Büyük Veri Analizi

Ders 2

Ali Mertcan KOSE Ph.D.
amertcankose@ticaret.edu.tr

İstanbul Ticaret Üniversitesi

Giriş

Bu bölümde, Spark ve Spark Veri Çerçeveleri ile Python API'si PySpark'ı kullanarak nasıl çalışılacağı hakkında daha fazla bilgi edineceğiz. PySpark bize petabayt ölçeğinde verileri işleme olanağı sağlarken, aynı zamanda gerçek zamanlı olarak petabayt ölçeğinde makine öğrenimi (ML) algoritmaları da uygular. Bu bölüm, PySpark'ta Spark Veri Çerçeveleri kullanılarak veri işleme kısmına odaklanacaktır.

dağıtılmış bir veri koleksiyonudur. R ve Python Veri Çerçevelerinden esinlenilmiştir ve arka uçta onları hızlı, optimize edilmiş ve ölçeklenebilir kılan karmaşık optimizasyonlara sahiptir. Veri Çerçevesi API'si, Project Tungsten'in bir parçası olarak geliştirilmiştir ve Spark'ın performansını ve ölçeklenebilirliğini artırmak için tasarlanmıştır. İlk olarak Spark 1.3 ile tanıtılmıştır. Spark Veri Çerçeveleri, selefleri olan RDD'lere göre kullanımı ve işlenmesi çok daha kolaydır. RDD'ler gibi değiştirilemezler ve tembel yüklemeyi desteklerler; bu da bir eylem çağrılmadıkça Veri Çerçeveleri üzerinde hiçbir dönüşüm gerçekleştirilmediği anlamına gelir. Veri Çerçeveleri için yürütme planı Spark tarafından hazırlanır ve bu nedenle daha optimize edilmiştir; bu da Veri Çerçeveleri üzerindeki işlemleri RDD'lere göre daha hızlı hale getirir.

Spark Veri Çerçevelerine Başlarken

Spark DataFrames'i kullanmaya başlamak için öncelikle SparkContext adı verilen bir şey oluşturmamız gerekiyor. SparkContext, dahili hizmetleri yapılandırır ve Spark yürütme ortamından komut yürütmeyi kolaylaştırır.

Spark Veri Çerçevelerine Başlarken

SparkContext'i oluşturmak için aşağıdaki kod parçacığı kullanılır:

```
from pyspark import SparkContext
sc = SparkContext()
```

Spark Veri Çerçevelerine Başlarken

DataFrame'lerle çalışmaya başlamadan önce bir SQLContext oluşturmamız da gerekiyor. Spark'taki SQLContext, Spark içinde SQL benzeri işlevler sağlayan bir sınıftır. SparkContext kullanarak SQLContext oluşturabiliriz:

```
from pyspark.sql import SQLContext sqlc = SQLContext(sc)
```

Spark Veri Çerçevelerine Başlarken

Spark'ta bir DataFrame oluşturmanın üç farklı yolu vardır:

- DataFrame'in şemasını programatik olarak belirleyebilir ve verileri manuel olarak girebiliriz. Ancak, Spark genellikle büyük verileri işlemek için kullanıldığından, bu yöntem küçük test/örnek vakalar için veri oluşturmanın dışında pek işe yaramaz.
- Bir DataFrame oluşturmanın bir diğer yöntemi de Spark'taki mevcut bir RDD nesnesindendir. Bu faydalıdır, çünkü bir DataFrame üzerinde çalışmak, doğrudan RDD'lerle çalışmaktan çok daha kolaydır.
- Ayrıca, bir Spark DataFrame oluşturmak için verileri doğrudan bir veri kaynağından okuyabiliriz. Spark, CSV, JSON, Parquet, RDBMS tabloları ve Hive tabloları dahil olmak üzere çeşitli harici veri kaynaklarını destekler.

Bir Veri Çerçevesinin Şemasını Belirleme

Bu alıştırmada, şemayı manuel olarak belirleyip verileri Spark'a girerek küçük bir örnek Veri Çerçevesi oluşturacağız. Bu yöntemin pratikte pek bir uygulaması olmasa da, Spark Veri Çerçeveleri'ni kullanmaya başlamak için iyi bir başlangıç noktası olacaktır:

1. Gerekli dosyaları içe aktarıyoruz:

```
from pyspark import SparkContext
sc = SparkContext()
from pyspark.sql import SQLContext
sqlc = SQLContext(sc)
```

Bir Veri Çerçevesinin Şemasını Belirleme

2. PySpark modülünden SQL yardımcı programlarını içe aktarın ve örnek Veri Çerçevesinin şemasını belirtin:

```
from pyspark.sql import *
na_schema = Row("Name","Age")
```

3. Belirtilen şemaya göre DataFrame için satırlar oluşturun:

```
row1 = na_schema("Ankit", 23)
row2 = na_schema("Tyler", 26)
row3 = na_schema("Preity", 36)
```

Bir Veri Çerçevesinin Şemasını Belirleme

4. Satırları birleştirerek DataFrame'i oluşturun:

```
na_list = [row1, row2, row3]
df_na = sqlc.createDataFrame(na_list)
type(df_na)
```

5. Şimdi aşağıdaki komutu kullanarak DataFrame'i gösterelim:

```
df_na.show()
```

Mevcut bir RDD'den Veri Çerçevesi Oluşturma

Bu alıştırmada, Spark'ta mevcut bir RDD nesnesinden küçük bir örnek DataFrame oluşturacağız:

1. DataFrame'e dönüştüreceğimiz bir RDD nesnesi oluşturalım:

```
data = [("Ankit",23),("Tyler",26),("Preity",36)]
data_rdd = sc.parallelize(data)
type(data_rdd)
```

2. RDD nesnesini bir DataFrame'e dönüştürün:

```
data_sd = sqlc.createDataFrame(data_rdd)
```

3. Şimdi aşağıdaki komutu kullanarak DataFrame'i gösterelim:

```
data_sd.show()
```

CSV Dosyası Kullanarak Bir Veri Çerçevesi Oluşturma

Bir Veri Çerçevesi oluşturmak için çeşitli veri kaynakları kullanılabilir. Bu alıştırmada, scikit-learn kütüphanesindeki veri kümeleri altında bulunan açık kaynaklı Iris veri kümesini kullanacağız. Iris veri kümesi, 3 Iris çiçeği türünün (Iris Setosa, Iris Virginica ve Iris Versicolor) her biri için 50 kayıt içeren 150 kayıt içeren çok değişkenli bir veri kümesidir.

Veri kümesi, her bir Iris türü için beş öznitelik içerir: taç yaprağı uzunluğu, taç yaprağı genişliği, çanak yaprağı uzunluğu, çanak yaprağı genişliği ve tür. Bu veri kümesini, Spark'a okuyacağımız harici bir CSV dosyasında sakladık:

CSV Dosyası Kullanarak Bir Veri Çerçevesi Oluşturma

1. Databricks web sitesinden PySpark CSV okuyucu paketini indirin ve yükleyin:

```
pyspark -packages com.databricks:spark-csv_2.10:1.4.0
```

2. CSV dosyasındaki verileri Spark DataFrame'e okuyun:

```
{df = sqlc.read.format('com.databricks.spark.csv').\
options(header='true',

inferschema='true').load('iris.csv') \

type(df)}
```

3. Şimdi aşağıdaki komutu kullanarak DataFrame'i gösterelim:

```
df.show(4)
```

Spark Veri Çerçevelerinden Çıktı Yazma

Spark, Spark Veri Çerçevelerinde depolanan verileri yerel bir Pandas Veri Çerçevesine veya CSV gibi harici yapılandırılmış dosya biçimlerine yazma olanağı sağlar. Ancak, bir Spark Veri Çerçevesini yerel bir Pandas Veri Çerçevesine dönüştürmeden önce, verilerin yerel sürücü belleğine sığacağından emin olun. Aşağıdaki alıştırmada, Spark Veri Çerçevesinin bir Pandas Veri Çerçevesine nasıl dönüştürüleceğini inceleyeceğiz.

Bir Spark Veri Çerçevesini Pandas Veri Çerçevesine Dönüştürme

Bu alıştırmada, önceki alıştırmada Iris veri kümesinin önceden oluşturulmuş Spark Veri Çerçevesini kullanacağız ve bunu yerel bir pandas Veri Çerçevesine dönüştüreceğiz. Daha sonra bu Veri Çerçevesini bir CSV dosyasına kaydedeceğiz. Aşağıdaki adımları uygulayın:

1. Aşağıdaki komutu kullanarak Spark Veri Çerçevesini bir pandas Veri Çerçevesine dönüştürün:

```
import pandas as pd
df.toPandas()
```

2. Şimdi pandas DataFrame'i bir CSV dosyasına yazmak için aşağıdaki komutu kullanın:

Spark Veri Çerçevelerini Keşfetme

Spark Veri Çerçevelerinin geleneksel RDD'lere göre sunduğu en büyük avantajlardan biri, veri kullanım ve inceleme kolaylığıdır. Veriler, Veri Çerçevelerinde daha yapılandırılmış bir tablo biçiminde saklanır ve bu nedenle anlaşılması daha kolaydır. Satır ve sütun sayısı gibi temel istatistikleri hesaplayabilir, şemaya bakabilir ve ortalama ve standart sapma gibi özet istatistikleri hesaplayabiliriz.

Temel Veri Çerçevesi İstatistiklerini Görüntüleme

Bu alıştırmada, verilerin ilk birkaç satırının temel DataFrame istatistiklerini ve tüm sayısal DataFrame sütunları ile tek bir DataFrame sütununun özet istatistiklerini göstereceğiz:

1. DataFrame şemasına bakın. Şema, konsolda ağaç biçiminde görüntülenir:

```
df.printSchema()
```

2. Şimdi, Spark DataFrame'in sütun adlarını yazdırmak için aşağıdaki komutu kullanın:

```
df.schema.names
```

Temel Veri Çerçevesi İstatistiklerini Görüntüleme

3. Spark DataFrame'de bulunan satır ve sütun sayısını almak için aşağıdaki komutu kullanın:

```
{##Counting the number of rows in DataFrame \

df.count()#134}

{## Counting the number of columns in DataFrame \

len(df.columns)#5}
```

4. Verilerin ilk n satırını alalım. Bunu head() metodunu kullanarak yapabiliriz. Ancak, verileri daha iyi bir formatta gösterdiği için show() metodunu kullanıyoruz:

```
df.show(4)
```

Temel Veri Çerçevesi İstatistiklerini Görüntüleme

5. Şimdi, Veri Çerçevesindeki tüm sayısal sütunlar için ortalama ve standart sapma gibi özet istatistiklerini hesaplayın:

```
df.describe().show()
```

6. Bir Spark DataFrame'in tek bir sayısal sütununun özet istatistiklerini hesaplamak için aşağıdaki komutu kullanın:

```
f.describe('Sepalwidth').show()
```

Spark Veri Çerçevelerine Başlarken

Bu etkinlikte, önceki bölümlerde öğrenilen kavramları kullanacak ve üç yöntemi de kullanarak bir Spark Veri Çerçevesi oluşturacağız. Ayrıca Veri Çerçevesi istatistiklerini hesaplayacak ve son olarak aynı verileri bir CSV dosyasına yazacağız. Bu etkinlik için herhangi bir açık kaynaklı veri kümesini kullanmaktan çekinmeyin:

1. Şemayı manuel olarak belirterek örnek bir Veri Çerçevesi oluşturun.
2. Mevcut bir RDD'den örnek bir Veri Çerçevesi oluşturun.
3. Verileri bir CSV dosyasından okuyarak örnek bir Veri Çerçevesi oluşturun.

Spark Veri Çerçevelerine Başlarken

4. 3-adımda okunan örnek Veri Çerçevesinin ilk yedi satırını yazdırın.
5. 4-adımda okunan örnek Veri Çerçevesinin şemasını yazdırın.
6. Örnek Veri Çerçevesindeki satır ve sütun sayısını yazdırın.
7. Veri Çerçevesinin ve herhangi 2 ayrı sayısal sütunun özet istatistiklerini yazdırın.
8. Örnek DataFrame'in ilk 7 satırını, alıştırmalarda belirtilen her iki yöntemi de kullanarak bir CSV dosyasına yazın.

Spark DataFrames ile Veri İşleme

Veri manipülasyonu, herhangi bir veri analizi için ön koşuldur. Verilerden anlamlı çıkarımlar elde etmek için öncelikle verileri anlamamız, işlememiz ve düzenlememiz gerekir. Ancak bu adım, veri boyutunun artmasıyla özellikle zorlaşır. Veri ölçeği nedeniyle, filtreleme ve sıralama gibi basit işlemler bile karmaşık kodlama sorunlarına dönüşür. Spark DataFrame'ler, büyük verilerde veri manipülasyonunu çocuk oyuncağı haline getirir. Spark DataFrame'lerde veri manipülasyonu, sıradan Pandas DataFrame'lerde çalışmaya oldukça benzer. Spark DataFrame'lerdeki veri manipülasyon işlemlerinin çoğu, basit ve sezgisel tek satırlık ifadeler kullanılarak yapılabilir. Bu veri manipülasyonu alıştırmaları için, önceki alıştırmalarda oluşturduğumuz Iris veri kümesini içeren Spark DataFrame'i kullanacağız.

Veri Çerçevesinden Sütunları Seçme ve Yeniden Adlandırma

Bu alıştırmada, önce withColumnRenamed metodunu kullanarak sütunu yeniden adlandıracağız ve ardından select metodunu kullanarak şemayı seçip yazdıracağız. Aşağıdaki adımları uygulayın:

1. withColumnRenamed() metodunu kullanarak bir Spark DataFrame'in sütunlarını yeniden adlandırın:

```
df = df.withColumnRenamed('Sepal.Width','Sepalwidth')
```

2. Select yöntemini kullanarak bir Spark Veri Çerçevesinden tek bir sütun veya birden fazla sütun seçin:

```
df.select('Sepalwidth','Sepallength').show(4)
```

Veri Çerçevesine Sütun Ekleme ve Kaldırma

Bu alıştırmada, withColumn metodunu kullanarak veri kümesine yeni bir sütun ekleyeceğiz ve daha sonra drop fonksiyonunu kullanarak bu sütunu kaldıracağız. Şimdi aşağıdaki adımları uygulayalım:

1. withColumn metodunu kullanarak bir Spark Veri Çerçevesine yeni bir sütun ekleyin:

```
df = df.withColumn('Half_sepal_width',
df['Sepalwidth']/2.0)
```

2. Yeni eklenen sütunla birlikte veri setini göstermek için aşağıdaki komutu kullanın:

```
df.show(4)
```

Veri Çerçevesine Sütun Ekleme ve Kaldırma

3. Şimdi, bir Spark Veri Çerçevesindeki bir sütunu kaldırmak için burada gösterilen bırakma yöntemini kullanın:

```
df = df.drop('Half_sepal_width')
```

4. Sütunun kaldırıldığını doğrulamak için veri setini gösterelim:

```
df.show(4)
```

Bir Veri Çerçevesinde Farklı Değerleri Görüntüleme ve Sayma

Bir DataFrame'deki farklı değerleri görüntülemek için distinct().show() metodunu kullanırız. Benzer şekilde, farklı değerleri saymak için distinct().count() metodunu kullanacağız. Farklı değerleri toplam sayımla birlikte yazdırmak için aşağıdaki prosedürleri uygulayın:

1. Select yöntemiyle birlikte distinct yöntemini kullanarak bir Spark DataFrame'in herhangi bir sütunundaki farklı değerleri seçin:

```
df.select('Species').distinct().show()
```

2. Bir Spark Veri Çerçevesinin herhangi bir sütunundaki farklı değerleri saymak için, farklı yöntemle birlikte count yöntemini kullanın:

```
df.select('Species').distinct().count()
```

Bir Veri Çerçevesinin Yinelenen Satırlarını Kaldırma ve Satırlarını Filtreleme

Bu alıştırmada, veri kümesinden yinelenen satırları nasıl kaldıracağımızı ve daha sonra aynı sütunda filtreleme işlemlerini nasıl gerçekleştireceğimizi öğreneceğiz. Şu adımları uygulayın:

1. dropDuplicates() yöntemini kullanarak bir Veri Çerçevesinden yinelenen değerleri kaldırın:

```
df.select('Species').dropDuplicates().show()
```

Bir Veri Çerçevesinin Yinelenen Satırlarını Kaldırma ve Satırlarını Filtreleme

2. Bir veya daha fazla koşul kullanarak bir Veri Çerçevesindeki satırları filtreleyin. Bu birden fazla koşul, Pandas Veri Çerçeveleri için yaptığımız gibi, ve (&) veya | gibi Boole operatörleri kullanılarak Veri Çerçevesine birlikte aktarılabilir:

```
#Filtering using a single condition
df.filter(df.Species=='setosa').show(4)
```

3. Şimdi, sütunu birden fazla koşul kullanarak filtrelemek için aşağıdaki komutu kullanın:

```
df.filter((df.Sepallength>5)&(df.Species=='setosa')).show(4
```

Bir Veri Çerçevesinde Satırları Sıralama

Bu alıştırmada, bir Veri Çerçevesindeki satırları artan ve azalan düzende nasıl sıralayacağımızı inceleyeceğiz. Şu adımları uygulayalım:

1. Bir veya daha fazla koşul kullanarak bir Veri Çerçevesindeki satırları artan veya azalan düzende sıralayın:
   `df.orderBy(df.Sepallength).show(5)`
2. Satırları azalan düzende sıralamak için aşağıdaki komutu kullanın:

```
df.orderBy(df.Sepallength.desc()).show(5)
```

Bir Veri Çerçevesinde Değerleri Toplama

Bir Veri Çerçevesindeki değerleri bir veya daha fazla değişkene göre gruplandırabilir ve ortalama, toplam, sayım ve daha birçok toplu metrik hesaplayabiliriz. Bu alıştırmada, Süsen veri kümesindeki her çiçek türü için ortalama çanak yaprağı genişliğini hesaplayacağız. Ayrıca her tür için satır sayısını da hesaplayacağız:

1. Her tür için ortalama çanak yaprağı genişliğini hesaplamak için aşağıdaki komutu kullanın:

```
df.groupby('Species').agg('Sepalwidth' : 'mean').show()
```

2. Şimdi, aşağıdaki komutu kullanarak her tür için satır sayısını hesaplayalım:

```
df.groupby('Species').count().show()
```

Spark DataFrames ile Veri İşleme

Bu etkinlikte, önceki bölümlerde öğrenilen kavramları kullanarak Iris veri kümesi kullanılarak oluşturulan Spark Veri Çerçevesi'ndeki verileri işleyeceğiz. Bir Spark Veri Çerçevesi'ndeki verilerle çalışma becerimizi test etmek için temel veri işleme adımlarını uygulayacağız. Bu etkinlik için herhangi bir açık kaynaklı veri kümesini kullanmaktan çekinmeyin. Kullandığınız veri kümesinin hem sayısal hem de kategorik değişkenlere sahip olduğundan emin olun:

1. Veri Çerçevesi'nin herhangi beş sütununu yeniden adlandırın. Veri Çerçevesi'nde birden fazla sütun varsa, tüm sütunları yeniden adlandırın.
2. Veri Çerçevesi'nden iki sayısal ve bir kategorik sütun seçin.
3. Kategorik değişkendeki farklı kategori sayısını sayın.
4. İki sayısal sütunu toplayıp çarparak Veri Çerçevesi'nde iki yeni sütun oluşturun.

Spark DataFrames ile Veri İşleme

5. Her iki orijinal sayısal sütunu da silin.
6. Verileri kategorik sütuna göre sıralayın.
7. Kategorik değişkendeki her farklı kategori için toplam sütununun ortalamasını hesaplayın.
8. 7- adımda hesaplanan tüm ortalama değerlerin ortalamasından büyük değerlere sahip satırları filtreleyin.
9. Elde edilen Veri Çerçevesini, yalnızca benzersiz kayıtlar içerdiğinden emin olmak için kopyalayın.

Spark'taki Grafikler

Verileri etkili bir şekilde görselleştirme becerisi son derece önemlidir. Verilerin görsel temsilleri, kullanıcının verileri daha iyi anlamasına ve metin biçiminde fark edilmeyebilecek eğilimleri ortaya çıkarmasına yardımcı olur. Python'da her biri kendi bağlamına sahip çok sayıda grafik türü mevcuttur.

Spark'taki Grafikler

Spark Veri Çerçeveleri için çubuk grafikler, yoğunluk grafikleri, kutu grafikleri ve doğrusal grafikler de dahil olmak üzere bu grafiklerden bazılarını, yaygın olarak kullanılan Python çizim paketleri olan Matplotlib ve Seaborn'u kullanarak inceleyeceğiz. Burada dikkat edilmesi gereken nokta, Spark'ın büyük verilerle ilgilendiğidir. Bu nedenle, çizim yapmadan önce veri boyutunuzun makul olduğundan (yani bilgisayarınızın RAM'ine sığdığından) emin olun. Bu, verileri çizimden önce filtreleyerek, toplayarak veya örnekleyerek elde edilebilir. Küçük olan Iris veri kümesini kullanıyoruz, bu nedenle veri boyutunu azaltmak için herhangi bir ön işleme adımı yapmamıza gerek yok.

Küçük boyutlu Iris veri setini kullandığımız için veri boyutunu küçültmek için herhangi bir ön işleme adımına ihtiyacımız yok.

Çubuk Grafiği Oluşturma

Bu alıştırmada, her tür için mevcut kayıt sayısını bir çubuk grafik kullanarak çizmeye çalışacağız. Önce verileri toplamamız ve her tür için kayıt sayısını saymamız gerekecek. Daha sonra bu toplanan verileri standart bir Pandas Veri Çerçevesine dönüştürebilir ve Matplotlib ve Seaborn paketlerini kullanarak istediğimiz türde grafikler oluşturabiliriz:

1. İlk olarak, her çiçek türü için satır sayısını hesaplayın ve sonucu bir Pandas Veri Çerçevesine dönüştürün:

```
data = df.groupby('Species').count().toPandas()
```

Çubuk Grafiği Oluşturma

2. Şimdi, ortaya çıkan pandas DataFrame'inden bir çubuk grafiği oluşturun:

```
import seaborn as sns
import matplotlib.pyplot as plt
sns.barplot( x = data['Species'], y = data['count'])
plt.xlabel('Species')
plt.ylabel('count')
plt.title('Number of rows per species')
```

Doğrusal Model Grafiği Oluşturma

Bu alıştırmada, iki farklı değişkenin veri noktalarını çizip üzerlerine düz bir çizgi yerleştireceğiz. Bu, iki değişkene doğrusal bir model yerleştirmeye benzer ve iki değişken arasındaki korelasyonları belirlemeye yardımcı olabilir:

1. Pandas DataFrame'inden bir veri nesnesi oluşturun:

```
data = df.toPandas() sns.lmplot(x = "Sepallength", y
= "Sepalwidth", data = data)
```

2. Aşağıdaki komutu kullanarak Veri Çerçevesini çizin:

```
plt.show()
```

KDE Grafiği ve Kutu Grafiği Oluşturma

Bu alıştırmada, bir çekirdek yoğunluk tahmini (KDE) grafiği ve ardından bir kutu grafiği oluşturacağız. Şu talimatları izleyin:

1. İlk olarak, bir değişkenin dağılımını gösteren bir KDE grafiği çizin.

```
import seaborn as sns
data = df.toPandas()
sns.kdeplot(data.Sepalwidth, shade = True)
plt.show()
```

2. Şimdi, aşağıdaki komutu kullanarak Iris veri kümesi için kutu grafiklerini çizin:

```
sns.boxplot(x = "Sepallength", y = "Sepalwidth", data
= data)
plt.show()
```

Spark'taki Grafikler

Bu etkinlikte, verilerimizi farklı türde grafikler kullanarak görsel olarak incelemek için Python'un çizim kütüphanelerini kullanacağız. Bu etkinlik için Kaggle'daki (https://www.kaggle.com/ruiromanini/mtcars) mtcars veri kümesini kullanıyoruz:

1. Jupyter Notebook'a gerekli tüm paketleri ve kütüphaneleri aktarın.
2. Verileri mtcars veri kümesinden Spark nesnesine okuyun.
3. Veri kümenizdeki herhangi bir sürekli sayısal değişkenin ayrık frekans dağılımını bir histogram kullanarak görselleştirin:
4. Veri setindeki kategorilerin yüzdelik payını pasta grafiği kullanarak görselleştirin: