Laborbericht im Modul  
Data Science and Big Data

W3M20018.1

Gruppe 3:

Manmeet Kaur (CAS-TMINF20-W),  
Daniel Kirste (CAS-WWB20-W),  
Felix Öchnser (CAS-TMIE19-W),  
Rouven Schnapka (CAS-WMWI20-W),  
Mario Scholz (CAS-WMWI20-S)

Datum: 23.12.2020

Link zum Repository: <https://github.com/dkirste/datascience2020>

# Big Data Prototyp am Beispiel eines Webshops

## Aufgabenstellung

Das Ziel dieser Gruppenaufgabe war es, eine Anwendung zu entwickeln, wie sie im Rahmen der Vorlesung stückweise aufgebaut wurde. Dazu sollte eine konkrete Anwendungsidee entwickelt werden, welche die in der Vorlesung vorgestellten konkreten Implementierungen verwendet und auf einer Lambda- oder Kappa-Architektur aufbaut. Die Wahl der passenden Architektur ist dabei eine zu treffende Designentscheidung.

## Vorstellung Use-Case

Der vorliegende Use-Case befasst sich mit der Betrachtung eines Warenkorbs (eng. shopping cart) eines fiktiven Web-Shops für Elektronikartikel. Hierbei sollen die hinzugefügten Produkte über eine geeignete Architektur erfasst und verarbeitet werden. Die zugrundeliegende Architektur wird im nachfolgenden Abschnitt näher aufgegriffen und dargestellt.

## Aufbau Architektur

Der beschriebene Use-Case wurde mithilfe einer Kappa-Architektur realisiert. Dabei kommen verschiedene Komponenten zum Einsatz, welche in Interaktion zueinanderstehen.  Eine Übersicht des Zusammenspiels der einzelnen Komponenten, kann nachfolgender Abbildung entnommen werden.

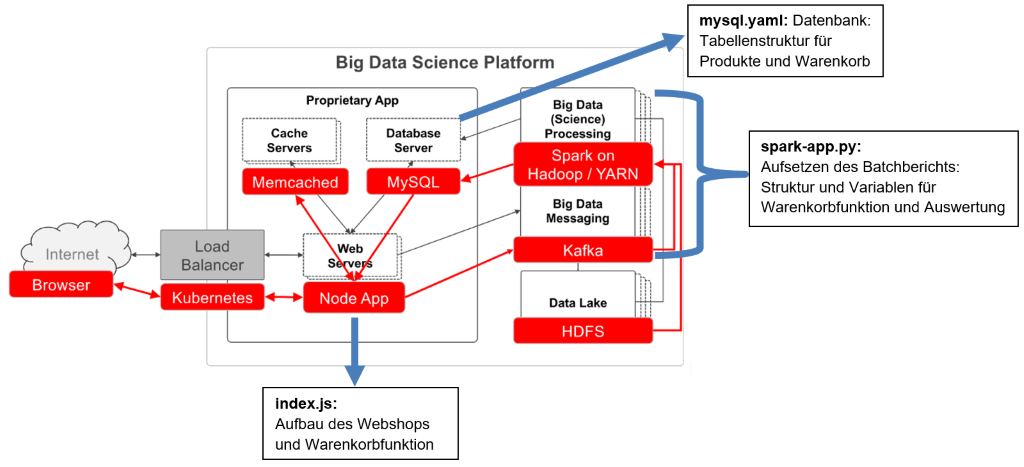


Abbildung 1 Big Data Science Platform

Durch das Hinzufügen eines Artikels zu dem Warenkorb (Button “Buy this Article“) auf der Homepage des Web-Shops sendet der Browser die Informationen an einen Load-Balancer, welcher in der vorliegenden Anwendung durch Kubernetes realisiert wurde. Über den Load-Balancer gelangt die Anfrage zur JavaScript-basierten Web-Applikation, welche die Verteilung der Informationen und Daten an den Cache-Server und das Kafka-Cluster übernimmt. Die Produkte auf der Webseite werden von der MySQL-Datenbank geladen. Um diese zu entlasten wird der Cacher-Server verwendet, der die Daten zwischenspeichert. Wird ein Produkt in den Warenkorb gelegt, so wird mittels JavaScript eine Nachricht (eng. message) in das Topic “cart-data” des Kafka-Clusters gesendet. Die Web-Applikation ist somit der Producer im Kafkakontext. Die Spark App ist Consumer des Topics “cart-data”. Neue Messages werden von dieser ausgewertet und in Batches verarbeitet. Die Spark App speichert gleichzeitig die aus den Messages gewonnen Informationen persistent in der MySQL-Datenbank ab. Zur Darstellung des Warenkorbs, werden die Daten von der Web-Applikation wiederum direkt aus der Datenbank geladen.

## Vorstellung elementarer Codebestandteile

Die nachstehende Funktion wird verwendet, um die gespeicherten Informationen zu den im Warenkorb hinterlegten Artikeln aus der Datenbank abzurufen. Die maximale Anzahl der auszugebenden Einträge im Warenkorb, kann dabei über die Mitgabe des Parameters “productCount” vorgegeben werden.

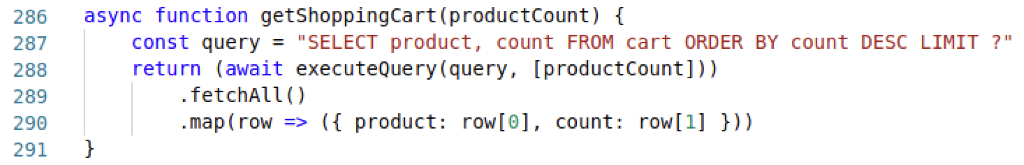


Abbildung Funktion „productCount“

Wird im Web-Shop ein einzelner Artikel ausgewählt, wird ein Fetch auf /pushToCart/<productID> ausgeführt. Im Rahmen dieses Fetch wird die JavaScript Funktion “sendTrackingMessage” ausgeführt, die eine Message an das Kafka-Cluster sendet. Die zu übertragenden Informationen beinhalten dabei die Bezeichnung des Produktes, sowie den Zeitstempel des Zugriffs. Im Erfolgsfall wird in der Konsole die Rückmeldung des erfolgreichen Sendens der Informationen an Kafka quittiert. Andernfalls wird eine entsprechende Fehlermeldung an die Konsole ausgegeben.



Abbildung Fetch „pushToCart“

# Data Science am Beispiel Human Resources

## Aufgabenstellung

Das Ziel der Aufgabenstellung ist es, eine Data Science Anwendung zu entwickeln und dabei die in der Vorlesung kennengelernten Techniken einzusetzen. Die Anwendung wird gemäß des CRISP-DM Vorgehensmodells (siehe Abbildung 1) strukturiert und soll dessen Ablauf folgen.

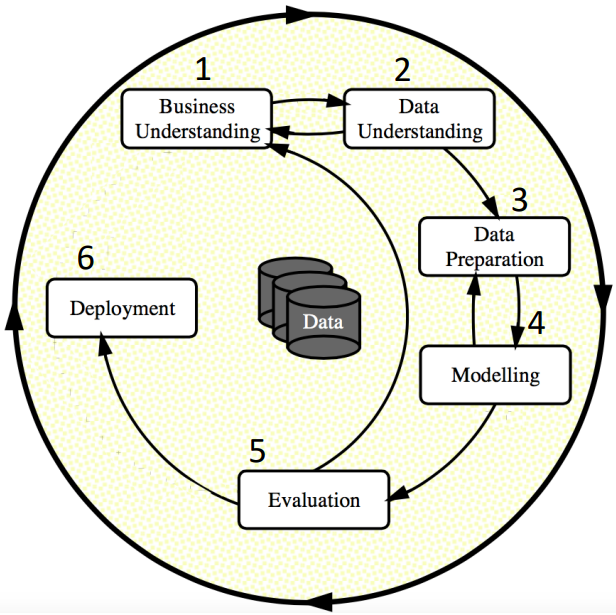


Abbildung 4 CRISP-DM Process Modell

Gewünscht sind hierbei ein oder mehrere Jupyter Notebooks, die unter Verwendung von pyspark und der mllib das Modell erzeugen (Schritt 4) und evaluieren (Schritt 5). Welche Algorithmen sie einsetzen, ist ihnen selbst überlassen. Das Deployment (Schritt 6) soll auf einem Hadoop Cluster erfolgen (gegebenenfalls als Python Anwendung). Wichtig ist die Dokumentation ihres iterativen Vorgehens. Grundlegende Entwicklungsstufen und Varianten des Modells und der Datenaufbereitung müssen für den Leser erhalten bleiben.

## Use-Case 1 - mushrooms

### Business Understanding

Im ersten Anlauf wurde versucht, ein Algorithmus zu entwickeln, welcher anhand von verschiedenen Merkmalen von Pilzen ausgibt, ob diese essbar oder giftig sind.

### Data Understanding

Für den oben beschriebenen Business-Case wurde zu Beginn ein passender Datensatz gesucht. Dieser enthält über 23 Arten von Pilzen, wobei jede Art wird als definitiv essbar, definitiv giftig oder von unbekannter Essbarkeit identifiziert werden. Diese letztere Klasse wird der giftigen Art zugeordnet. Des Weiteren enthält der ausgewählte Datensatz die Informationen von 8.124 Pilzen, welche mit 21 Kriterien beschrieben sind.

### Data Preparation

Um in diesen einen tieferen Einblick zu erhalten, wurde versucht, diesen für die weitere Verarbeitung vorzubereiten. Dies umfasste die Umwandlung der zeichenbasierten Informationen in Zahlwerte, welche von den Algorithmen genutzt werden können. Bei dieser Aufbereitung sind wir auf Probleme bei der Indexierung gestoßen, welche im nachfolgenden näher beschrieben werden.

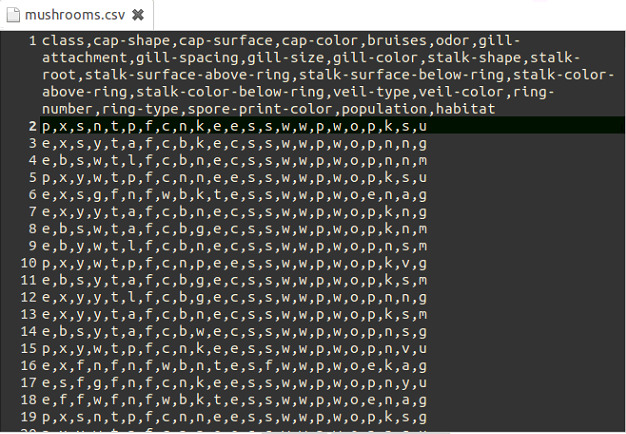


Abbildung 5 Datensatz mushrooms unbehandelt

Wie auf Abbildung 5 zu sehen ist, sind alle Daten zeichenbasierend gespeichert. Wie oben beschrieben, werden diese mithilfe von Indexern (siehe Abbildung 6) umgewandelt.

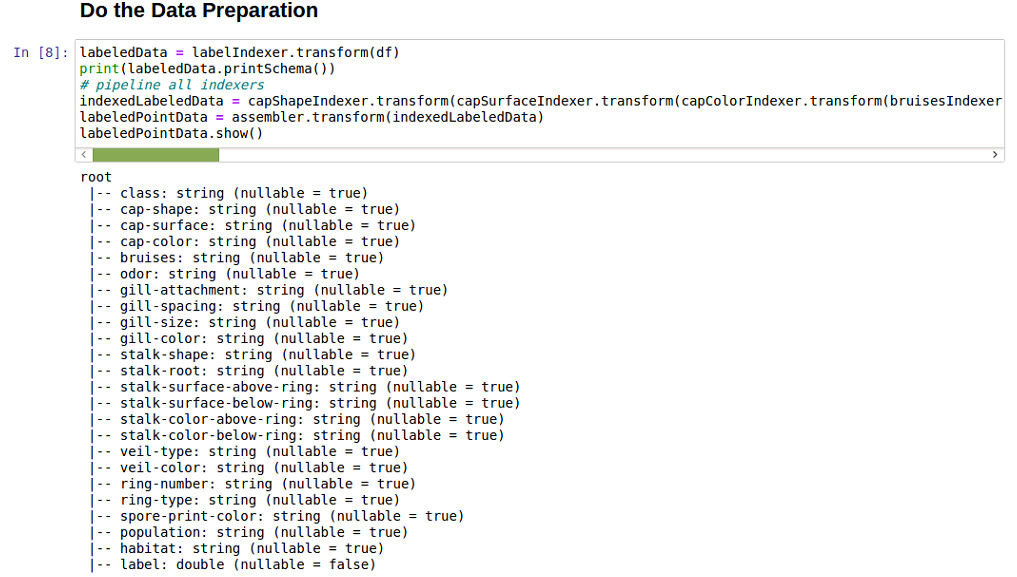


Abbildung 6 Data Preparation mushrooms

Die umgewandelten Daten sind in Abbildung 7 ersichtlich.

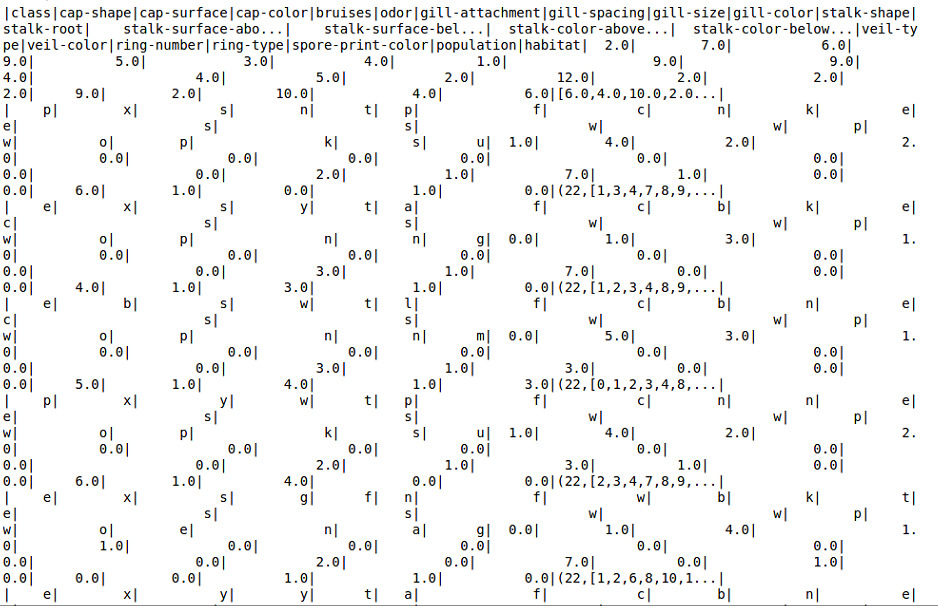


Abbildung Datensatz mushrooms behandelt

Wie aus den Screenshots ersichtlich wird, ist der Aufwand für die Datenaufbereitung sehr hoch. Des Weiteren enthält der Datensatz keinen unique Identifier, weshalb die Ergebnisse der Algorithmen nicht validiert werden können. Aufgrund der begrenzten Bearbeitungszeit entschied sich deshalb das Team, einen anderen Business-Case zu bearbeiten.

## Use-Case 2 – Human Ressources

### Business Understanding

Im ausgewählten Business-Case soll nun herausgefunden werden, welcher Mitarbeiter das Unternehmen verlassen wird.

### Data Understanding

Hierfür wurde ein passender Datensatz zu Mitarbeiterinformationen mit 312 Beobachtungen, die durch jeweils 35 Merkmale (siehe Abbildung 8) repräsentiert werden, ausgewählt. Dieser Datensatz beinhaltet Daten, wie zum Beispiel Name, Gehalt und Position im Unternehmen.



Abbildung 8 Merkmale Human Ressources

Um ein initiales Verständnis über den Datensatz zu erlangen, wurde mit den bestehenden Daten eine Visualisierung für einen Anwendungsfall durchgeführt. Spezifisch wurde der Leistungsindex in Relation zum Rekrutierungsweg dargestellt. So wäre beispielsweise eine Erkenntnis aus dem Graphen, dass Bewerber von diversen Jobmessen die Leistungserwartungen vergleichsweise am Häufigsten übertreffen.

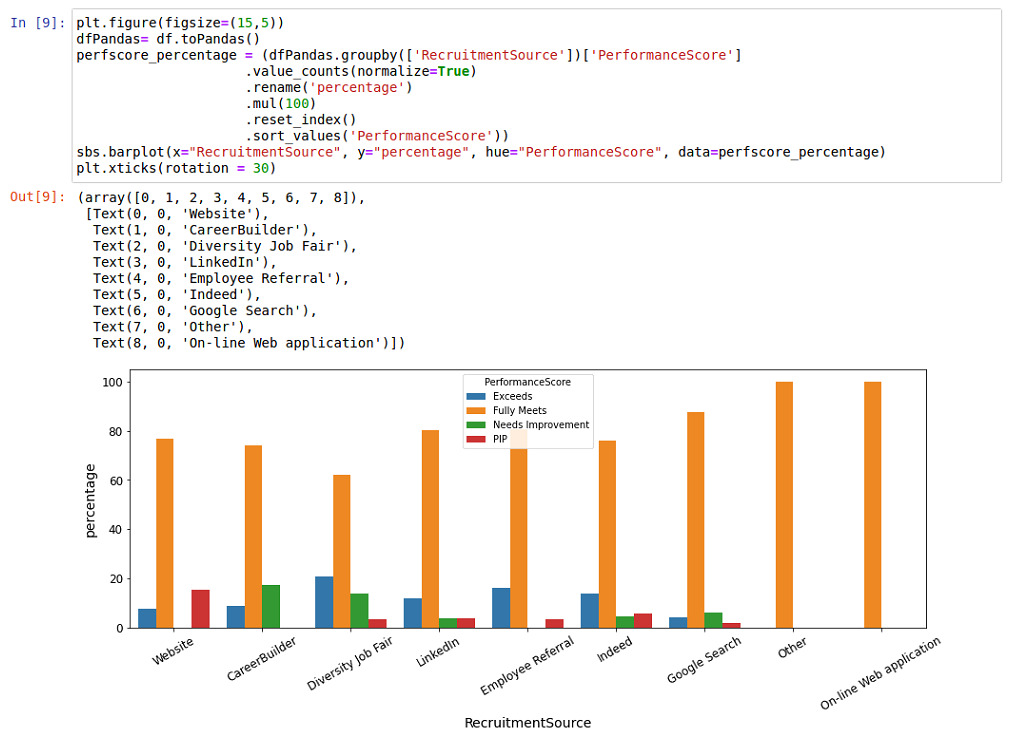


Abbildung 9 Leistungsindex in Relation zum Rekrutierungsweg

### Data Preparation

Um die Daten in den Algorithmen nutzen zu können, wurden alle String-Werte in Integer umgewandelt. Im Falle von Datumsfeldern wurden diese mithilfe der Funktion „unix\_timestamp“ in Long formatiert. Für die String-Werte wurden, wie schon im Use-case 1, im Zuge des Preprocessings die in pyspark zu Verfügung stehenden Index-Funktionalitäten verwendet.

### Modeling

Zu Beginn der Modellierung wurde eine zusätzliche Spalte erstellt. Diese soll die Differenz zwischen des Einstellungs- und Kündigungsdatums eines Mitarbeiters berechnen. Des Weiteren wurden zunächst die jeweiligen Stundenlöhne berechnet und anschließend mithilfe des Bucketizers den verschiedenen zuvor definierten Werten zugeordnet. Hierdurch konnte die Anzahl der unterschiedlichen Werte deutlich reduziert werden.

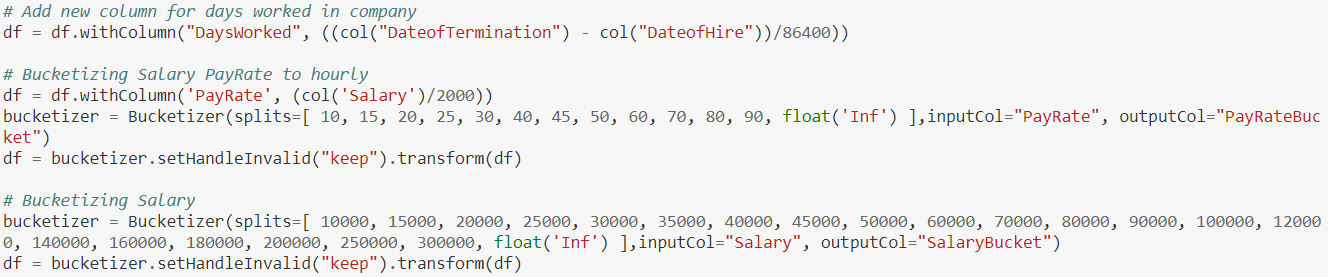


Abbildung 10 Berechnung „DaysWorked“

Die resultierende Spalte „days worked“ berechnet mithilfe der Funktion dateDiff das gewünschte Ergebnis. Auf Basis dieser Spalte wurde ein Scatter-Plot aufgesetzt, welches unter anderem auf diese Differenz zugreift und diese zusammen mit dem Stundenlohn der Mitarbeiter darstellt (siehe Abbildung 10). Der Status des jeweiligen Mitarbeiters wird zudem über die Farbe visualisiert.

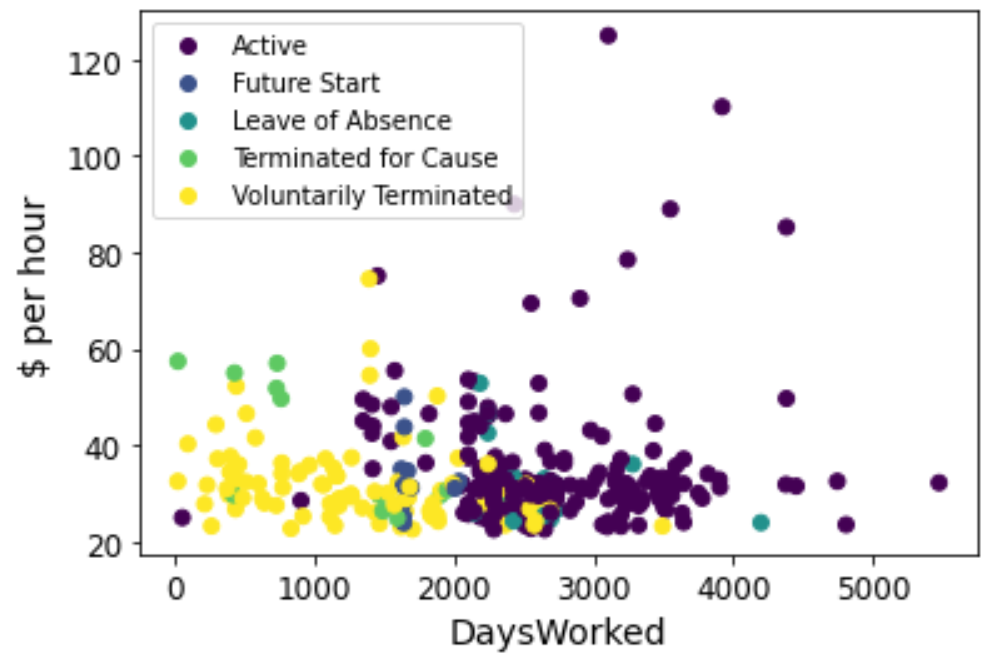


Abbildung 10 Scatter-Plot

### Evaluation

Validierung

### Deployment