

# Laborbericht im Modul Data Science and Big Data

---

W3M20018.1

<b>Mitglieder Gruppe 3</b>	Manmeet Kaur (CAS-TMINF20-W; 6935679) Daniel Kirste (CAS-WWB20-W; 3678893) Felix Öchnser (CAS-TMIE19-W; 8912778) Rouven Schnapka (CAS-WMWI20-W; 5074486) Mario Scholz (CAS-WMWI20-S; 6425929)
----------------------------	---

<b>Abgabedatum</b>	23.12.2020
--------------------	------------

<b>Link zum Repository:</b>	<a href="https://github.com/dkirste/datascience2020">https://github.com/dkirste/datascience2020</a>
-----------------------------	---

# Inhalt

Big Data Prototyp am Beispiel eines Webshops .....	1
Aufgabenstellung .....	1
Vorstellung Use-Case .....	1
Aufbau Architektur .....	1
Vorstellung elementarer Codebestandteile .....	2
Data Science am Beispiel Human Resources .....	3
Aufgabenstellung .....	3
Use-Case 1 - Mushrooms .....	4
Business Understanding.....	4
Data Understanding .....	4
Data Preparation .....	5
Use-Case 2 – Human Resources.....	6
Business Understanding.....	6
Data Understanding .....	6
Data Preparation .....	8
Modeling .....	9
Scatter-Plot.....	9
Decision-Tree .....	9
Clustering .....	11
Random-Forest.....	12
Evaluation .....	13
Clustering .....	13
Euklidische Distanz .....	14
Random-Forest.....	14
Deployment.....	15

# 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 Anwendungs idee 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.

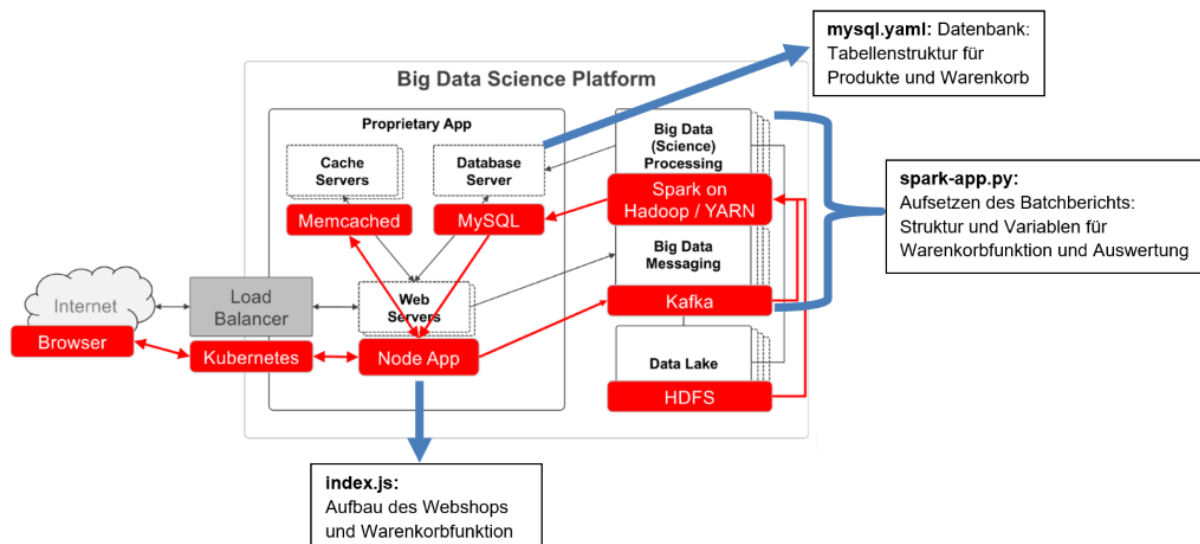


Abbildung 1 Big Data Science Platform

Durch das Aufrufen des Webshops sendet der Browser eine Anfrage an einen Load-Balancer, welcher in der vorliegenden Anwendung auf einem Kubernetes Cluster 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.

```
286  async function getShoppingCart(productCount) {
287      const query = "SELECT product, count FROM cart ORDER BY count DESC LIMIT ?"
288      return (await executeQuery(query, [productCount]))
289          .fetchAll()
290          .map(row => ({ product: row[0], count: row[1] }))
291  }
```

Abbildung 2 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.

```
330  // PushToCart
331  app.get("/pushToCart/:product", (req, res) => {
332      // Send the tracking message to Kafka
333      sendTrackingMessage({
334          product: req.params["product"],
335          timestamp: Math.floor(new Date() / 1000)
336      }).then(() => console.log("Sent to kafka"))
337          .catch(e => console.log("Error sending to kafka", e))
338      res.send(`<h1>Succeeded!`)
339  });
```

Abbildung 3 Fetch „pushToCart“

# Data Science am Beispiel Human Resources

## Aufgabenstellung

Das Ziel der Aufgabenstellung ist es, ein Data Science Anwendung zu entwickeln und dabei die in der Vorlesung kennengelernten Techniken einzusetzen. Die Anwendung wird gemäß des CRISP-DM Vorgehensmodells (siehe Abbildung 4) strukturiert und soll dessen Ablauf folgen.

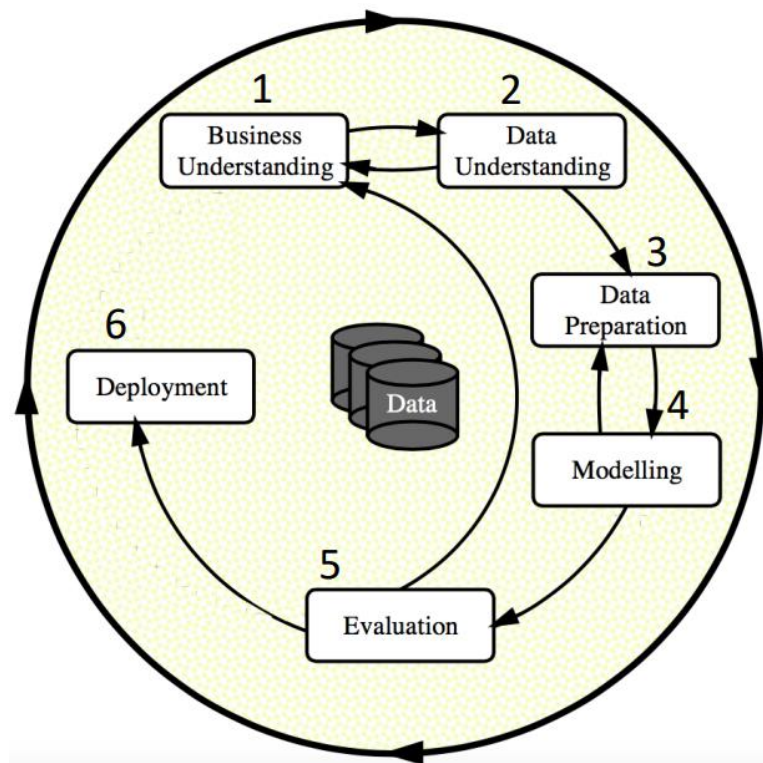


Abbildung 4 CRISP-DM Process Modell

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

Die erste Phase konzentriert sich auf die präzise Beschreibung der betriebswirtschaftlichen Problemstellung und deren Ziele. Auf Basis dieser Ziele sollen im weiteren Verlauf des Datamining-Prozesses konkrete Anforderungen an die Datenanalyse formuliert und verfolgt werden können.

Im vorliegenden Fall steht die Betrachtung von Pilzen im Vordergrund, welche anhand verschiedener Merkmale in die Kategorien „essbar“ und „giftig“ eingeteilt werden können. Ein mögliches Ziel dieser Implementierung könnte die Klassifizierung von Pilzen mittels verschiedener Merkmale sein.

### Data Understanding

In der zweiten Phase sollen Anforderungen an die Rohdaten für den Business-Case definiert werden. Dazu gehören Tätigkeiten, wie das Sammeln von Daten, deren Beschreibung und Untersuchung, sowie deren Bewertung.

Auf den Prozess der Datensammlung wurde in diesem Laborbericht aufgrund der beschränkten Zeit verzichtet. Stattdessen wurde ein bereits erfasster Datensatz der Webseite kaggle.de verwendet. Der verwendete Datensatz enthält 23 Arten von Pilzen, wobei jede Art als definitiv essbar, definitiv giftig oder von unbekannter Essbarkeit identifiziert wird. Diese letztere Klasse wird der giftigen Art zugeordnet. Klassifiziert werden die Pilze anhand von 21 Kriterien. Darunter zählen Stielfarbe, Stielform, Ringnummer, Ringtyp, Kappenform, Kappenoberfläche, etc. Die Kriterien werden durch Buchstaben repräsentiert, die die Abkürzung des Kriteriums darstellen. Beispielsweise eine spitze Kappenform wird durch den Buchstaben „s“ repräsentiert. Diese Darstellung kann Abbildung 5 entnommen werden. Eine flache Kappenform durch den Buchstaben „f“. Im Datensatz sind die Information von 8.124 Pilzen anhand der 21 Kriterien erhoben worden.

```
1 class,cap-shape,cap-surface,cap-color,bruises,odor,gill-  
  attachment,gill-spacing,gill-size,gill-color,stalk-shape,stalk-  
  root,stalk-surface-above-ring,stalk-surface-below-ring,stalk-color-  
  above-ring,stalk-color-below-ring,veil-type,veil-color,ring-  
  number,ring-type,spore-print-color,population,habitat  
2 p,x,s,n,t,p,f,c,n,k,e,e,s,s,w,w,p,w,o,p,k,s,u  
3 e,x,s,y,t,a,f,c,b,k,e,c,s,s,w,w,p,w,o,p,n,n,g  
4 e,b,s,w,t,l,f,c,b,n,e,c,s,s,w,w,p,w,o,p,n,n,m  
5 p,x,y,w,t,p,f,c,n,n,e,e,s,s,w,w,p,w,o,p,k,s,u  
6 e,x,s,g,f,n,f,w,b,k,t,e,s,s,w,w,p,w,o,e,n,a,g  
7 e,x,y,y,t,a,f,c,b,n,e,c,s,s,w,w,p,w,o,p,k,n,g  
8 e,b,s,w,t,a,f,c,b,g,e,c,s,s,w,w,p,w,o,p,k,n,m  
9 e,b,y,w,t,l,f,c,b,n,e,c,s,s,w,w,p,w,o,p,n,s,m  
10 p,x,y,w,t,p,f,c,n,p,e,e,s,s,w,w,p,w,o,p,k,v,g  
11 e,b,s,y,t,a,f,c,b,g,e,c,s,s,w,w,p,w,o,p,k,s,m  
12 e,x,y,y,t,l,f,c,b,g,e,c,s,s,w,w,p,w,o,p,n,n,g  
13 e,x,y,y,t,a,f,c,b,n,e,c,s,s,w,w,p,w,o,p,k,s,m  
14 e,b,s,y,t,a,f,c,b,w,e,c,s,s,w,w,p,w,o,p,n,s,g  
15 p,x,y,w,t,p,f,c,n,k,e,e,s,s,w,w,p,w,o,p,n,v,u  
16 e,x,f,n,f,n,f,w,b,n,t,e,s,f,w,w,p,w,o,e,k,a,g  
17 e,s,f,g,f,n,f,c,n,k,e,e,s,s,w,w,p,w,o,p,n,y,u  
18 e,f,f,w,f,n,f,w,b,k,t,e,s,s,w,w,p,w,o,e,n,a,g  
19 p,x,s,n,t,p,f,c,n,n,e,e,s,s,w,w,p,w,o,p,k,s,g
```

Abbildung 5 Datensatz mushrooms unbehandelt



## Data Preparation

In der Data Preparation werden die Daten ausgewählt, bereinigt, transformiert und formatiert. Dies umfasst die Umwandlung der zeichenbasierten Informationen in Zahlwerte, welche von den Algorithmen genutzt werden können. Die Umwandlung wird durch StringIndexer realisiert. Die vorbereiteten Daten sind auszugsweise in Abbildung 6 dargestellt. Hierbei wurden einige Probleme mit dem Datensatz offensichtlich. Zum einen ist in den Daten keine eindeutige Nummer zur Identifizierung des konkreten Pilzes vorhanden. Es kann somit keine „händische“ Überprüfung stattfinden, ob ein späterer Algorithmus richtig funktioniert. Zum anderen gehen Informationen zur Ähnlichkeit von Merkmalen verloren. Würde beispielsweise ein Clustering der Kappenform stattfinden, müssten spitze und sehr spitze Kappenformen näher zueinander sein als spitze zu flachen Kappenformen. Durch die Zuordnung von Zahlenwerten geht dieser Zusammenhang verloren. Aufgrund der aufgetretenen Probleme wurde sich dazu entschieden einen neuen Datensatz zu wählen, der Großteils auf numerischen Daten basiert und eine eindeutige Identifizierung zulässt.

class	cap-shape	cap-surface	cap-color	bruises	odor	gill-attachment	gill-spacing	gill-size	gill-color	stalk-shape	stalk-root	stalk-surface-above...	stalk-surface-below...	stalk-color-above...	stalk-color-below...	veil-type	veil-color	ring-number	ring-type	spore-print-color	population	habitat	2.0	7.0	6.0	9.0
4.0	2.0	9.0	2.0	4.0	10.0	5.0	2.0	12.0	2.0	2.0	2.0	6.0	[6.0,4.0,10.0,2.0...	6.0	4.0	10.0	2.0	9.0	3.0	4.0	1.0	2.0	9.0	7.0	6.0	9.0
	p	x	s	n	t	p	f	w	c	n	w	k	p	e												
e			s			s																				
w		o	p		k	s		u	1.0	4.0	2.0															2.
0		0.0			0.0			0.0																		0.0
0.0			0.0		2.0		1.0		7.0	1.0																0.0
0.0	e	6.0	x	1.0	0.0	1.0		0.0	(22,[1,3,4,7,8,9,...																	e
c			s		y	t	a		f	c	b	w	k	p												
w		o	p		n		n	g	0.0	1.0	3.0															1.
0		0.0			0.0			0.0																		0.0
0.0			0.0		3.0		1.0		7.0	0.0																0.0
0.0	e	4.0	b	1.0	3.0	w	t	l	0.0	(22,[1,2,3,4,8,9,...																e
c			s		w		s		f	c	b	w	n	p												1.
w		o	p		n		n	m	0.0	5.0	3.0															0.0
0		0.0			0.0			0.0																		0.0
0.0			0.0		3.0		1.0		3.0	0.0																0.0
0.0	p	5.0	x	1.0	4.0	w	t	p	3.0	(22,[0,1,2,3,4,8,...																e
e			s		w		s		f	c	n	w	n	p												2.
w		o	p		k		s	u	1.0	4.0	2.0															0.0
0		0.0			0.0			0.0																		0.0
0.0			0.0		2.0		1.0		3.0	1.0																0.0
0.0	e	6.0	x	1.0	4.0	g	f	n	0.0	(22,[2,3,4,7,8,9,...																t
e			s				n	s		w		w	k	p												
w		o	e		n		a	g	0.0	1.0	4.0															1.
0		1.0			0.0			0.0																		0.0
0.0			0.0		2.0			0.0																		1.0
0.0	e	0.0	x	0.0	1.0	y	t	a	0.0	(22,[1,2,6,8,10,1...																e
					y				f	c	b	n														

Abbildung 6 Datensatz mushrooms behandelt

## Use-Case 2 – Human Resources

Als alternativer Datensatz, der vorwiegend auf numerischen Daten beruht, wurde das „Human Resources Dataset“ ausgewählt. Dieser beinhaltet anonymisierte Personaldaten eines amerikanischen Unternehmens.

### Business Understanding

Da die Akquise neuer Mitarbeiter für Unternehmen sehr zeitaufwändig und kostenintensiv ist, besteht großes Interesse darin, potenzielle Abgänge bestehender Mitarbeiter zu erkennen. Im ausgewählten Business-Case soll ein Vorhersagemodell entwickelt werden, welches es den Unternehmen ermöglicht, diese Mitarbeiter frühzeitig zu identifizieren. Anhand dieser Erkenntnisse können anschließend entsprechende Gegenmaßnahmen eingeleitet werden.

### Data Understanding

Für den Business-Case wurde ein passender Datensatz zu Informationen von 312 Mitarbeitern ausgewählt, die jeweils durch 35 Merkmale (siehe Abbildung 7) repräsentiert werden. Dieser Datensatz beinhaltet Daten, wie zum Beispiel Name, Gehalt und Position im Unternehmen.

```
root
|-- EmpLastname: string (nullable = true)
|-- EmpFirstname: string (nullable = true)
|-- EmpID: integer (nullable = true)
|-- MarriedID: integer (nullable = true)
|-- MaritalStatusID: integer (nullable = true)
|-- GenderID: integer (nullable = true)
|-- EmpStatusID: integer (nullable = true)
|-- DeptID: integer (nullable = true)
|-- PerfScoreID: integer (nullable = true)
|-- FromDiversityJobFairID: integer (nullable = true)
|-- Salary: integer (nullable = true)
|-- TermID: integer (nullable = true)
|-- PositionID: integer (nullable = true)
|-- Position: string (nullable = true)
|-- State: string (nullable = true)
|-- Zip: integer (nullable = true)
|-- DOB: string (nullable = true)
|-- Sex: string (nullable = true)
|-- MaritalDesc: string (nullable = true)
|-- CitizenDesc: string (nullable = true)
|-- HispanicLatino: string (nullable = true)
|-- RaceDesc: string (nullable = true)
|-- DateofHire: string (nullable = true)
|-- DateofTermination: string (nullable = true)
|-- TermReason: string (nullable = true)
|-- EmploymentStatus: string (nullable = true)
|-- Department: string (nullable = true)
|-- ManagerName: string (nullable = true)
|-- ManagerID: integer (nullable = true)
|-- RecruitmentSource: string (nullable = true)
|-- PerformanceScore: string (nullable = true)
|-- EngagementSurvey: double (nullable = true)
|-- EmpSatisfaction: integer (nullable = true)
|-- SpecialProjectsCount: integer (nullable = true)
|-- LastPerformanceReview Date: string (nullable = true)
|-- DaysLateLast30: integer (nullable = true)
|-- Absences: integer (nullable = true)
```

Abbildung 7 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, die über Empfehlungen in das Unternehmen aufgenommen wurden, prozentual am meisten die Erwartungen erfüllen oder übertreffen. Zu sehen ist die in Abbildung 8, anhand des grünen Balkens (Needs Improvement) der Gruppe „Employee Referral“. Die Gruppen „Other“ und „Online Web Application“ können vernachlässigt werden, da hierbei nur ein Datenwert („Fully Meets“) auftritt. Dies deutet daraufhin, dass wahrscheinlich die Daten nicht aussagekräftig genug sind.

```
In [9]: plt.figure(figsize=(15,5))
dfPandas = df.toPandas()
perfscore_percentage = (dfPandas.groupby(['RecruitmentSource'])['PerformanceScore']
                        .value_counts(normalize=True)
                        .rename('percentage')
                        .mul(100)
                        .reset_index()
                        .sort_values('PerformanceScore'))
sbs.barpplot(x="RecruitmentSource", y="percentage", hue="PerformanceScore", data=perfscore_percentage)
plt.xticks(rotation = 30)
```

```
Out[9]: (array([0, 1, 2, 3, 4, 5, 6, 7, 8]),
 [Text(0, 0, 'Website'),
  Text(1, 0, 'CareerBuilder'),
  Text(2, 0, 'Diversity Job Fair'),
  Text(3, 0, 'LinkedIn'),
  Text(4, 0, 'Employee Referral'),
  Text(5, 0, 'Indeed'),
  Text(6, 0, 'Google Search'),
  Text(7, 0, 'Other'),
  Text(8, 0, 'On-line Web application')])
```

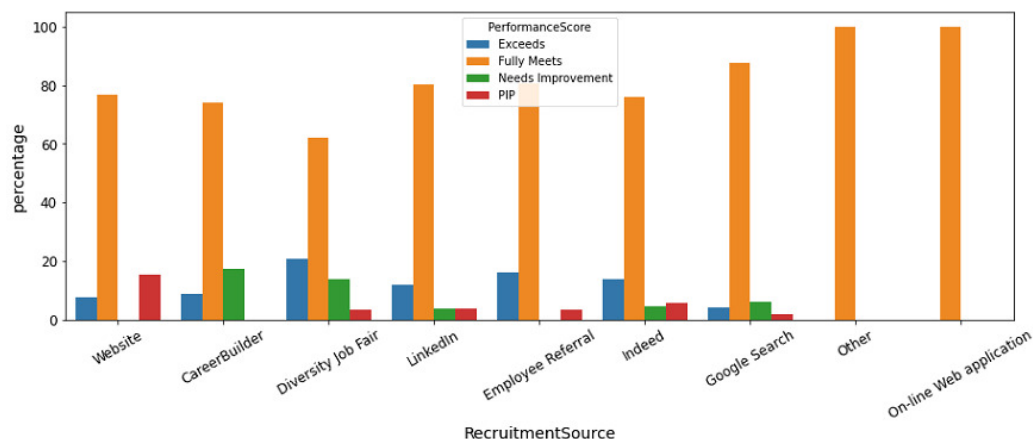


Abbildung 8 Leistungsindex in Relation zum Rekrutierungsweg

## Data Preparation

Um die Daten in den Algorithmen nutzen zu können, wurden diese im Rahmen der Vorverarbeitung (Preprocessing) behandelt. Dabei wurden wie schon in Use-Case 1 alle String- in Integer-Werte mithilfe der in pyspark zur Verfügung stehenden Index-Funktionalitäten umgewandelt. Die Datumsfelder wurden in einen Unixtimestamp (Datentyp: Long) konvertiert, um diese weiterhin korrekt verwenden zu können. Die Umwandlung ist der Abbildung 9 zu entnehmen. Aus den Unixtimestamps des Einstellungsdatums und des Kündigungsdatum wurden die „DaysWorked“ berechnet. Im Fall, dass ein Mitarbeiter noch in der Firma arbeitet, wurde der 01.01.2021 als „Kündigungsdatum“ angenommen, um die Berechnung zu ermöglichen. Nach Vervollständigung und den ersten Versuchen des Modelings wurde deutlich, dass die Daten noch nicht ausreichend vorverarbeitet wurden. Im Datensatz traten vereinzelt „Null“ Werte auf, die, wie in Abbildung 9 zu sehen ist, mit der Funktion „na.fill“ ausgebessert und durch repräsentative Werte ersetzt wurden. Beim Versuch der Implementierung eines Decision-Trees ergab sich der Fehler, dass zu viele unterschiedliche Datenwerte im Datensatz vorhanden sind. Daraufhin wurden die Gehaltsdaten und die daraus errechneten Stundenlöhne (eng. Payrate) Buckets zugeordnet. Diese verringern die Varianz der Daten, dadurch, dass diese sozusagen gerundet werden. Ein Gehalt von 58.900\$ wird so beispielsweise dem Bucket 8 zugeordnet, dass alle Gehälter zwischen 55.000\$ und 60.000\$ repräsentiert. Als Schrittweite wurden 5.000\$ gewählt. Ab 200.000\$ wurde das Bucket bis unendlich gesetzt, da hier nur der Geschäftsführer liegt.

```
# Fill null values
df = df.na.fill({'TermReason': 'Unknown', 'ManagerID': 0, 'DaysLateLast30': 0, 'DateofTermination': '1/01/2021', 'LastPerformanceReview_Date': '1/01/2021'})

# Convert date-strings to dates
df = df.withColumn('DateofHire', unix_timestamp(col('DateofHire'), 'M/dd/yyyy'))
df = df.withColumn('DOB', unix_timestamp(col('DOB'), 'MM/dd/yy'))
df = df.withColumn('DateofTermination', unix_timestamp(col('DateofTermination'), 'M/dd/yyyy'))
df = df.withColumn('LastPerformanceReview_Date', unix_timestamp(col('LastPerformanceReview_Date'), 'M/dd/yyyy'))

# Add new column for days worked in company
df = df.withColumn("DaysWorked", ((col("DateofTermination") - col("DateofHire"))/86400))

# Bucketizing Salary PayRate to hourly
df = df.withColumn('PayRate', (col('Salary')/2000))
bucketizer = Bucketizer(splits=[ 10, 15, 20, 25, 30, 40, 45, 50, 55, 60, 65, 70, 75, 80, 85, 90, float('Inf') ],inputCol="PayRate", outputCol="PayRateBucket")
df = bucketizer.setHandleInvalid("keep").transform(df)

# Bucketizing Salary
bucketizer = Bucketizer(splits=[ 10000, 15000, 20000, 25000, 30000, 35000, 40000, 45000, 50000, 55000, 60000, 65000, 70000, 75000, 80000, 85000, 90000, 95000, 100000, 105000, 110000, 115000, 120000, 125000, 130000, 135000, 140000, 145000, 150000, 155000, 160000, 165000, 170000, 175000, 180000, 185000, 190000, 195000, 200000, float('Inf') ],inputCol="Salary", outputCol="SalaryBucket")
df = bucketizer.setHandleInvalid("keep").transform(df)
```

Abbildung 9 Berechnung „DaysWorked“

## Modeling

Im Rahmen der Modellierung werden die für die Aufgabenstellung geeigneten Methoden des Data Minings auf den in der Datenvorbereitung erstellten Datensatz ausgewählt. Typisch für diese Phase sind die Erstellung von Testmodellen, deren Bewertung, sowie die anschließende Optimierung der verwendeten Parameter.

### Scatter-Plot

Der Stundenlohn der einzelnen Mitarbeiter in Form eines Scatter-Plots gegen die bereits gearbeiteten Tage im Unternehmen aufgetragen (siehe Abbildung 10). Der Status des jeweiligen Mitarbeiters wird zudem über die Farbe visualisiert. Hierbei lassen sich bereits Cluster erkennen. Eine mögliche These wäre hierbei: „Je weniger Tage ein Mitarbeiter gearbeitet hat und je geringer sein Gehalt ist, desto wahrscheinlicher ist dessen freiwillige Kündigung“. Daraus ergibt sich die These, dass ein Mitarbeiter mit hohem Gehalt und vielen Tagen im Unternehmen wahrscheinlich nicht gekündigt wird oder freiwillig das Unternehmen verlässt.

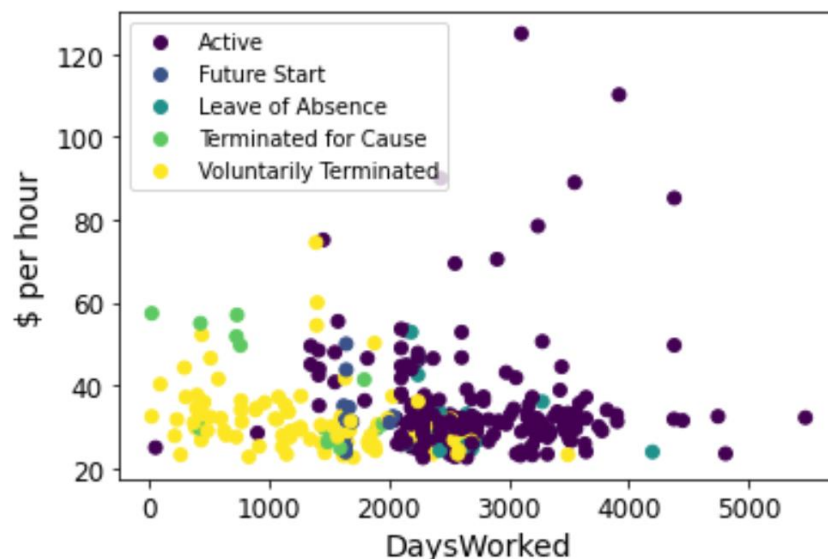


Abbildung 10 Scatter-Plot Datensatz

### Decision-Tree

Um eine Vorhersage treffen zu können, welcher Mitarbeiter das Unternehmen als nächstes verlassen wird, wurde ein Entscheidungsbaum (eng. Decision-Tree) aufgebaut. Dieser Decision-Tree wird mit der Variable paramGrid parametrier. Hier wurde beispielsweise die maximale Tiefe des Entscheidungsbaumes durch maxDepth auf fünf Ebenen eingestellt. Des Weiteren wurde mit dem Parameter minInfoGain vorgegeben, dass nur Features mit einem minimalen Informationsgehalt von 0,05 bei der Entscheidung berücksichtigt werden.

Um im weiteren Verlauf den Algorithmus trainieren und validieren zu können, wurde der Datensatz in einen Trainings- und einen Testdatensatz im Verhältnis 60 zu 40 aufgeteilt. Als Validierungsansatz wurde auf die Methode der Cross-Validation gesetzt und anschließend das Ergebnis dargestellt.

```

featureIndexer = VectorIndexer(inputCol="features",outputCol="indexedFeatures", maxCategories=6)

predConverter = IndexToString(inputCol="prediction",outputCol="predictedLabel",labels=termIndexer.labels)

dt = DecisionTreeClassifier(labelCol="Termd", featuresCol="features")

paramGrid = ParamGridBuilder().addGrid(dt.maxDepth, [ 5 ]) \
                                .addGrid(dt.minInfoGain, [0.05]) \
                                .addGrid(dt.minInstancesPerNode, [10]) \
                                .addGrid(dt.maxBins, [28]) \
                                .build()

splits = df.randomSplit([0.6, 0.4 ], 1234)
train = splits[0]
test = splits[1]
pipeline = Pipeline(stages= [termIndexer, empLastnameIndexer, empFirstnameIndexer, positionIndexer, stateIdIndexer, sexIndexer,
evaluator = BinaryClassificationEvaluator(labelCol="Termd",rawPredictionCol="rawPrediction", metricName="areaUnderROC")

cv = CrossValidator(estimator=pipeline, evaluator=evaluator,estimatorParamMaps=paramGrid,numFolds=3, parallelism=2)
cvModel = cv.fit(train)

treeModel = cvModel.bestModel.stages[7]
print("Learned classification tree model:\n",treeModel)
print("Best Params: \n", treeModel.explainParams())

predictions = cvModel.transform(test)
predictions.select("prediction", "Termd", "predictedLabel", "rawPrediction", "Termd", "features").show()

accuracy = evaluator.evaluate(predictions)
print("Test Error = ",(1.0 - accuracy))

```

Abbildung 11 Decision-Tree Coding

Wie Abbildung 12 zu entnehmen ist, folgt ein Test Error von 0. Da dies einer perfekten Vorhersage entspricht, kann hierbei von einem Implementierungsfehler ausgegangen werden. Im Rahmen der Implementierung wurde eine Vielzahl an Versuchen unternommen den Entscheidungsbaum richtig zu implementieren. Eine starke Vereinfachung der Daten auf 5 verschiedene Features hat ebenfalls zu keiner Veränderung des Test Errors geführt. Aufgrund der begrenzten Bearbeitungszeit des Laborberichts wurde sich dazu entscheiden, den Entscheidungsbaum nicht mehr weiter zu verfolgen.

prediction	Termd	predictedLabel	rawPrediction	Termd	features
0.0	0	0	[129.0,0.0]	0	(24,[2,3,4,5,8,11...]
1.0	1	1	[0.0,50.0]	1	[1.0,1.0,1.0,5.0,...]
1.0	1	1	[0.0,50.0]	1	(24,[1,3,4,5,7,8,...]
0.0	0	0	[129.0,0.0]	0	(24,[1,2,3,4,5,8,...]
0.0	0	0	[129.0,0.0]	0	(24,[2,3,4,5,8,11...]
1.0	1	1	[0.0,50.0]	1	(24,[2,3,4,5,7,8,...]
0.0	0	0	[129.0,0.0]	0	(24,[1,3,4,5,8,11...]
1.0	1	1	[0.0,50.0]	1	(24,[3,4,5,7,8,11...]
1.0	1	1	[0.0,50.0]	1	[0.0,0.0,1.0,4.0,...]
0.0	0	0	[129.0,0.0]	0	[1.0,1.0,0.0,1.0,...]
1.0	1	1	[0.0,50.0]	1	(24,[2,3,4,5,7,8,...]
0.0	0	0	[129.0,0.0]	0	[1.0,1.0,0.0,1.0,...]
0.0	0	0	[129.0,0.0]	0	(24,[1,3,4,5,6,8,...]
1.0	1	1	[0.0,50.0]	1	[1.0,1.0,0.0,5.0,...]
0.0	0	0	[129.0,0.0]	0	(24,[3,4,5,8,11,1...]
0.0	0	0	[129.0,0.0]	0	(24,[0,1,2,3,4,5,...]
0.0	0	0	[129.0,0.0]	0	(24,[2,3,4,5,8,11...]
0.0	0	0	[129.0,0.0]	0	(24,[3,4,5,8,9,11...]
0.0	0	0	[129.0,0.0]	0	(24,[3,4,5,8,11,1...]
0.0	0	0	[129.0,0.0]	0	(24,[2,3,4,5,8,11...]

only showing top 20 rows

Test Error = 0.0

Abbildung 12 Decision-Tree Ergebnisse

## Clustering

Um neue Erkenntnisse über den Datensatz zu erlangen, wurde ein Clustering auf den Datensatz angewendet. Ziel hierbei war es, so die Daten in Cluster einzuteilen, dass von den Clustern auf das Angestelltenverhältnis rückgeschlossen werden kann. Somit soll bestenfalls aufgezeigt werden können, welche Mitarbeiter Merkmale besitzen, die bei anderen Mitarbeitern zu einem Abgang geführt haben.

Hierfür wurden diese Informationen in einem Feature-Array mithilfe des auf Abbildung 13 gezeigten Vektorassembler zusammengefasst.

```
vecAssembler = VectorAssembler(inputCols=["PayRate", "DaysWorked"], outputCol="features")
new_df = vecAssembler.transform(df)
```

Abbildung 13 Vector Assembler Clustering

Für die Cluster-Analyse wurde auf den KMeans-Algorithmus gesetzt, welcher ein Verfahren zur Vektorquantisierung darstellt. Mithilfe des Parameters ‚k‘ wird die Anzahl der bekannten Cluster vorgegeben. Der Seed ist der Initialisierungswert, der eine zufällige Clusterbildung und Variation der Clusterbildung ermöglicht.

Zur späteren Evaluation des Clusterings wird der Datensatz in Trainings und Testdaten aufgeteilt (Verhältnis 95 zu 5). Der Parameter K wird auf 5 gesetzt, damit 5 Cluster gebildet werden. Der Seed wurde zufällig gewählt. Nach mehreren Versuchen hat sich ergeben, dass der Seed 1337 zum besten Ergebnis führt.

```
kmeans = KMeans(k=5, seed=1337)

# Split data
splits = new_df.randomSplit([0.95, 0.05], 420)
training = splits[0]
test = splits[1]
```

Abbildung 14 KMeans Clustering

Das Ergebnis der Cluster-Analyse wird mithilfe eines Scatter-Plots, wie er auf Abbildung 15 dargestellt ist, visualisiert. Um das Ergebnis direkt mit dem Angestelltenverhältnis vergleichen zu können, wird dieses nochmals auf Abbildung 16 dargestellt.

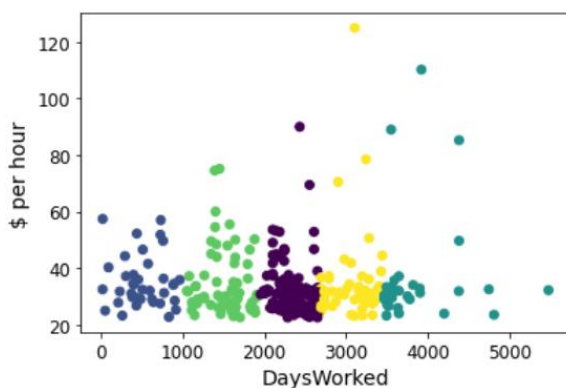


Abbildung 15 Scatter-Plot Clustering

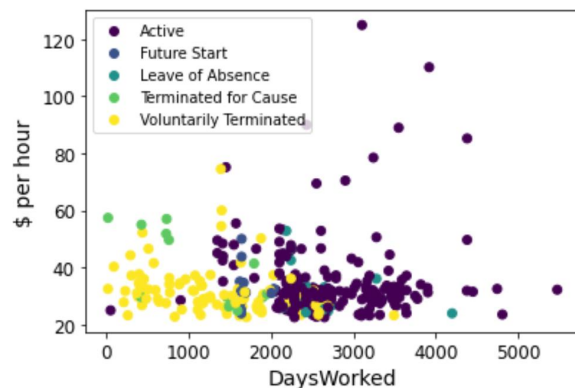


Abbildung 16 Scatter-Plot Datensatz

Wie der direkte Vergleich zeigt, finden sich die meisten aktiven Mitarbeiter (Abbildung 16 „Active“) in den oberen Clustern (Cluster 3 bis 5) wieder. Des Weiteren sind die Mitarbeiter, welche das Unternehmen freiwillig verlassen haben (Abbildung 16 „Voluntarily Terminated“), den ersten beiden Clustern zugeordnet. Eine weitere bzw. detailliertere Aussage ist auf dieser Basis leider nicht möglich.



## Random-Forest

Als Alternative zum Decision-Tree und um eine detailliertere Aussage treffen zu können, als bislang mit dem Clustering möglich ist, wurde daraufhin ein Random-Forest angewandt. Dies ist ein Algorithmus, der sich für Klassifizierungs- und Regressionsaufgaben nutzen lässt. Er kombiniert die Ergebnisse vieler verschiedener Entscheidungsbäume, um bestmögliche Entscheidungen zu treffen. In diesem Fall sagt er aus, ob ein Mitarbeiter das Unternehmen verlässt.

Zunächst wurde, wie Abbildung 17 zeigt, der Datensatz in Trainings- und Testdaten im Verhältnis 90 zu 10 aufgeteilt.

```
labeledPointDataSet = new_df
splits = labeledPointDataSet.randomSplit([0.9, 0.1], 420)
training = splits[0]
test = splits[1]
```

Abbildung 17 Random Forest Aufteilung Trainings- und Testdaten

Um den Algorithmus anwenden zu können wurde zunächst der bereits vorhandene Vektorassembler mit den Parametern EngagementSurvey und SpecialProjectsCount erweitert.

```
vecAssembler = VectorAssembler(inputCols=["PayRate", "DaysWorked", "EngagementSurvey", "SpecialProjectsCount"], outputCol="features")
new_df = vecAssembler.transform(df)
```

Abbildung 18 Vector Assembler Erweiterung für Random Forest

Anschließend wurde der Klassifizierungsalgorithmus (RandomForestClassifier) definiert und mithilfe diverser Einstellungen auf unseren Fall hin optimiert. Zu Beginn bestand der Wunsch, den Angestelltenstatus mithilfe des Random-Forest-Classifiers zu bestimmen. Hierbei stießen wir jedoch bei der Implementierung auf ähnliche Fehlerbilder wie beim zuvor angewandten Decision-Tree. Die Fehlerursache bestand höchstwahrscheinlich darin, dass es sich beim Angestelltenstatus nicht um einen binären Wert handelt. Offensichtlich ist es auf unserer Datenbases nicht möglich, ein Feature vorherzusagen, welches mehr als zwei Ausprägungen einnehmen kann. Daher entschied sich das Team das binäre Feature „Termd“ für die weitere Betrachtung zu verwenden. „Termd“ repräsentiert den Status des Mitarbeiters, ob dieser gekündigt ist (1) oder aktiv im Unternehmen arbeitet (0). Diese Umstellung könnte auch beim Entscheidungsbaum ein möglicher Lösungsansatz gewesen sein, konnte jedoch aufgrund des begrenzten Zeitrahmens nicht näher verfolgt werden.

```
rf = RandomForestClassifier(labelCol="Termd", featuresCol="features", impurity="gini", \
minInstancesPerNode=10, featureSubsetStrategy='sqrt', subsamplingRate=0.95, seed=12345)
```

Abbildung 19 Random Forest Aufteilung Trainings- und Testdaten

Die Modellierungsphase wird mit dem Training des Random Forest abgeschlossen.

```
rfModel = rf.fit(training)
```

Abbildung 20 Random Forest Aufteilung Trainings- und Testdaten

## Evaluation

Im Evaluationsschritt werden die erstellten Modelle und deren Ergebnisse bewertet und mit der zu Beginn definierten Aufgabenstellung abgeglichen. Ziel ist es, das für den Use-Case passendste Modell auszuwählen und gegebenenfalls zu optimieren.

### Clustering

Zur Evaluierung der vom Algorithmus definierten Cluster werden im ersten Schritt die Testdaten visualisiert.

```
evaluate_model = model.transform(test)

pddf_pred = evaluate_model.toPandas().set_index('EmpID')
pddf_pred.head()
scatter = plt.scatter(pddf_pred.DaysWorked, pddf_pred.PayRate, c=pddf_pred.prediction)
plt.xlabel("DaysWorked")
plt.ylabel("$ per hour")
plt.show()
```

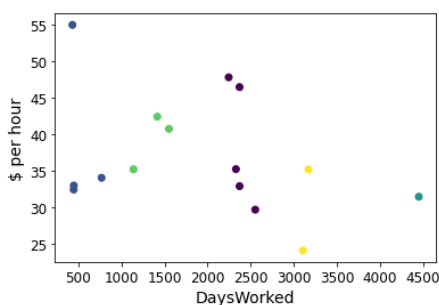


Abbildung 21 Evaluierung Clustering

Zusammen mit der Zuordnung zwischen Cluster und Angestelltenstatus wird die Vorhersage textuell ausgegeben, ob ein Mitarbeiter zur kündigungsgefährdeten Gruppe gehört.

So kann zum Beispiel aus der ersten Konsolenausgabe herausgelesen werden, dass die Mitarbeiterin „Sarah Akinkuolie“ mit einer Bezahlung von 32 \$ pro Stunde und einer Unternehmenszugehörigkeit von 447 Tagen zur kündigungsanfälligen Gruppe gehört.

```
# Writing columns into dict
eval_dict = evaluate_model.toPandas().set_index('EmpID').T.to_dict('list')
#print(eval_dict)
job_status = ['will stay.', 'will terminate.', 'will stay.', 'will terminate.', 'will stay.']
for empID in eval_dict:
    print("{0} {1} earns {2:.0f} dollars per hour with {3:.0f} days worked and {4}"
          .format(
              eval_dict[empID][1],
              eval_dict[empID][0],
              eval_dict[empID][-5],
              eval_dict[empID][-6],
              job_status[int(eval_dict[empID][-1])]
          )
    )
```

```
Sarah Akinkuolie earns 32 dollars per hour with 447 days worked and will terminate.
Colby Andreola earns 48 dollars per hour with 2244 days worked and will stay.
Renee Becker earns 55 dollars per hour with 432 days worked and will terminate.
Mia Brown earns 32 dollars per hour with 4449 days worked and will stay.
David Gordon earns 24 dollars per hour with 3105 days worked and will stay.
Alfred Hitchcock earns 35 dollars per hour with 2328 days worked and will stay.
Ming Huynh earns 34 dollars per hour with 770 days worked and will terminate.
Lindsey Langford earns 33 dollars per hour with 448 days worked and will terminate.
Binh Le earns 41 dollars per hour with 1552 days worked and will terminate.
Giovanni Leruth earns 35 dollars per hour with 3168 days worked and will stay.
Louis Punjabhi earns 30 dollars per hour with 2552 days worked and will stay.
Andrew Szabo earns 46 dollars per hour with 2370 days worked and will stay.
Charlie Wang earns 42 dollars per hour with 1416 days worked and will terminate.
Jordan Winthrop earns 35 dollars per hour with 1140 days worked and will terminate.
Jason Woodson earns 33 dollars per hour with 2370 days worked and will stay.
```

Abbildung 22 Textuelle Ausgabe Clustering

## Euklidische Distanz

Zur weiteren Evaluierung der Cluster wurde die euklidische Distanz berechnet. Diese repräsentiert den Abstand zwischen zwei Punkten als Strecke in einem Raum. In unserem Fall gibt diese Funktion die Distanz zwischen den einzelnen Clustermitten an.

Die Silhouette gibt zudem für eine Beobachtung an, wie gut die Zuordnung zu den beiden nächst gelegenen Clustern ist.

```
evaluator = ClusteringEvaluator()

silhouette = evaluator.evaluate(train_model)
print("Silhouette with squared euclidean distance = " + str(silhouette))

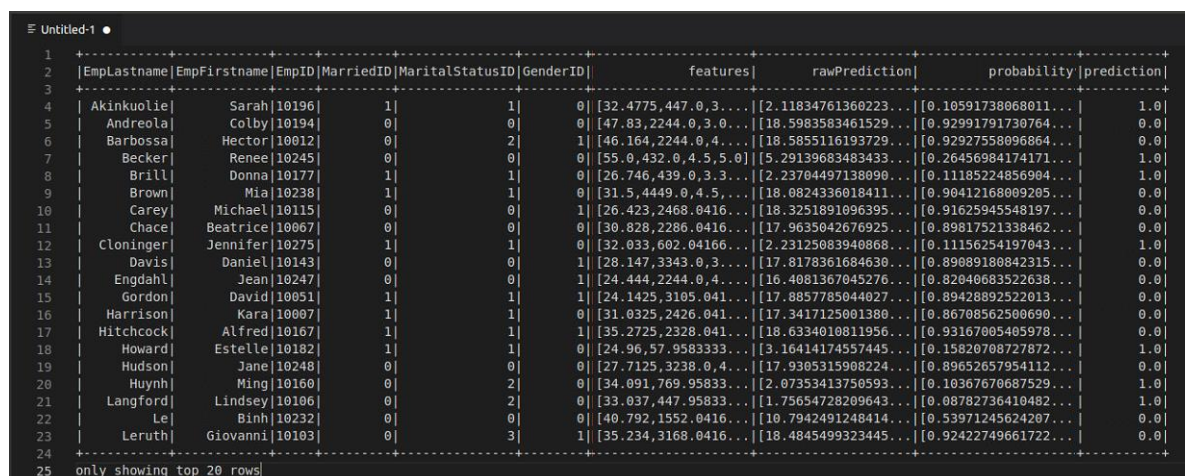
centers = model.clusterCenters()
print("Cluster Centers: ")
for center in centers:
    print(center)

Silhouette with squared euclidean distance = 0.6766397020968253
Cluster Centers:
[3.31744112e+01 2.33859813e+03 4.11728972e+00 1.43925234e+00]
[ 35.23352703 519.05405405 4.10594595 1.45945946]
[3.71466833e+01 3.85566667e+03 4.14733333e+00 9.33333333e-01]
[3.45861406e+01 1.52101562e+03 4.04421875e+00 1.14062500e+00]
[3.44007241e+01 3.03587931e+03 4.14086207e+00 6.20689655e-01]
```

Abbildung 23 Euklidische Distanz und Silhouette Clustering

## Random-Forest

Um den trainierten Random-Forest-Classifizier bewerten zu können, werden nun die Testdaten in den Random-Forest eingegeben, um dessen Genauigkeit zu bestimmen. Die dabei ausgegebene Entscheidung (eng. Prediction) des Random-Forests stellt den Wert für „Termd“ dar.



	EmpLastname	EmpFirstname	EmpID	MarriedID	MaritalStatusID	GenderID	features	rawPrediction	probability	prediction
1	Akinkuolie	Sarah	10196	1	1	0	[32.4775, 447.0, 3.0, ...]	[2.11834761360223...	[0.10591738068011...	1.0
2	Andreola	Colby	10194	0	0	0	[47.83, 2244.0, 3.0, ...]	[18.5983583461529...	[0.92991791730764...	0.0
3	Barbossa	Hector	10012	0	2	1	[46.164, 2244.0, 4.0, ...]	[18.5855116193729...	[0.92927558096864...	0.0
4	Becker	Renee	10245	0	0	0	[55.0, 432.0, 4.5, 5.0]	[5.29139683483433...	[0.26456984174171...	1.0
5	Brill	Donna	10177	1	1	0	[26.746, 439.0, 3.3, ...]	[2.23704497138090...	[0.11185224856904...	1.0
6	Brown	Mia	10238	1	1	0	[31.5, 4449.0, 4.5, ...]	[18.0824336018411...	[0.90412168009205...	0.0
7	Carey	Michael	10115	0	0	1	[26.423, 2468.0416...	[18.3251891096395...	[0.91625945548197...	0.0
8	Chace	Beatrice	10067	0	0	0	[30.828, 2286.0416...	[17.9635042676925...	[0.89817521338462...	0.0
9	Cloninger	Jennifer	10275	1	1	0	[32.033, 602.04166...	[2.23125083940868...	[0.11156254197043...	1.0
10	Davis	Daniel	10143	0	0	1	[28.147, 3343.0, 3.0, ...]	[17.8178361684630...	[0.89809180842315...	0.0
11	Engdahl	Jean	10247	0	0	1	[24.444, 2244.0, 4.0, ...]	[16.4081367045276...	[0.82040683522638...	0.0
12	Gordon	David	10051	1	1	1	[24.1425, 3105.041...	[17.8857785044027...	[0.89428892522013...	0.0
13	Harrison	Kara	10007	1	1	0	[31.0325, 2426.041...	[17.3417125001380...	[0.86708562500690...	0.0
14	Hitchcock	Alfred	10167	1	1	1	[35.2725, 2328.041...	[18.6334010811956...	[0.93167005405978...	0.0
15	Howard	Estelle	10182	1	1	0	[24.96, 57.9583333...	[3.16414174557445...	[0.15820708727872...	1.0
16	Hudson	Jane	10248	0	0	0	[27.7125, 3238.0, 4.0, ...]	[17.9305315908224...	[0.89652657954112...	0.0
17	Huynh	Ming	10160	0	2	0	[34.091, 769.95833...	[2.07353413750593...	[0.10367670687529...	1.0
18	Langford	Lindsey	10106	0	2	0	[33.037, 447.95833...	[1.75654728209643...	[0.08782736410482...	1.0
19	Le	Binh	10232	0	0	0	[40.792, 1552.0416...	[10.7942491248414...	[0.53971245624207...	0.0
20	Leruth	Giovanni	10103	0	3	1	[35.234, 3168.0416...	[18.4845499323445...	[0.92422749661722...	0.0

Abbildung 24 Ausgabe Random-Forest

Durch den Vergleich der Prediction und dem korrekten Wert von „Termd“ lässt sich eine Fehlerwahrscheinlichkeit von 10% ermitteln (siehe Abbildung 25). Der Random-Forest gibt somit zu 90% ein richtiges Ergebnis aus. In Anbetracht dessen, dass es sich hierbei um menschliche Daten handelt und der Kündigungsstatus von menschlichen Entscheidungen, entweder vom Mitarbeiter selbst oder dem Chef abhängt, kann eine Trefferwahrscheinlichkeit von 90% als sehr gutes Ergebnis gewertet werden.

```
predictions = rfModel.transform(test)

evaluator = BinaryClassificationEvaluator(labelCol="Termd", rawPredictionCol="rawPrediction", metricName="areaUnderROC")
accuracy = evaluator.evaluate(predictions)
print("Test Error", (1.0 - accuracy))

Test Error 0.1009615384615385
```

Abbildung 25 Euklidische Distanz und Silhouette Clustering

## Deployment

Die Deployment-Phase bildet in der Regel die Endphase eines Data Mining-Projektes. Hier werden die gewonnenen Erkenntnisse so geordnet und präsentiert, dass der Auftraggeber dieses Wissen nutzen kann. Dazu gehört eine eventuelle Implementierungsstrategie, die Überwachung der Gültigkeit der Modelle, ein zusammenfassender Bericht und eine Präsentation.

Für eine mögliche Implementierung wurde ein pyspark-basierter Ansatz gewählt. Den Abschluss dieser Projektarbeit stellt die Vollendung dieses Berichtes dar.