Data Mining und Maschinelles Lernen

Prof. Kristian Kersting
Steven Lang
Felix Friedrich

TECHNISCHE UNIVERSITÄT DARMSTADT

Sommersemester 2022
Übungsblatt 4

Benötigte Dateien
Alle benötigten Datensätze und Skriptvorlagen finden Sie in unserem Moodle-Kurs:
https://moodle.informatik.tu-darmstadt.de/course/view.php?id=1058

4.1 Entscheidungsbäume - Vor- und Nachteile

Entscheidungsbäume (engl. Decision Tree (DTs)) sind eine nichtparametrische, überwachte Lernmethode, die zur Klassifizierung und Regression verwendet wird. Ziel ist es, ein Modell zu erstellen, das den Wert einer Zielvariablen vorhersagt, indem einfache Entscheidungsregeln gelernt werden, die aus den Datenmerkmalen abgeleitet werden.

Entscheidungsbäume lernen Wenn-dann-sonst-Entscheidungsregeln um einen Datensatz zu approximieren. Je tiefer der Baum ist, desto komplexer sind die Entscheidungsregeln und desto passender ist das Modell.

a) **Vorteile**
Nennen Sie drei Vorteile von Entscheidungsbäumen:

b) **Nachteile**

Nennen Sie drei Nachteile von Entscheidungsbäumen:

4.2 Entscheidungsbäume - Klassifikation

`sklearn.tree.DecisionTreeClassifier` ist eine Klasse, die eine Mehrklassenklassifizierung eines Datensatzes durchführen kann.

Wie bei anderen Klassifikatoren nimmt `DecisionTreeClassifier` zwei Arrays als Eingabe entgegen: ein Array `X`, *sparse* oder *non-sparse*, mit der Größe `[n_samples, n_features]`, das die Trainingsproben enthält, und ein Array `Y` mit ganzzahligen Werten, mit der Größe `[n_samples]`, das die Klassenbezeichnungen für die Trainingsproben enthält.

a) **Unsichere Vorhersagen**

Entscheidungsbäume erlauben es eine (Pseudo)-Wahrscheinlichkeit für die Entscheidungen anzugeben. Finden Sie alle Entscheidungen des Modells für den Testdatensatz, für welches das Modell keine 100 % Zuordnung treffen konnte. Verwenden Sie dabei die Methode `clf.predict_proba`.

b) **Entscheidungsregionen**

Um die Entscheidungsregionen zu visualisieren, können wir eine Vorhersage für jeden möglichen Datenpunkt um die gültige Region um den 2D Iris-Datensatz machen.

Stellen Sie die Entscheidungsregionen In der Methode `plot_decision_boundary` dar.

Unterscheiden Sie dabei visuell die Trainings- und Testmenge und markieren Sie alle Punkte von a), bei welchem das Modell keine klare Entscheidung treffen konnte. Sie können dabei den Quelltext von `plot_decision_boundary` von `dmml_u2.zip/utils.py` als Referenz verwenden und erweitern.

c) **Kreuzvalidierung**

Wir können visuell erkennen, dass der Entscheidungsbaum mit zunehmender Tiefe dazu neigt, sich zu spezialisieren, um zu den einzelnen Datenpunkten zu passen, was zu komplexeren Entscheidungsregionen führt. Um den optimalen Wert für die maximale Tiefe zu finden, können wir außerdem eine Kreuzvalidierung des 2D-Iris-Datensatzes über einen Bereich möglicher Werte für den Parameter für die maximale Tiefe durchführen.

Implementieren Sie die Methode `get_acc_per_depth`, welche eine 10-fache Kreuzvalidierung durchführt und die korrespondierenden Train- und Testgenauigkeiten zurückgibt.

4.3 Entscheidungsbäume - Regression

Entscheidungsbäume können auch zur Regression verwendet werden. Dazu erstellen wir eine verrauschte Sinusfunktion. Wir können jetzt Entscheidungsbäume anpassen, um die Sinusfunktion zu erlernen. Als Ergebnis werden lokale lineare Regressionen gelernt, die sich der Sinuskurve annähern.

a) **Regression**

Verwenden Sie den `sklearn.tree.DecisionTreeRegressor` in der Methode `get_dt_prediction` um eine Regression mittels eines Entscheidungsbaum anzuwenden. Geben Sie anschließend die Vorhersagen des Modells für gegebenen Trainingdatensatz zurück.

b) **Ergebnisvisualisierung**

Wir können sehen, dass, wenn die maximale Tiefe des Baumes (gesteuert durch den Parameter `max_depth`) zu hoch eingestellt ist, die Entscheidungsbäume zu feine Details der Trainingsdaten lernen und aus dem Rauschen lernen, d.h. sie *überanpassen* sich.

Zeigen Sie die verschiedenen Anpassungen für die unterschiedlichen Einstellungen von `max_depth` in der Methode `plot_regression_models` an.