Büyük Veri Analizi

Ders 3

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

İstanbul Ticaret Üniversitesi

Giriş

Önceki bölümde, Spark Veri Çerçevelerinin temel kavramlarını öğrendik ve bunları büyük veri analizinde nasıl kullanabileceğimizi gördük. Bu bölümde, bir adım daha ileri giderek verilerdeki eksik değerlerin nasıl ele alınacağını ve Spark Veri Çerçeveleri ile korelasyon analizini öğreneceğiz. Bu kavramlar, makine öğrenimi ve keşifsel veri analizi için veri hazırlamada bize yardımcı olacak. Okuyucuya bir bağlam sağlamak için bu kavramları kısaca ele alacağız, ancak odak noktamız bunların Spark Veri Çerçeveleri ile uygulanması. Bu bölümdeki alıştırmalar için de önceki bölümde kullandığımız aynı Iris veri kümesini kullanacağız. Ancak Iris veri kümesinde eksik değer bulunmadığından, orijinal veri kümesinden Sepallength sütunundan iki girişi ve Petallength sütunundan bir girişi rastgele kaldırdık. Böylece, artık eksik değerleri olan bir veri kümemiz var ve PySpark kullanarak bu eksik değerleri nasıl ele alacağımızı öğreneceğiz. Ayrıca, Iris veri setindeki değişkenler arasındaki korelasyona, korelasyon katsayılarını ve korelasyon matrisini hesaplayarak bakacağız.

Giriş

Egzersizlere başlamadan önce aşağıdaki adımların izlenmesi gerekmektedir:

1. Jupyter not defterine gerekli tüm modülleri ve paketleri aktarın:

```
import findspark
findspark.init()
import pyspark
import random
```

2. Now, use the following command to set up SparkContext:

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

3. Similarly, use the following command to set up SQLContext in the Jupyter notebook:

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

4. Read the Iris dataset from the CSV file into a Spark DataFrame:

```
df = sqlc.read.format('com.databricks.spark.csv').options(h
= 'true',
inferschema = 'true').load('/Users/iris.csv')
```

5. The output of the preceding command is as follows:

```
df.show(5)
```

Kayıp Gözlemler

Kendilerine herhangi bir değer atanmamış veri girişlerine eksik değerler denir. Gerçek dünyada, verilerde eksik değerlerle karşılaşmak yaygındır. Değerler, sistemin/yanıtlayıcının yanıt vermemesi, veri bozulması ve kısmi silme gibi çok çeşitli nedenlerle eksik olabilir.

Bazı alanların eksik değerler içerme olasılığı diğerlerinden daha yüksektir. Örneğin, anketlerden toplanan gelir verilerinde, insanların gelirlerini açıklamak istememeleri nedeniyle eksik değerler bulunma olasılığı yüksektir. Bununla birlikte, bu, veri analitiği dünyasının en büyük sorunlarından biridir.

Kayıp Gözlemler

Eksik veri yüzdesine bağlı olarak, eksik değerler veri hazırlama ve keşifsel analizde önemli bir zorluk oluşturabilir. Bu nedenle, veri analizine başlamadan önce eksik veri yüzdesini hesaplamak önemlidir.

Aşağıdaki alıştırmada, PySpark DataFrame'lerindeki eksik değer girişlerinin sayısını nasıl tespit edip hesaplayacağımızı öğreneceğiz.

Bir Veri Çerçevesinde Eksik Değerleri Sayma

Bu alıştırmada, PySpark DataFrame sütunundaki eksik değerlerin nasıl sayılacağını öğreneceğiz:

1. Spark DataFrame'de eksik değerler olup olmadığını kontrol etmek için aşağıdaki komutu kullanın:

```
from pyspark.sql.functions import isnan, when, count,
col
df.select([count(when(isnan(c) | col(c).isNull(),
c)).alias(c) for c in df.columns]).show()
```

2. Şimdi, PySpark DataFrame df nesnesine yüklenen Iris veri kümesinin Sepallength sütunundaki eksik değerleri sayacağız:

```
df.filter(col('Sepallength').isNull()).count()
```

Tüm Veri Çerçevesi Sütunlarındaki Eksik Değerleri Sayma

Bu alıştırmada, bir PySpark DataFrame'in tüm sütunlarında bulunan eksik değerleri sayacağız:

1. İlk olarak, burada gösterildiği gibi gerekli tüm modülleri içe aktarın:

```
from pyspark.sql.functions import isnan, when, count, col
```

2. Şimdi aşağıdaki komutu kullanarak verileri gösterelim:

```
df.select([count(when(isnan(i) | col(i).isNull(), i)).alias(i) for i in df.columns]).show()
```

Çıktı, PySpark DataFrame'deki Seapllength sütununda 2 eksik giriş ve Petallength sütununda 1 eksik giriş olduğunu gösteriyor.

Tüm Veri Çerçevesi Sütunlarındaki Eksik Değerleri Sayma

3. Basit bir yol, her sütun için eksik olmayan değerlerin sayısını ve bir dizi başka özet istatistiği veren describe() fonksiyonunu kullanmaktır. Not defterinde aşağıdaki komutu çalıştıralım:

```
df.describe().show(1)
```

Gördüğümüz gibi, Sepallength sütununda 148 eksik olmayan değer bulunmaktadır; bu da 2 eksik girdi olduğunu, Petallength sütununda ise 149 eksik olmayan değer bulunduğunu göstermektedir; bu da 1 eksik girdi olduğunu göstermektedir.

Aşağıdaki bölümde, DataFrame'den eksik değerlerin nasıl bulunacağını inceleyeceğiz.

Veri Çerçevesinden Eksik Değer Kayıtlarını Getirme

Aşağıdaki kodu kullanarak PySpark DataFrame'deki eksik değer girişlerini içeren kayıtları da filtreleyebiliriz:

```
df.where(col('Sepallength').isNull()).show()
```

Show fonksiyonu, bir PySpark DataFrame'in ilk 20 kaydını görüntüler. Sepallength sütununda yalnızca iki kayıt eksik olduğundan, burada yalnızca iki kayıt elde ederiz.

Spark Veri Çerçevelerinde Eksik Değerlerin İşlenmesi

Eksik değer işleme, veri biliminin karmaşık alanlarından biridir. Eksik veri türüne ve mevcut iş kullanım senaryosuna bağlı olarak, eksik değerleri işlemek için kullanılan çeşitli teknikler vardır. Bu yöntemler, basit mantık tabanlı yöntemlerden regresyon ve KNN gibi gelişmiş istatistiksel yöntemlere kadar uzanır. Ancak, eksik değerleri ele almak için kullanılan yöntemden bağımsız olarak, eksik değer verileri üzerinde aşağıdaki iki işlemden birini gerçekleştireceğiz:

- Verilerden eksik değerlere sahip kayıtları kaldırma
- Eksik değer girişlerini sabit bir değerle yükleme

Bu bölümde, bu iki işlemin de PySpark DataFrame'leri ile nasıl yapılacağını inceleyeceğiz.

Eksik Değerlere Sahip Kayıtları Bir Veri Çerçevesinden Kaldırma

Bu alıştırmada, PySpark DataFrame için eksik değer girişleri içeren kayıtları kaldıracağız. Aşağıdaki adımları uygulayalım:

1. Belirli bir sütundaki eksik değerleri kaldırmak için aşağıdaki komutu kullanın:

```
df.select('Sepallength').dropna().count()
```

Önceki kod, Sepallength sütununda eksik girişler içeren iki kayıt PySpark DataFrame'den kaldırıldığından, çıktı olarak 148 döndürecektir.

2. PySpark DataFrame'den herhangi bir sütun için eksik değer girişi içeren tüm kayıtları kaldırmak için aşağıdaki komutu kullanın:

```
df.dropna().count()
```

Eksik Değerlere Sahip Kayıtları Bir Veri Çerçevesinden Kaldırma

Alıştırma 2: Tüm Veri Çerçevesi Sütunlarındaki Eksik Değerleri Sayma'da gördüğümüz gibi, Veri Çerçevesi'nde eksik değerlere sahip 3 kayıt vardı: Sepallength sütunu için eksik girişlere sahip iki kayıt ve Petallength sütunu için eksik girişe sahip bir kayıt. Önceki kod, üç kaydı da silerek PySpark Veri Çerçevesi'nde 147 tam kayıt döndürüyor.

Bu alıştırmada, PySpark DataFrame sütunundaki eksik değer girişlerini sabit bir sayısal değerle değiştireceğiz.

Bir Veri Çerçevesi Sütununda Eksik Değerleri Sabitle Doldurma

DataFrame'imizin iki sütununda eksik değerler var: Sepallength ve Petallength:

1. Şimdi, bu iki sütundaki eksik değer girişlerini sabit bir sayısal değer olan 1 ile değiştirelim:

```
y = df.select('Sepallength','Petallength').fillna(1)
```

2. Şimdi, yeni oluşturduğumuz y veri çerçevesindeki eksik değerleri sayalım.

```
y.select([count(when(isnan(i) | col(i).isNull(),
i)).alias(i) for i in y.columns]).show()
```

Bazen, DataFrame'deki tüm eksik değerleri tek bir sabit değerle değiştirmek isteriz.

Bir Veri Çerçevesi Sütununda Eksik Değerleri Sabitle Doldurma

3. PySpark DataFrame'deki tüm eksik değerleri sabit bir sayısal değer olan 1 ile değiştirmek için aşağıdaki komutu kullanın:

Yeni veri çerçevesinde eksik değer olmamalıdır:

```
z = df.fillna(1)
```

4. Şimdi, yeni oluşturduğumuz z Veri Çerçevesindeki eksik değerleri sayalım. Yeni Veri Çerçevesinde eksik değer olmamalıdır:

```
z.select([count(when(isnan(k) | col(k).isNull(),
k)).alias(k) for k in
z.columns]).show()
```

Korelasyon

Korelasyon, iki sayısal değişken arasındaki ilişki düzeyinin istatistiksel bir ölçüsüdür. İki değişkenin birbiriyle ne kadar yakın ilişkili olduğuna dair bir fikir verir. Örneğin, yaş ve gelir oldukça yakın ilişkili değişkenlerdir. Ortalama gelirin belirli bir eşik değer aralığında yaşla birlikte arttığı gözlemlenmiştir. Dolayısıyla, yaş ve gelirin birbiriyle pozitif korelasyona sahip olduğunu varsayabiliriz.

Bu ilişkiyi hesaplamak için kullanılan en yaygın metrik, yaygın olarak Pearson korelasyon katsayısı veya kısaca korelasyon katsayısı olarak bilinen Pearson Ürün-Moment Korelasyonu'dur. Adını mucidi Karl Pearson'dan almıştır.

Korelasyon

Pearson korelasyon katsayısı, iki değişkenin kovaryansının standart sapmalarının çarpımına bölünmesiyle hesaplanır. Korelasyon değeri -1 ile +1 arasındadır; 1 veya -1'e yakın değerler güçlü ilişkiyi, 0'a yakın değerler ise zayıf ilişkiyi gösterir. Katsayının (+, -) işareti, ilişkinin pozitif (her iki değişken birlikte artar/azalır) veya negatif (tersi) olduğunu gösterir.

Korelasyon, istatistiksel analizde büyük önem taşır, çünkü verileri açıklamaya yardımcı olur ve bazen değişkenler arasındaki tahmini ilişkileri vurgular. Bu bölümde, PySpark'ta değişkenler arasındaki korelasyonun nasıl hesaplanacağını ve bir korelasyon matrisinin nasıl hesaplanacağını öğreneceğiz.

Korelasyonun Hesaplanması

Bu alıştırmada, iki sayısal değişken arasındaki Pearson korelasyon katsayısının değerini ve PySpark DataFrame'imizin tüm sayısal sütunları için bir korelasyon matrisini hesaplayacağız. Korelasyon matrisi, tüm sayısal sütunların birbirleriyle olan korelasyonunu görselleştirmemize yardımcı olur:

1. İki değişken arasındaki korelasyonu hesaplamak için aşağıdaki adımları izleyin:

```
df.corr('Sepallength', 'Sepalwidth')
```

2. İlgili modülleri burada gösterildiği gibi içe aktarın:

```
from pyspark.mllib.stat import Statistics import
pandas as pd
```

Korelasyonun Hesaplanması

3. Aşağıdaki komutla verilerdeki eksik değerleri kaldırın:

```
z = df.fillna(1)
```

4. Korelasyon matrisini hesaplamadan önce sayısal olmayan sütunları kaldırmak için aşağıdaki komutu kullanın:

```
a = z.drop('Species')
```

5. Şimdi aşağıdaki komutu kullanarak korelasyon matrisini hesaplayalım:

```
features = a.rdd.map(lambda row:  row[0:])
correlation_matrix = Statistics.corr(features, method="pea
```

Korelasyonun Hesaplanması

6. Matrisi kolay görselleştirme için bir Pandas Veri Çerçevesine dönüştürmek üzere aşağıdaki komutu çalıştırın:

```
correlation_df = pd.DataFrame(correlation_matrix)
```

7. Pandas DataFrame'in dizinlerini, orijinal PySpark DataFrame'deki sütun adlarıyla yeniden adlandırın:

```
correlation_df.index, correlation_df.columns = a.columns,
a.columns
```

8. Şimdi pandas DataFrame'ini aşağıdaki komutla görselleştirin:

```
correlation_df
```

PySpark ile Eksik Değer İşleme ve Korelasyon Analizi DataFrames

Bu etkinlikte, Iris veri kümesindeki eksik değerleri tespit edip işleyeceğiz. Ayrıca, korelasyon matrisini hesaplayacak ve güçlü korelasyonlar gösteren değişkenleri birbirleriyle çizip grafiğe doğrusal bir çizgi yerleştirerek doğrulayacağız:

1. Jupyter not defterine paketleri ve kütüphaneleri içe aktarma başlangıç prosedürünü gerçekleştirin.
2. SparkContext ve SQLContext'i kurun.
3. CSV dosyasındaki verileri bir Spark nesnesine okuyun:
4. Sepallength sütunundaki eksik değerleri sütun ortalamalarıyla doldurun.

PySpark ile Eksik Değer İşleme ve Korelasyon Analizi DataFrames

5. Veri kümesi için korelasyon matrisini hesaplayın. Gerekli modülleri içe aktardığınızdan emin olun.
6. PySpark DataFrame'den String sütunlarını kaldırın ve Spark DataFrame'de korelasyon matrisini hesaplayın.
7. Korelasyon matrisini bir Pandas DataFrame'e dönüştürün:
8. Gerekli modülleri yükleyin ve güçlü pozitif korelasyon gösteren değişken çiftlerini çizmek için verileri çizin ve bunlara doğrusal bir çizgi yerleştirin.

Bu, x = "Petallength", y = "Petalwidth" için grafiktir: