Prof. Dr.-Ing. Ingo Elsen
Big Data Analytics I

Praktikum

Liebe Studierende,

Wenn Sie jeweils eine Aufgabe vollständig erledigt haben, laden Sie bitte Ihre Lösung als jupyter notebook <Name>_<MatrNr>_BDAI_<Aufgabe>.ipynb in Ilias hoch, damit ich sie testen kann. Wenn das funktioniert, bekommen Sie das Testat. Sie sehen das in Ilias, wenn ich die Abgabe auf „bestanden“setze.

Die Aufgaben 2 und 3 müssen am Cluster realisiert werden[1]. Mitunter erzeugen Sie bei der Bearbeitung Artefakte, die für die Bewertung der Abgabe herangezogen werden können. Das Lösen der Aufgaben ist nützlich, da hier der Stoff zu den einzelnen Werkzeugen aus der Vorlesung vertieft wird.

Viel Erfolg und freundliche Grüße
Ingo Elsen

Aufgabe 1:

a) Arbeiten Sie bitte die Kapitel 1-3 in „Data Science from Scratch“ (bzw. der deutschen Ausgabe) durch! Machen Sie sich mit der Entwicklungsumgebung `spyder` und `jupyter`, sowie der für `jupyter` verwendeten Markup-Sprache („Markdown“) vertraut.

b) Schreiben Sie ein Programm, dass einen Zahlzähler als Map-Reduce mittels der entsprechenden Python-Funktionen implementiert. Dazu liest das Programm eine lokale Textdatei `P.csv` ein und gibt die Häufigkeit der einzelnen Zahlen tabellarisch aus. Nutzen Sie dazu die entsprechenden Python Funktionen! Stellen Sie dabei sicher, dass nur Zahlen erfasst werden - die Datei könnte Fehler beinhalten.

c) Erstellen Sie ein Histogramm der Zahlenhäufigkeiten so, dass auf der x-Achse die Zahlen der Größe nach angeordnet sind!

Aufgabe 2:

Machen Sie sich mit Apache Drill vertraut[2]. Ein Notebook wird im JupyterLab bereitgestellt, das Ihnen beim Einstieg helfen kann.

a) Verschaffen Sie sich einen Überblick über eine Menge von Sensordaten, die Sie im Drill unter `dfs.bda1.`co2_data.tsv`` finden. Beantworten Sie die folgenden Fragen, indem Sie die Queries für Drill SQL ausführen:

[1] Damit Sie diese Resource nutzen können, müssen sich per VPN an das FH-Netzwerk angeschlossen sein. Prüfen Sie frühzeitig, dass Sie sich per Cisco Anyconnect verbinden können

[2] Hinweis: Nutzen Sie dafür das bereitgestellte Drill-Cluster. Sie können es via PyDrill im JupyterHub auf dem Server Jupiter erreichen

a1) Wieviele verschiedene Sensoren (angegeben im Feld `source`) enhält die Datenmenge?

a2) Wieviele Datenpunkte je Sensor liegen vor?

a3) Was ist der höchste, und was der niedrigste Temperaturwert?

a4) Was ist der durchschnittliche Co2-Wert je Sensor?

b) Bereiten Sie per SELECT Query die Werte in einzelnen Spalten so vor, dass sie sinnvolle Datentypen aufweisen:

b1) Sowohl *humidity*, *temperature* als auch *co2* sollen als auf zwei Nachkommastellen gerundete Fließkommazahlen verfügbar sein[3].

b2) Die erste Spalte gibt den Zeitstempel als Unix Epoch mit Mikrosekunden an. Machen Sie daraus einen Drill Timestamp[4].

c) Überführen Sie die zuvor erzeugte Query in eine wiederverwendbare View:

c1) Die View sollte Ihnen die Daten als benannte Spalten im gewünschten Datenformat anbieten und den ersten Eintrag (die alte Kopfzeile) überspringen, so dass nur noch Daten zurückgegeben werden.

c2) Die View muss unter dem Pfad `dfs.tmp.`co2_view_<IhrKürzel>`` abgelegt werden. Prüfen Sie, ob die View tatsächlich funktioniert und geben Sie das `CREATE VIEW` Statement mit ab.

c3) Demonstrieren Sie die Nutzung Ihrer View, indem Sie die fünf wärmsten Zeitpunkte (Temperatur, Zeitpunkt und Sensor) jedes Sensors ausgeben lassen.

Aufgabe 3:

Sie finden in der JupyterLab Instanz ein Notebook `Pyspark Kickstart.ipynb`. Nutzen Sie Apache Spark für diese Aufgabe.

a) Laden Sie mit Pyspark die Datei `hdfs://149.201.88.42:9000/data/bda1/co2_data.tsv`[5] und überführen Sie sie in einen Dataframe.

a1) Zeigen Sie, wie Sie die Kopfzeile aus der Datei für das Parsen in den DataFrame überspringen

a2) Finden Sie eine Lösung dafür, die optimalen Datentypen verwenden zu können[6]. Gelöst ist die Aufgabe, wenn mit SparkSQL Timestamp, Float und String verwendet wurden.

b) Führen Sie Schema-On-Read mit Spark SQL durch und machen Sie zu **Aufgabe 2b** äquivalente Abfragen auf diesem Schema.

c) Spark SQL ist dann sinnvoll, wenn das Schema vorliegt. Es ist aber nicht zwingend nötig. Erarbeiten Sie eine Lösung, die ohne Schema auskommt und auf Spark SQL verzichtet. Schreiben Sie ein Pyspark Script, das mittels RDD[7] die Antworten auf

[3] SQL kennt standardmäßig die Methode `CAST` und Drill bietet darüberhinaus `CONVERT_TO` und `CONVERT_FROM` an. Sie können darüber in der Dokumentation lesen. Nutzen Sie einen geeigneten Weg.

[4] Die Dokumentation von Drill bietet unter Anderem einen Hinweis auf die nützliche Funktion `TO_TIMESTAMP()`. Es kann sein, dass Sie einen Fehler erhalten: Finden Sie eine Lösung für das Problem.

[5] Sie können nur von der JupyterHub Instanz BDA1 auf dieses Dateisystem zugreifen. Kopieren Sie die Datei nicht sondern gehen Sie davon aus, dass sich die Daten im entfernten System ändern könnten.

[6] Spark bietet Möglichkeiten, das Lesen und Schreiben von Dateien zu parametrisieren.

[7] Zentrales Konzept für Spark: Das Resilient Distributed Dataset

die Fragen aus Aufgabe 2a) liefert.

Aufgabe 4:

a) Erarbeiten Sie sich das nötige Wissen zum Einlesen von Daten in einen pandas Dataframe aus „Python for Data Analysis“ (bzw. der deutschen Ausgabe).

b) Laden sie den Iris-Datensatz in einen Dataframe
(Datei: `datasets.zip/datasets/iris/data.all`) und drucken Sie grundlegende Statistiken zu den Daten (wie geht das einfach mit einem Dataframe?)!

c) Visualisieren Sie den Datensatz

- Als Scattermatrix (mittels des python Paketes `matplotlib`)
- Als Boxplots für die einzelnen Spalte des Datensatzes

d) Welche Schlüsse können Sie aus den Daten und deren Visualisierungen hinsichtlich einer möglichen Klassifikation der Daten ziehen?