Büyük Veri Analizi

Ders 6

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

İstanbul Ticaret Üniversitesi

Giriş

Bir süredir veri sektöründeyseniz, farklı veri kaynaklarıyla çalışmanın, bunları analiz etmenin ve tüketilebilir iş raporlarında sunmanın zorluğunu anlayacaksınız. Spark'ı Python'da kullanırken, düz dosyalar, JSON formatındaki REST API'leri vb. gibi çeşitli kaynaklardan veri okumanız gerekebilir. Gerçek dünyada, verileri doğru formatta elde etmek her zaman zordur ve veri toplamak için birkaç SQL işlemi gerekir. Bu nedenle, herhangi bir veri bilimcisinin farklı dosya formatlarını ve farklı kaynakları nasıl kullanacağını, temel SQL işlemlerini nasıl gerçekleştireceğini ve bunları tüketilebilir bir formatta nasıl sunacağını bilmesi zorunludur.

Bu bölüm, farklı veri türlerini okumak, üzerinde SQL işlemleri gerçekleştirmek, tanımlayıcı istatistiksel analiz yapmak ve eksiksiz bir analiz raporu oluşturmak için genel yöntemler sunmaktadır. Farklı veri türlerinin PySpark'a nasıl okunacağını anlayarak başlayacak ve ardından bunlar üzerinde çeşitli analizler ve grafikler oluşturacağız.

Spark'ta Farklı Veri Kaynaklarından Veri Okuma

Spark'ın avantajlarından biri, çeşitli veri kaynaklarından veri okuyabilme yeteneğidir. Ancak bu tutarlı değildir ve her Spark sürümünde değişmeye devam eder. Bu bölümün bu kısmında CSV ve JSON formatındaki dosyaların nasıl okunacağı açıklanacaktır.

1. Öncelikle gerekli paketleri Jupyter notebook'a aktaralım:

```
import os
import pandas as pd
import numpy as np
import collections
from sklearn.base import TransformerMixin
import random
import pandas_profiling
```

2. Ardından, gösterildiği gibi gerekli tüm kütüphaneleri içe aktarın:

```
import seaborn as sns
import time
import re
import os
import matplotlib.pyplot as plt
```

Şimdi, veri setimizi daha görünür hale getirecek ve daha yüksek kontrast sağlayacak olan tik temalarını kullanalım:

```
sns.set(style="ticks")
```

Spark'ta Farklı Veri Kaynaklarından Veri Okuma

1. Şimdi aşağıdaki komutu kullanarak çalışma dizinini değiştirin:

```
os.chdir("/Users/svk/Desktop/packt_exercises")
```

2. Spark oturumunu oluşturmak için Spark'a gerekli olan kütüphaneleri içe aktaralım:

```
pyspark.sql'den SparkSession'ı içe aktar
spark = SparkSession.builder.appName('ml-bank').getOrCreate
```

Spark'ta Farklı Veri Kaynaklarından Veri Okuma

3. Şimdi aşağıdaki komutu kullanarak df_csv Spark nesnesini oluşturduktan sonra CSV verilerini okuyalım:

```
df_csv = spark.read.csv('bank.csv', sep=';', header =
True,
inferSchema = True)
```

4. Aşağıdaki komutu kullanarak şemayı yazdırın:

```
df_csv.printSchema()
```

PySpark Nesnesini Kullanarak JSON Verilerini Okuma

JSON verilerini okumak için, SQL bağlamını ayarladıktan sonra
read.json(".json uzantılı dosya adı") fonksiyonunu yazmanız gerekir:

Spark Veri Çerçevesinde SQL İşlemleri

Spark'taki bir DataFrame, satır ve sütunlardan oluşan dağıtılmış bir koleksiyondur. İlişkisel bir veritabanındaki veya bir Excel sayfasındaki bir tabloyla aynıdır. Bir Spark RDD/DataFrame, büyük miktarda veriyi işlemede verimlidir ve ister yapılandırılmış ister yapılandırılmamış olsun, petabaytlarca veriyi işleyebilir.

Spark, Veri Çerçevesini sütunlara ayırarak veriler üzerindeki sorguları optimize eder ve bu da Spark'ın şemayı anlamasına yardımcı olur. En sık kullanılan SQL işlemlerinden bazıları arasında verileri alt kümelere ayırma, birleştirme, filtreleme, belirli sütunları seçme, sütunları silme, tüm boş değerleri silme ve yeni sütunlar ekleme gibi işlemler yer alır.

PySpark'ta Veri Okuma ve SQL İşlemleri Gerçekleştirme

Verilerin özet istatistikleri için, DataFrame'deki tüm sütunlar için sayım, ortalama, standart sapma, maksimum ve minimum hakkında bilgi sağlayacak olan spark_df.describe().show() fonksiyonunu kullanabiliriz.

Örneğin, ele aldığımız veri setinde (banka pazarlama veri seti (https://raw.githubusercontent.com/mertcank1/BDA/refs/heads/main/bank.csv) özet istatistik verileri şu şekilde elde edilebilir:

PySpark'ta Veri Okuma ve SQL İşlemleri Gerçekleştirme

1. Yeni bir Jupyter not defteri oluşturduktan sonra, gerekli tüm paketleri burada gösterildiği gibi içe aktarın:

```
import os
import pandas as pd
import numpy as np
```

2. Şimdi aşağıdaki komutu kullanarak çalışma dizinini değiştirin:

```
os.chdir("/Users/svk/Desktop/packt_exercises")
```

PySpark'ta Veri Okuma ve SQL İşlemleri Gerçekleştirme

3. Spark oturumunu oluşturmak için Spark'a gereken tüm kütüphaneleri içe aktarın:

```
from pyspark.sql import SparkSession
spark = SparkSession.builder.appName('ml-bank').getOrCreate
```

PySpark'ta Veri Okuma ve SQL İşlemleri Gerçekleştirme

4. Spark nesnesini kullanarak CSV dosyasından verileri oluşturun ve okuyun; aşağıdaki şekilde gösterildiği gibi:

```
spark_df = spark.read.csv('bank.csv', sep=';', header
= True, inferSchema= True)
```

5. Şimdi aşağıdaki komutu kullanarak Spark nesnesinin ilk beş satırını yazdıralım:

```
spark_df.head(5)
```

PySpark'ta Veri Okuma ve SQL İşlemleri Gerçekleştirme

6. Önceki çıktı yapılandırılmamış. Yapılandırılmış verileri almak için öncelikle veri türlerini belirleyelim. Her sütunun veri türünü yazdırmak için aşağıdaki komutu kullanın:

```
spark_df.printSchema()
```

7. Şimdi elimizdeki veriler hakkında net bir fikir edinmek için toplam satır ve sütun sayısını ve adlarını hesaplayalım:

```
spark_df.count()

len(spark_df.columns), spark_df.columns
```

PySpark'ta Veri Okuma ve SQL İşlemleri Gerçekleştirme

8. Aşağıdaki komutu kullanarak DataFrame'in özet istatistiklerini yazdırın:

```
spark_df.describe().show()
```

Bir Veri Çerçevesinden birden fazla sütun seçmek için spark_df.select('col1','c
'col3') fonksiyonunu kullanabiliriz. Örneğin, aşağıdaki komutu kullanarak bakiye ve y sütunlarından ilk beş satırı seçelim:

```
spark_df.select('balance','y').show(5)
```

PySpark'ta Veri Okuma ve SQL İşlemleri Gerçekleştirme

9. İki değişken arasındaki ilişkiyi seviye sıklıkları açısından belirlemek için çapraz tablo kullanılabilir. İki sütun arasında çapraz tablo elde etmek için spark_df.crosstab('col1', col2) fonksiyonunu kullanabiliriz. Çapraz tablo, iki kategorik değişken arasında gerçekleştirilir, sayısal değişkenler arasında değil:

```
spark_df.crosstab('y', 'marital').show()
```

PySpark'ta Veri Okuma ve SQL İşlemleri Gerçekleştirme

10. Şimdi veri setine yeni bir sütun ekleyelim:

```
#sample sets
sample1 = spark_df.sample(False, 0.2, 42)
sample2 = spark_df.sample(False, 0.2, 43)

#train set
train = spark_df.sample(False, 0.8, 44)
train.withColumn('balance_new', train.balance /2.0).
select('balance','balance_new').show(5)
```

11. Aşağıdaki komutu kullanarak yeni oluşturulan sütunu silin:

```
train.drop('balance_new)
```

İki Veri Çerçevesini Oluşturma ve Birleştirme

Bu alıştırmada, UCI Makine Öğrenimi Deposu'ndan banka pazarlama verilerini (https://archive.ics.uci.edu/ml/datasets/bank+marketing) çıkarıp kullanacağız. Amaç, PySpark kullanarak bir Spark Veri Çerçevesi üzerinde birleştirme işlemleri gerçekleştirmektir. Veriler, Portekizli bir bankacılık kuruluşunun doğrudan pazarlama kampanyalarıyla ilgilidir. Pazarlama kampanyaları telefon görüşmelerine dayanıyordu. Genellikle, ürünün (banka vadeli mevduat) abone olup olmayacağına (evet) veya abone olmayacağına (hayır) erişmek için aynı müşteri için birden fazla kişiyle iletişime geçmek gerekiyordu. Şimdi, mevcut banka pazarlama verilerinden iki Veri Çerçevesi oluşturalım ve bunları birincil anahtar temelinde birleştirelim:

İki Veri Çerçevesini Oluşturma ve Birleştirme

1. Öncelikle Jupyter not defterine gerekli başlık dosyalarını aktaralım:

```
import os
import pandas as pd
import numpy as np
import pyspark
```

2. Şimdi aşağıdaki komutu kullanarak çalışma dizinini değiştirin:

```
os.chdir("/Users/svk/Desktop/packt_exercises")
```

3. Spark oturumunu oluşturmak için Spark'a gereken tüm kütüphaneleri içe aktarın:

```
from pyspark.sql import SparkSession
spark = SparkSession.builder.appName('ml-bank').getOrCreate
```

İki Veri Çerçevesini Oluşturma ve Birleştirme

4. Aşağıdaki komutu kullanarak CSV dosyalarındaki verileri bir Spark nesnesine okuyun:

```
spark_df = spark.read.csv('bank.csv', sep=';', header
= True, inferSchema= True)
```

5. Spark nesnesinden ilk beş satırı yazdırın:

```
spark_df.head(5)
```

6. Şimdi, birincil anahtarı (ID) kullanarak iki DataFrame'i birleştirmek için, önce onu iki DataFrame'e bölmemiz gerekecek.

İki Veri Çerçevesini Oluşturma ve Birleştirme

7. İlk olarak ID sütunu olan yeni bir DataFrame ekleyin:

```
from pyspark.sql.functions import monotonically_increasing_
train_with_id = spark_df.withColumn("ID", monotonically_inc
```

8. Ardından ID2 adında başka bir sütun oluşturun:

```
train_with_id = train_with_id.withColumn('ID2',
train_with_id.ID)
```

9. Aşağıdaki komutu kullanarak DataFrame'i bölün:

```
train_with_id1 = train_with_id.drop('balance', "ID2")
train_with_id2 = train_with_id.select('balance',
"ID2")
```

İki Veri Çerçevesini Oluşturma ve Birleştirme

10. Şimdi train_with_id2'nin ID sütun adlarını değiştirin:

```
train_with_id2 = train_with_id2.withColumnRenamed("ID2",
"ID")
```

11. Aşağıdaki komutu kullanarak train_with_id1 ve train_with_id2'yi birleştirin:

```
train_merged = train_with_id1.join(train_with_id2,
on=['ID'], how='left_ outer')
```

Veri Çerçevesini Alt Kümeleme

Bu alıştırmada, UCI Makine Öğrenimi Deposu'ndan (https://archive.ics. uci.edu/ml/datasets/bank+marketing) banka pazarlama verilerini çıkarıp kullanacağız.

Amaç, Spark Veri Çerçevesi üzerinde PySpark kullanarak filtreleme/alt kümeleme işlemleri gerçekleştirmektir. Banka pazarlama verilerinde bakiyenin 0'dan büyük olduğu Veri Çerçevesini alt kümeye ayıralım:

```
import os
import pandas as pd
import numpy as np
import pyspark
```

Veri Çerçevesini Alt Kümeleme

2. Şimdi aşağıdaki komutu kullanarak çalışma dizinini değiştirin:

```
os.chdir("/Users/svk/Desktop/packtunderline exercises")
```

3. Spark oturumunu oluşturmak için Spark'a gereken tüm kütüphaneleri içe aktarın:

```
from pyspark.sql import SparkSession
spark = SparkSession.builder.appName('ml-bank').getOrCreate
```

Veri Çerçevesini Alt Kümeleme

4. Şimdi, aşağıdaki komutu kullanarak CSV verilerini bir Spark nesnesi olarak okuyun:

```
spark_df = spark.read.csv('bank.csv', sep=';', header
= True, inferSchema= True)
```

5. Veri Çerçevesini alt kümelere ayırmak ve filtrelemek için SQL sorguları çalıştıralım:

```
train_subsetted = spark_df.filter(spark_df.balance >
0.0)
pd.DataFrame(train_subsetted.head(5))
```

İstatistiksel Ölçümler Oluşturma

Python, istatistiksel modüllere sahip genel amaçlı bir dildir. Sayısal değişkenler için veri dağılımının belirlenmesi, bir korelasyon matrisi oluşturulması, kategorik değişkenlerdeki seviyelerin sıklığı ve tanımlayıcı mod gibi tanımlayıcı analizler gibi birçok istatistiksel analiz Python'da gerçekleştirilebilir.

Veri dağılımını belirlemek ve normalleştirmek, doğrusal regresyon ve destek vektör makineleri gibi parametrik modeller için önemlidir. Bu algoritmalar, verilerin normal dağılıma sahip olduğunu varsayar. Veriler normal dağılıma sahip değilse, bu durum verilerde sapmaya yol açabilir. :

İstatistiksel Ölçümler Oluşturma

Tanımlanan değişkenler daha sonra Yeo-Johnson veya Box-Cox yöntemi kullanılarak normalleştirilir. Özelliklerin önemini belirlemek, tahmin tekniklerinin kullanıldığı bir veri bilimi projesinde önemlidir. Bu, önemli değişkenleri belirlemek için çeşitli istatistiksel teknikler kullanıldığından, genel olarak istatistiksel analiz kapsamına girer. Burada kullanılan yöntemlerden biri, değişken önem analizi için kapsamlı bir RandomForest algoritması olan Boruta'dır. Bunun için BorutaPy paketini kullanacağız:

Plotly Kullanarak Görselleştirme Oluşturma

Bu etkinlikte, UCI Makine Öğrenimi Deposu'ndan banka pazarlama verilerini çıkarıp kullanacağız. Amaç, Python'da Plotly kullanarak görselleştirmeler oluşturmaktır.

Plotly kullanarak görselleştirmeyi oluşturmak için aşağıdaki adımları uygulayın:

1. Gerekli kitaplıkları ve paketleri Jupyter not defterine aktarın.
2. Plotly'nin veri görselleştirmesini görselleştirmesi için gerekli kitaplıkları aktarın:

```
import plotly.graph_objs as go
from plotly.plotly import iplot
import plotly as py
```

Plotly Kullanarak Görselleştirme Oluşturma

3. bank.csv dosyasındaki verileri Spark DataFrame'e okuyun.
4. Sisteminizde kullandığınız Plotly sürümünü kontrol edin. Güncellenmiş sürümü çalıştırdığınızdan emin olun. pip install plotly –upgrade komutunu kullanın ve ardından aşağıdaki kodu çalıştırın:

```
from plotly import _version_ from plotly.offline
import download_plotlyjs, init_ notebook_mode, plot,
iplot
```

5. Şimdi Plotly kullanarak grafikleri çizmek için gerekli kütüphaneleri içe aktarın:

```
import plotly.plotly as py
import plotly.graph_objs as go
from plotly.plotly import iplot
init_notebook_mode(connected=True)
```

Plotly Kullanarak Görselleştirme Oluşturma

6. Aşağıdaki komutta Plotly kimlik bilgilerini burada gösterildiği gibi ayarlayın:

```
plotly.tools.set_credentials_file(username='Your_Username',
api_key='Your_ API_Key')
```