Beleg 3 – Aufgabenstellung

Thema: Supervised Learning mit Naive Bayes

Ziel der Belegarbeit ist es, das Klasssifikationsverfahren Naive Bayes zu implementieren und es dann auf den bekannten Titanic Datensatz (https://www.kaggle.com/c/titanic/) anzuwenden. Dabei soll für eine Menge von Titanic-Passagieren möglichst treffsicher vorausgesagt werden, ob diese das Unglück überleben oder nicht.

Das Projekt teilt sich auf in die Klassen/Objekte: NaiveBayes, TitanicDataSet, CreatePrediction, CreatePredictionSimpleModel, TitanicStatistics, Utils und VegasUtils wobei nur in den ersten beiden Klassen Funktionen implementiert werden müssen – der Rest ist Infrastruktur, die Ihnen zur Lösung der Aufgabenstellungen bereit gestellt wird.

Dabei werden Ihnen die folgenden Funktionalitäten bereitgestellt:

- Objekt Utils: Methoden zum Laden und Parsen des Titanic-Datensatzes, Methoden zum Erzeugen eines für kaggle erforderlichen Files zur Datenübermittlung.
- Object CreatePrediction und CreatePredictionSimpleModel: Ausführbares Programm zur Erzeugung eines Files zur Übermittlung an kaggle.

Der Beleg teilt sich auf in zwei Teile:

1. Object NaiveBayes: In diesem Objekt soll der Naive Bayes Algorithmus implementiert werden und
2. Object TitanicDataSet: In diesem Objekt wird der Algorithmus mit dem Titanic-Datensatz angewendet.

Noch ein kleiner Hinweis: Falls die Dokumentation der Funktionen an manchen Stellen ungenau ist, dann schauen Sie sich einfach die Tests an. Diese dienen ebenfalls der Spezifikation. Falls Sie die Testresultate nicht nachvollziehen können, dann kann es hilfreich sein, sich noch einmal in die Vorlesung anzuschauen. In der Belegarbeit wird das vorgestellte Beispiel 1:1 umgesetzt.

Starten Sie mit dem ersten Teil, der Implementierung von Naive Bayes.

Teil 1: Implementierung von Naive Bayes

Für die Implementierung von Naive Bayes sind die Klassen/Objekte NaiveBayes sowie NaiveBayesTest (enthält Tests für die Klasse NaiveBayes) relevant. Die Tests orientieren sich genau an das in der Vorlesung präsentierte Beispiel der Vorhersage von Zugverspätungen. Dies ist dem Buch „Bramer, M.: Principles of Data Mining, Springer Press, 2016“ entnommen worden und kann auch dort nachgelesen werden.

Das Beispiel umfasst 20 Trainingsdatensätze mit den Attributen „day“,“season“,“wind“ und “rain“. Die Datensätze sind als Map modelliert. Diese Datensätze werden für die gesamte Implementierung des Algorithmus verwendet. Bitte beachten Sie, dass in dem Beispiel alle Attributwerte vom Typ String sind – um die Funktionen jedoch generisch zu halten, werden die Datensätze mit Map[String, Any] typisiert. Dies hat den Vorteil, dass Sie bspw. auch Zahlen (Int, Double, etc.) als Attributwerte verwenden können. Die Verwendeung dieser generischen Typen hat jedoch zur Folge, dass Sie an einigen Stellen eine konkrete Typisierung der verwendeten Typen (z.B. durch asInstanceOf[…] oder Pattern Matching) vornehmen müssen, um diese dann auch benutzen zu können.

Starten Sie mit den Funktionen countAttributeValues, getAttributes und getAttributeValues. Sie vereinfachen das Handling der Attribute. In der ersten Funktion werden innerhalb einer Menge von Datensätzen die Anzahl unterschiedlicher Attributwerte für jedes vorkommende Attribut gezählt. Die zweite Funktion sucht in einer Menge von Datensätzen alle Attribute (Attributnamen) die dort vorkommen. Die dritte Funktion extrahiert für jedes Attribut, die konkret vorkommenden Attributwerte und speichert sie innerhalb eines Sets.

Bestehen die ersten drei Funktionen die Tests, können Sie die nächsten drei Funktionen calcPriorPropabilities, calcAttribValuesForEachClass und calcConditionalPropabilitiesForEachClass angehen. In der ersten Funktion werden die A-priori-Wahrscheinlichkeiten (Prior Propabilites) ermittelt. Die A-priori-Wahrscheinlichkeit ist die relative Häufigkeit des Vorkommens einer Klasse im Datensatz. In der nächsten Funktion (calcAttribValuesForEachClass) müssen Sie die Anzahl der Vorkommen der einzelnen Attributwerte innerhalb der einzelnen Klassen ermitteln. Daraus lassen sich dann die Bedingten Wahrscheinlichkeiten der Attributwerte bestimmen. Diese werden errechnet, in dem die Anzahl der Vorkommen der Attributwerte innerhalb einer Klasse durch die absolute Häufigkeit des Vorkommens einer Klasse, geteilt wird (siehe Vorlesung).

Auf Basis der implementierten Funktionen können jetzt Vorhersagen getroffen werden, in dem die Wahrscheinlichkeitswerte für jede Klasse berechnet werden. Dies erfolgt in der Funktion calcClassValuesForPrediction. Diese bekommt als Parameter einen Datensatz für den die Vorhersage getroffen werden soll, dann die bereits berechneten Wahrscheinlichkeiten sowie die A-priori-Wahrscheinlichkeiten. Die Funktion muss jetzt die entsprechenden Werte aus den „Tabellen“ heraussuchen. Aus der Multiplikation der zusammengehörigen Werte ergibt sich dann die Wahrscheinlichkeit des Auftretens einer Klasse. Diese Berechnung erfolgt für alle Klassen. Jetzt muss nur noch die Klasse mit der größten Wahrscheinlichkeit gefunden werden, was in der Methode findBestFittingClass erfolgt.

Aus den Funktionen kann jetzt das Machine Learning Model gebildet werden. Es ist als Funktion repräsentiert, die ein zu klassifizierenden Datensatz sowie dem Attributname des Schlüsselelements auf den Schlüssel des Datensatzes sowie die vorhergesagte Klasse abbildet. Die Funktion applyModel wendet das Modell auf eine Mengen von Testdaten an und erzeugt für jeden Datensatz eine Schlüssel/Klassen-Kombination. Versuchen Sie diese Methoden zu verstehen – Sie müssen für den Titanic Datensatz selbst geschrieben werden.

Jetzt muss nur noch die Funktion calcConditionalPropabilitiesForEachClassWithSmoothing implementiert werden. Sie entspricht der Funktion calcConditionalPropabilitiesForEachClass nur dass dabei ein add one-Smoothing angewendet werden soll. Wie das funktioniert, können Sie der Vorlesung entnehmen.

Teil 2: Anwendung von Naive Bayes auf den Titanic Datensatz

Ist der Naive Bayes Algorithmus implementiert, so soll dieser auf den bekannten Titanic Datensatz angewendet werden. Dazu müssen die Daten entsprechend der Vorgaben aufbereitet werden, ein Modell erstellt werden (wie im Zugbeispiel) und das auf einen Satz Testdaten angewendet werden. Die Überprüfung der Ergebnisse erfolgt über die Seite von kaggle, für die Sie sich vorab einen Zugang einrichten müssen.

In dem Anwendungsbeispiel soll für eine Menge von Titanic-Passagieren möglichst treffsicher vorausgesagt werden, ob sie das Unglück überleben oder nicht. Der Datensatz enthält von allen Passagieren Informationen über das Geschlecht, Alter, Fahrpreis, Passagierklasse, Einstiegshafen, Kabine und die mitreisende Familie – wobei manche Attributwerte nicht belegt sind (fehlende Informationen).

Der Titanic Datensatz ist aufgeteilt in einen Trainings- und einen Testdatensatz. Der Trainingsdatensatz enthält Information über das Überleben des Passagiers (zugehörige Klasse) und ist dafür da, „das Modell zu lernen“.

Die Bewertung der Voraussage erfolgt über den Testdatensatz. Auf ihm wird das erlernte Modell angewendet, d.h. es wird eine Tabelle generiert, die die PassengerID und die Vorhersage enthält.

Eine endgültige Evaluation erfolgt dann über die kaggle-Seite. Nur hier sind die Informationen über das Überleben der Restpassagiere verfügbar.

Machen Sie sich im ersten Schritt mit dem Datensatz vertraut. Hierzu finden Sie schon einige Informationen in der Vorlesung. Weiter können Sie natürlich den Datensatz explorieren, in dem Sie ein Jupyter-Notebook mit Scala-Kernel verwenden. Solch eine Umgebung kann über docker problemlos (in der Regel) eingerichtet werden. Mit:

docker pull popatry/almond-images:almond-2.11.12-0.1.7 können Sie das Image herunterladen und mit:

docker run -it --rm -p 8888:8888 popatry/almond-images:almond-2.11.12-0.1.7 starten. Achtung: Hier wird die Scala-Version 2.11 verwendet, da die Bibliothek vegas für 2.11 kompiliert ist.

Machen Sie sich dann gegebenenfalls mit dem Vegas-Framework vertraut. Leider gibt es nur wenig Dokumentation – als Startpunkt kann ich ihnen das folgende Video von der Spark-Summit-Conference (https://www.youtube.com/watch?v=5GfQVsnHj88),die vegalite-Doku (https://vega.github.io/vega-lite-v1/docs/) und die Vegas-Doku (https://github.com/vegas-viz/Vegas) empfehlen. Mit den Beispielen im Projekt werden Sie dann schnell hineinfinden.

Um sich mit der kaggle-Infrastruktur vertraut zu machen, implementieren Sie im ersten Schritt ein Modell, dass bei Frauen voraussagt, dass diese überleben und die Männer nicht. Dieses Modell soll in der Funktion simpleModel implementiert werden. Mit der Klasse CreatePredictionSimpleModel können Sie dieses Modell auf den Testdatensatz anwenden. Diese Anwendung erzeugt ein File „TitanicSimplePrediction.txt“ im Home-Verzeichnis des Projekts, der bei kaggle eingereicht werden kann (erstellen Sie dazu vorab einen Account). Nach Auswertung der Ergebnisse sollte eine Treffergenauigkeit von 76.555% herauskommen.

Bei der Anwendung des Naive Bayes Algorithmus auf den Titanic Datensatz sollen die Attribute age, sex und pclass verwendet werden. Gehen Sie bei der Implementierung folgendermaßen vor: Starten Sie mit der Funktion countAllMissingValues bei der für eine Liste von Attributen gezählt wird, wie häufig diese im übergebenen Datensatz fehlen (in den Maps nicht belegt sind).

In der nächsten Funktion sollen sie aus einem Record die Attribute extrahieren, die für die Modellbildung verwendet werden sollen. Im ersten Versuch sollen bspw. nur die Attribute age, sex und pclass herangezogen. Das Herausziehen dieser Attribute erfolgt in dieser Methode.

In der nächsten Methode wird der Trainingsdatensatz zusammengestellt. Dazu müssen die relevanten Attribute (age, sex und pclass) herausgezogen werden. Dann müssen die fehlenden Werte im Attribut age mit Werten belegt werden und eine Kategorisierung der Alterswerte vorgenommen werden. Jedes vorkommende Alter als eigenen Attributwert zu verwenden, wäre nicht zweckmäßig. Die Art der Kategorisierung ist beliebig, sollte aber sinnvoll sein (Informationen finden Sie dazu in den Vorlesungsvideos).

Ist der Datensatz vorbereitet, so kann dann – analog zum Zugbeispiel – das Machine Learning Modell gebildet werden. Getestet werden kann das Modell im Kleinen durch den Test „Create Model with Titanic Dataset". Hier müssten Sie allerdings das Attribut age im Hinblick auf Ihre gewählte Kategorisierung vorab anpassen. Haben Sie das getan, sollte bei der Attributbelegung die erste Person überleben und die zweite nicht.

Der zweite Test erfolgt, in dem Sie das Modell auf den gesamten Testdatensatz anwenden. Dies erfolgt in der Applikation CreatePrediction. Sie erstellt eine Datei namens „TitanicPredictions.txt“ im Homeverzeichnis des Projekts. Diese erzeugte Datei können Sie bei kaggle einreichen und sich dann die Anzahl der richtigen Vorhersagen berechnen lassen. Die Güte sollte nicht geringer sein, als mit dem einfachen Modell. Schreiben Sie die erreichte Güte in einen Textfile namens „Accuracy.txt“ und legen Sie diesen im Homeverzeichnis des Projekts ab.

Optional: Versuchen Sie Ihr erreichtes Ergebnis zu verbessern, in dem Sie die Attributauswahl verbessern. Hinweise hierzu können Sie der Vorlesung sowie den kaggle-

Foren entnehmen. Testen Sie mindestens vier verschiedene Attributkombinationen und stellen Sie die Ergebnisse innerhalb eines Textfiles dar.

Im zweiten Teil der Belegarbeit gibt es noch ein paar Funktionen zur Vorbereitung auf die Klausur. Sie sollen mittels dem MapReduce-Algorithmus implementiert werden. Hierzu gibt es die Basisimplementierung in BasicOperations, dann die MapReduce-Klasse für die Funktionen und die Klasse MapReduceTest, die entsprechende Tests für die Funktionen bereitstellt.

Die Projekte sollen in Gruppen von 1-2 Personen umgesetzt werden. Die Präsentation des Belegs erfolgt in der letzten Übung vor der Klausur. Wichtig: Erzeugen Sie ein Textfile bei dem Sie beschreiben, welche Attribute sie in welcher Form für die Vorhersage verwendet haben und welches Ergebnis Sie damit erreicht haben (mind. drei verschiedene Varianten)