch\_size=8,

    num\_train\_epochs=1,  # Epoch sayısını küçülttük

)

# 5. Performans Değerlendirme Fonksiyonu

def compute\_metrics(p):

    preds = np.argmax(p.predictions, axis=1)

    acc = accuracy\_score(p.label\_ids, preds)

    f1 = f1\_score(p.label\_ids, preds, average="weighted")

    return {"accuracy": acc, "f1": f1}

# 6. Model Eğitimi

trainer = Trainer(

    model=model,

    args=training\_args,

    train\_dataset=train\_dataset,

    eval\_dataset=eval\_dataset,

    compute\_metrics=compute\_metrics,

)

trainer.train()

# 7. Modeli Değerlendirme

eval\_results = trainer.evaluate()

print("Test Sonuçları:", eval\_results)

Test Sonuçları: {'eval\_loss': 0.2680715322494507, 'eval\_accuracy': 0.93, 'eval\_f1': 0.8962694300518135, 'eval\_runtime': 39.9171, 'eval\_samples\_per\_second': 5.01, 'eval\_steps\_per\_second': 0.626, 'epoch': 1.0}

#LLM kullanrak analiz - DistilGPT-2

# Gerekli kütüphanelerin yüklenmesi

!pip install transformers datasets evaluate scikit-learn

from transformers import AutoTokenizer, AutoModelForSequenceClassification, Trainer, TrainingArguments

from datasets import Dataset

import pandas as pd

from sklearn.metrics import accuracy\_score, f1\_score

import numpy as np

import os

# wandb izlemeyi kapatma

os.environ["WANDB\_DISABLED"] = "true"

url = "https://raw.githubusercontent.com/dD2405/Twitter\_Sentiment\_Analysis/master/train.csv"

df = pd.read\_csv(url)

# 'tweet' ve 'label' sütunlarını seçme ve sadece 100 örnekle sınırlama

df = df[['tweet', 'label']].sample(100, random\_state=42)

# Veriyi Hugging Face datasets formatına dönüştür

dataset = Dataset.from\_pandas(df)

# 2. Model ve Tokenizer Ayarlama (DistilGPT-2)

model\_name = "distilgpt2"  # Hafif bir model tercih edildi

tokenizer = AutoTokenizer.from\_pretrained(model\_name)

# padding token'ı ekleyelim

tokenizer.pad\_token = tokenizer.eos\_token

# Modelin pad\_token\_id'sini de güncelleyelim

model = AutoModelForSequenceClassification.from\_pretrained(model\_name, num\_labels=2)

model.config.pad\_token\_id = tokenizer.pad\_token\_id

# 3. Veri Setini Tokenize Etme

def tokenize\_function(examples):

    return tokenizer(examples['tweet'], padding="max\_length", truncation=True, max\_length=64)

tokenized\_datasets = dataset.map(tokenize\_function, batched=True)

# Eğitim ve test verisini ayır

train\_dataset, eval\_dataset = tokenized\_datasets.train\_test\_split(test\_size=0.2).values()

# 4. Eğitim Parametrelerini Ayarlama

training\_args = TrainingArguments(

    output\_dir="./results",

    evaluation\_strategy="epoch",

    logging\_dir="./logs",

    per\_device\_train\_batch\_size=4,  # Küçük batch boyutu

    per\_device\_eval\_batch\_size=4,

    num\_train\_epochs=1,  # Sadece 1 epoch

    logging\_steps=5,  # Daha sık logging (isteğe bağlı)

)

# 5. Performans Değerlendirme Fonksiyonu

def compute\_metrics(p):

    preds = np.argmax(p.predictions, axis=1)

    acc = accuracy\_score(p.label\_ids, preds)

    f1 = f1\_score(p.label\_ids, preds, average="weighted")

    return {"accuracy": acc, "f1": f1}

# 6. Model Eğitimi

trainer = Trainer(

    model=model,

    args=training\_args,

    train\_dataset=train\_dataset,

    eval\_dataset=eval\_dataset,

    compute\_metrics=compute\_metrics,

)

trainer.train()

# 7. Modeli Değerlendirme

eval\_results = trainer.evaluate()

print("Test Sonuçları:", eval\_results)

TweetEval veri setinnde LLM distilgpt

# Gerekli kütüphanelerin yüklenmesi

!pip install transformers datasets evaluate scikit-learn

from transformers import AutoTokenizer, AutoModelForSequenceClassification, Trainer, TrainingArguments

from datasets import load\_dataset

from sklearn.metrics import accuracy\_score, f1\_score

import numpy as np

import os

# wandb izlemeyi kapatma

os.environ["WANDB\_DISABLED"] = "true"

# 1. TweetEval Veri Setini Yükleme

dataset = load\_dataset("tweet\_eval", "emotion")

# 2. Model ve Tokenizer Ayarlama (DistilGPT-2)

model\_name = "distilgpt2"  # LLM diye geçiyor bir dil modeli tercih edildi

tokenizer = AutoTokenizer.from\_pretrained(model\_name)

# padding token'ı ekleyelim

tokenizer.pad\_token = tokenizer.eos\_token

# Modelin pad\_token\_id'sini de güncelleyelim

model = AutoModelForSequenceClassification.from\_pretrained(model\_name, num\_labels=dataset["train"].features["label"].num\_classes)

model.config.pad\_token\_id = tokenizer.pad\_token\_id

# 3. Veri Setini Tokenize Etme

def tokenize\_function(examples):

    return tokenizer(examples["text"], padding="max\_length", truncation=True, max\_length=64)

tokenized\_datasets = dataset.map(tokenize\_function, batched=True)

# Eğitim ve değerlendirme veri setlerini oluştur

train\_dataset = tokenized\_datasets["train"]

eval\_dataset = tokenized\_datasets["validation"]

# 4. Eğitim Parametrelerini Ayarlama

training\_args = TrainingArguments(

    output\_dir="./results",

    evaluation\_strategy="epoch",

    logging\_dir="./logs",

    per\_device\_train\_batch\_size=4,  # Küçük batch boyutu

    per\_device\_eval\_batch\_size=4,

    num\_train\_epochs=1,  # Sadece 1 epoch

    logging\_steps=5,  # Daha sık logging (isteğe bağlı)

)

# 5. Performans Değerlendirme Fonksiyonu

def compute\_metrics(p):

    preds = np.argmax(p.predictions, axis=1)

    acc = accuracy\_score(p.label\_ids, preds)

    f1 = f1\_score(p.label\_ids, preds, average="weighted")

    return {"accuracy": acc, "f1": f1}

# 6. Model Eğitimi

trainer = Trainer(

    model=model,

    args=training\_args,

    train\_dataset=train\_dataset,

    eval\_dataset=eval\_dataset,

    compute\_metrics=compute\_metrics,

)

trainer.train()

# 7. Modeli Değerlendirme

eval\_results = trainer.evaluate(tokenized\_datasets["test"])

print("Test Sonuçları:", eval\_results)

Test Sonuçları: {'eval\_loss': 0.7145752310752869, 'eval\_accuracy': 0.7318789584799437, 'eval\_f1': 0.7248910732187657, 'eval\_runtime': 189.1616, 'eval\_samples\_per\_second': 7.512, 'eval\_steps\_per\_second': 1.882, 'epoch': 1.0}

SPARK İLE tweet eval veri seti ÜZERİNDEN FONKSİYON DAĞILIMI VE PERFORMANS ANALİZİ

#Spark ile tweet eval veri seti fonksiyon dağılımı ve performans analizi

# Gerekli Kütüphaneleri Yükleme

!apt-get install openjdk-8-jdk-headless -qq > /dev/null

!wget -q https://archive.apache.org/dist/spark/spark-3.2.0/spark-3.2.0-bin-hadoop3.2.tgz  # Spark 3.2.0 kullanacağız

!tar -xvf spark-3.2.0-bin-hadoop3.2.tgz > /dev/null  # Dosyayı açıyoruz

!pip install -q findspark datasets

# JAVA ve SPARK ortamını ayarlama

import os

os.environ["JAVA\_HOME"] = "/usr/lib/jvm/java-8-openjdk-amd64"

os.environ["SPARK\_HOME"] = "/content/spark-3.2.0-bin-hadoop3.2"  # Spark 3.2.0 konumunu kullanıyoruz

# Spark ve Findspark'ı Başlat

import findspark

findspark.init()

# Gerekli Kütüphaneleri Yükleme

!apt-get install openjdk-8-jdk-headless -qq > /dev/null

!wget -q https://archive.apache.org/dist/spark/spark-3.2.0/spark-3.2.0-bin-hadoop3.2.tgz  # Spark 3.2.0 kullanacağız

!tar -xvf spark-3.2.0-bin-hadoop3.2.tgz > /dev/null  # Dosyayı açıyoruz

!pip install -q findspark datasets

# JAVA ve SPARK ortamını ayarlama

import os

os.environ["JAVA\_HOME"] = "/usr/lib/jvm/java-8-openjdk-amd64"

os.environ["SPARK\_HOME"] = "/content/spark-3.2.0-bin-hadoop3.2"  # Spark 3.2.0 konumunu kullanıyoruz

# Spark ve Findspark'ı Başlat

import findspark

findspark.init()

from pyspark.sql import SparkSession

from pyspark.sql.functions import col  # col fonksiyonunu içe aktarıyoruz

# Spark Oturumu Başlat

spark = SparkSession.builder \

    .appName("TweetEval Sentiment Analysis") \

    .getOrCreate()

# Hugging Face'ten Dataset Yükleme

from datasets import load\_dataset

import pandas as pd

# 'tweet\_eval' veri setini indirme

data = load\_dataset("tweet\_eval", "emotion")

train\_data = pd.DataFrame(data['train'])

train\_data.to\_csv("tweet\_eval\_emotion.csv", index=False)

# Spark DataFrame'e Yükleme

df = spark.read.csv("tweet\_eval\_emotion.csv", header=True, inferSchema=True)

df = df.select(col("text").alias("tweet"), col("label").cast("integer"))  # Sütunları yeniden adlandırma

# Veri Ön İşleme

from pyspark.sql.functions import lower, regexp\_replace, udf

from pyspark.sql.types import IntegerType

df = df.withColumn("tweet", lower(col("tweet")))  # Küçük harfe çevirme

df = df.withColumn("tweet", regexp\_replace(col("tweet"), "[^a-zA-Z\\s]", ""))  # Özel karakterleri temizleme

# Tokenization, TF-IDF ve Model Pipeline

from pyspark.ml.feature import Tokenizer, HashingTF, IDF

from pyspark.ml.classification import LogisticRegression

from pyspark.ml import Pipeline

from pyspark.ml.evaluation import MulticlassClassificationEvaluator

# Pipeline Oluşturma

tokenizer = Tokenizer(inputCol="tweet", outputCol="words")

hashing\_tf = HashingTF(inputCol="words", outputCol="rawFeatures", numFeatures=100)

idf = IDF(inputCol="rawFeatures", outputCol="features")

lr = LogisticRegression(featuresCol="features", labelCol="label")

pipeline = Pipeline(stages=[tokenizer, hashing\_tf, idf, lr])

# Modeli Eğitme

model = pipeline.fit(df)

# Tahminler

predictions = model.transform(df)

predictions.select("tweet", "label", "prediction").show(5)

# Performans Analizi

evaluator = MulticlassClassificationEvaluator(labelCol="label", predictionCol="prediction", metricName="accuracy")

accuracy = evaluator.evaluate(predictions)

print(f"Model Doğruluğu: {accuracy:.2f}")

# PySpark'ta Fonksiyon Dağıtımı

def tweet\_length(tweet):

    return len(tweet)

# UDF ile fonksiyon tanımlama

tweet\_length\_udf = udf(tweet\_length, IntegerType())

# Dağıtılmış fonksiyonu kullanma

df = df.withColumn("tweet\_length", tweet\_length\_udf(col("tweet")))

df.select("tweet", "tweet\_length").show(5)

Sonuç: +--------------------+-----+----------+

| tweet|label|prediction|

+--------------------+-----+----------+

|worry is a down p...| 2| 0.0|

|my roommate its o...| 0| 0.0|

|no but thats so c...| 1| 0.0|

|rooneys fucking u...| 0| 0.0|

|its pretty depres...| 3| 3.0|

+--------------------+-----+----------+

only showing top 5 rows

Model Doğruluğu: 0.51

+--------------------+------------+

| tweet|tweet\_length|

+--------------------+------------+

|worry is a down p...| 96|

|my roommate its o...| 92|

|no but thats so c...| 92|

|rooneys fucking u...| 97|

|its pretty depres...| 64|

+--------------------+------------+

only showing top 5 rows

2.deneme Bu sefer lojistik regresyon+spark

# Gerekli Kütüphaneleri Yükleme

!apt-get install openjdk-8-jdk-headless -qq > /dev/null

!wget -q https://archive.apache.org/dist/spark/spark-3.2.0/spark-3.2.0-bin-hadoop3.2.tgz

!tar xf spark-3.2.0-bin-hadoop3.2.tgz

!pip install -q findspark datasets

# JAVA ve SPARK ortamını ayarlama

import os

os.environ["JAVA\_HOME"] = "/usr/lib/jvm/java-8-openjdk-amd64"

os.environ["SPARK\_HOME"] = "/content/spark-3.2.0-bin-hadoop3.2"  # Spark 3.2.0 konumunu kullanıyoruz

# Spark ve Findspark'ı Başlat

import findspark

findspark.init()

from pyspark.sql import SparkSession

from pyspark.sql.functions import col, lower, regexp\_replace, udf

from pyspark.ml.feature import Tokenizer, HashingTF, IDF

from pyspark.ml.classification import LogisticRegression

from pyspark.ml import Pipeline

from pyspark.ml.evaluation import MulticlassClassificationEvaluator

# Spark Oturumu Başlat

spark = SparkSession.builder \

    .appName("TweetEval Sentiment Analysis") \

    .getOrCreate()

# Hugging Face'ten Dataset Yükleme

from datasets import load\_dataset

import pandas as pd

# 'tweet\_eval' veri setini indirme

data = load\_dataset("tweet\_eval", "emotion")

train\_data = pd.DataFrame(data['train'])

train\_data.to\_csv("tweet\_eval\_emotion.csv", index=False)

# Spark DataFrame'e Yükleme

df = spark.read.csv("tweet\_eval\_emotion.csv", header=True, inferSchema=True)

df = df.select(col("text").alias("tweet"), col("label").cast("integer"))  # Sütunları yeniden adlandırma

# Veri Ön İşleme - Küçük Harfe Çevirme ve Özel Karakterleri Temizleme

df = df.withColumn("tweet", lower(col("tweet")))  # Küçük harfe çevirme

df = df.withColumn("tweet", regexp\_replace(col("tweet"), "[^a-zA-Z\\s]", ""))  # Özel karakterleri temizleme

# Tokenization ve TF-IDF Özellik Çıkartımı

tokenizer = Tokenizer(inputCol="tweet", outputCol="words")

hashing\_tf = HashingTF(inputCol="words", outputCol="rawFeatures", numFeatures=1000)  # Daha fazla özellik

idf = IDF(inputCol="rawFeatures", outputCol="features")

# Modeli Tanımlama

lr = LogisticRegression(featuresCol="features", labelCol="label")

# Pipeline Tanımlama

pipeline = Pipeline(stages=[tokenizer, hashing\_tf, idf, lr])

# Modeli Eğitme

model = pipeline.fit(df)

# Tahminler

predictions = model.transform(df)

predictions.select("tweet", "label", "prediction").show(10)

# Performans Analizi

evaluator = MulticlassClassificationEvaluator(labelCol="label", predictionCol="prediction", metricName="accuracy")

accuracy = evaluator.evaluate(predictions)

print(f"Model Doğruluğu: {accuracy:.2f}")

# PySpark'ta Fonksiyon Dağıtımı

def tweet\_length(tweet):

    return len(tweet)

# UDF ile fonksiyon tanımlama

tweet\_length\_udf = udf(tweet\_length)

# Dağıtılmış fonksiyonu kullanma

df = df.withColumn("tweet\_length", tweet\_length\_udf(col("tweet")))

df.select("tweet", "tweet\_length").show(5)

+--------------------+-----+----------+

| tweet|label|prediction|

+--------------------+-----+----------+

|worry is a down p...| 2| 2.0|

|my roommate its o...| 0| 0.0|

|no but thats so c...| 1| 1.0|

|rooneys fucking u...| 0| 0.0|

|its pretty depres...| 3| 3.0|

|user but your pus...| 0| 0.0|

|making that yearl...| 3| 3.0|

|tiller and breezy...| 1| 1.0|

|user broadband is...| 0| 0.0|

|user look at thos...| 0| 0.0|

+--------------------+-----+----------+

only showing top 10 rows

Model Doğruluğu: 1.00

+--------------------+------------+

| tweet|tweet\_length|

+--------------------+------------+

|worry is a down p...| 96|

|my roommate its o...| 92|

|no but thats so c...| 92|

|rooneys fucking u...| 97|

|its pretty depres...| 64|

+--------------------+------------+

only showing top 5 rows

| **Epoch** | **Training Loss** | **Validation Loss** | **Accuracy** | **F1** |
| --- | --- | --- | --- | --- |
| 1 | 0.501700 | 0.703392 | 0.778075 | 0.776879 |

 [241/356 04:06 < 01:58, 0.97 it/s]

 [356/356 06:12]

Test Sonuçları: {'eval\_loss': 0.6476900577545166, 'eval\_accuracy': 0.7860661505981703, 'eval\_f1': 0.7835542928611373, 'eval\_runtime': 373.2136, 'eval\_samples\_per\_second': 3.807, 'eval\_steps\_per\_second': 0.954, 'epoch': 1.0}

LLM MODELİ

# Gerekli kütüphanelerin yüklenmesi

!pip install transformers datasets evaluate scikit-learn

from transformers import AutoTokenizer, AutoModelForSequenceClassification, Trainer, TrainingArguments

from datasets import load\_dataset

from sklearn.metrics import accuracy\_score, f1\_score

import numpy as np

import os

# wandb izlemeyi kapatma

os.environ["WANDB\_DISABLED"] = "true"

# 1. TweetEval Veri Setini Yükleme

dataset = load\_dataset("tweet\_eval", "emotion")

# 2. Model ve Tokenizer Ayarlama (GPT-2 LLM)

model\_name = "gpt2"  # GPT-2 kullanılıyor

tokenizer = AutoTokenizer.from\_pretrained(model\_name)

# padding token'ı ekleyelim

tokenizer.pad\_token = tokenizer.eos\_token

# Modelin pad\_token\_id'sini de güncelleyelim

model = AutoModelForSequenceClassification.from\_pretrained(model\_name, num\_labels=dataset["train"].features["label"].num\_classes)

model.config.pad\_token\_id = tokenizer.pad\_token\_id

# 3. Veri Setini Tokenize Etme

def tokenize\_function(examples):

    return tokenizer(examples["text"], padding="max\_length", truncation=True, max\_length=64)

tokenized\_datasets = dataset.map(tokenize\_function, batched=True)

# Eğitim ve değerlendirme veri setlerini oluştur

train\_dataset = tokenized\_datasets["train"]

eval\_dataset = tokenized\_datasets["validation"]

# 4. Eğitim Parametrelerini Ayarlama

training\_args = TrainingArguments(

    output\_dir="./results",

    evaluation\_strategy="epoch",

    logging\_dir="./logs",

    per\_device\_train\_batch\_size=4,  # Küçük batch boyutu

    per\_device\_eval\_batch\_size=4,

    num\_train\_epochs=1,  # Sadece 1 epoch

    logging\_steps=5,  # Daha sık logging (isteğe bağlı)

)

# 5. Performans Değerlendirme Fonksiyonu

def compute\_metrics(p):

    preds = np.argmax(p.predictions, axis=1)

    acc = accuracy\_score(p.label\_ids, preds)

    f1 = f1\_score(p.label\_ids, preds, average="weighted")

    return {"accuracy": acc, "f1": f1}

# 6. Model Eğitimi

trainer = Trainer(

    model=model,

    args=training\_args,

    train\_dataset=train\_dataset,

    eval\_dataset=eval\_dataset,

    compute\_metrics=compute\_metrics,

)

trainer.train()

# 7. Modeli Değerlendirme

eval\_results = trainer.evaluate(tokenized\_datasets["test"])

print("Test Sonuçları:", eval\_results)