Einführung in das Maschinelle Lernen **Universität Hamburg**
WiSe 2023/24 Sören Laue

Übungsblatt 7

Aufgabe 1 – 15%

Nehmen Sie an, sie sollten die Konstruktion eines Entscheidungsbaumes für Klassifikation selbst implementieren. Die Splits der Knoten sollten auf dem Klassifizierungsfehler basieren. Welche Methoden müssten Sie implementieren und was wäre die Laufzeit Ihres Algorithmus? Wie würden Sie die Konstruktion möglichst effizient machen, so dass Sie auch Datensätze mit vielen Millionen Datenpunkte bearbeiten können?

Aufgabe 2 – 15%

Stellen Sie sich vor, Sie sollen ein Machine-Learning-Modell evaluieren, das für eine Personalabteilung Bewerber anhand ihrer Lebensläufe vorsortiert. Das System entscheidet für jede:n Bewerber:in, ob er entweder *gut*, *abgelehnt* oder auf der *Warteliste* ist. Man möchte weder zu viele ungeeignete Kandidat:innen einladen, noch gute Kandidat:innen ablehnen. Um das System zu testen, wurde es für einige Zeit auf alle Bewerbungen angewendet, aber die Personalverantwortlichen haben die Einordnung in die drei Kategorien weiterhin manuell un ohne Kenntnis der Modell-Klassifikation getroffen. Jetzt soll entschieden werden, ob das System gut genug ist, damit man den Kategorien äbgelehntünd / oder "gut"grundsätzlich vertrauen kann. Folgende Informationen wurden zunächst erhoben:

- Die Precision für *abgelehnt* ist mit 95% sehr hoch.
- Der Recall für *gut* ist mit 85% vertretbar.
- Es wurden grundsätzlich keine Frauen in *abgelehnt* einsortiert.

Ein Mitarbeiter behauptet, das System dürfe nicht eingesetzt werden, weil es Männer unfair behandele. Wie würden Sie diese Aussage einordnen? Welche Informationen fehlen Ihnen noch? Welche Maße würden sie benutzen, um die Unfairness zu bewerten, und welche Hypothese haben Sie über das Ergebnis? Finden Sie das System nützlich, und warum ggf. nicht?

Aufgabe 3 – 20%

Für manche Klassifikations-Szenarien möchte man manuell Vorwissen (*Prior Knowledge*) in das Modell integrieren. Solches Wissen kommt oft von Fachexpert:innen oder der Fachliteratur, und kann vor allem bei kleinen Datensätzen, seltenen Features, spezialisierten Problemstellungen oder Weltweissen außerhalb der Daten nützlich sein. Ein Beispiel: Ein Klassifikator soll anhand von Features vorhersagen, ob es klug ist, ein Taxi zu nehmen oder ÖPNV. Features wären Dinge wie Weg-Distanz, der Zeit-Puffer, der Abstand zur letzten Gehaltszahlung, die grundsätzliche Präferenz und das Wetter. Während es aus den Daten wahrscheinlich offensichtlich ist, dass bei Regen normalerweise ein Taxi gebucht wird, gilt das nicht bei Glatteis (was aber selten vorkommt). Erklären Sie für Naive Bayes, Decision Trees und für kNN jeweils, wie man dieses Wissen in den Klassifikator integrieren könnte. Welche Herausforderungen sehen sie bei der Anwendung in der Praxis (ggf. auch für andere Problemstellungen als die Taxi-Entscheidung)?

Aufgabe 4 – 50%

In der Vorlesung haben Sie Entscheidungsbäume und Random Forests als Klassifikations-Algorithmen kennengelernt. In dieser Übung sollen Sie beide Algorithmen auf zwei Klassifizierungsprobleme anwenden. Für diese Übung sind die Datensätze `iris_2D.csv` und `adult_onehotencoded.csv` vorgesehen. Den ersten Datensatz kennen Sie bereits aus einer vorherigen Übung. Der zweite Datensatz enthält verschiedene Features (= Merkmale) von Personen und ein Label (die Klasse) das angibt, ob die jeweilige Person über 50.000$ verdient oder nicht. Er wurde bereits für Sie mit One-Hot-Encoding vorverarbeitet (für Interessierte: `OneHotEncoder` in scikitlearn). Ihnen steht ein Jupyter Notebook `random_forest.ipynb` zur Verfügung, das ein grobes Gerüst für die Aufgabe bereitstellt. Sie sind nicht verpflichtet das Notebook zu nutzen, es dient nur zur Orientierung und Hilfestellung.

Part 1 (20%)

Arbeiten Sie zunächst mit dem Datensatz `iris_2D.csv`. Trainieren Sie sowohl einen Entscheidungsbaum als auch einen Random Forest auf den Daten. Für diese Aufgabe sollten Sie die Reihenfolge der Datenpunkte randomisieren und ein Test-Set mit 20% der Daten erstellen (`train_test_split`). Für alle Methoden, die Zufallsgeneratoren verwenden, setzen Sie den `random_state=20`. **Zur Kontrolle**: Ihre trainierten Klassifizierungsmethoden sollten beide eine *Accuracy* von 0.7 erreichen, wenn sie alle Angaben beachtet haben.
Trainieren Sie einen weiteren Entscheidungsbaum und visualisieren Sie diesen (`plot_tree`). Sie sollen für diese Visualisierung die Komplexität des Baums einschränken. Probieren Sie selbst aus, welche Parameter sich hierfür gut eignen und visualisieren Sie ihre subjektiv beste Lösung.

Part 2 (30%)

Im nächsten Schritt arbeiten Sie mit dem Datensatz `adult_onehotencoded.csv`. Ermitteln Sie für einen Entscheidungsbaum und einen Random Forest die *Accuracy* für Trainings- und Validierungsset mittels Kreuzvalidierung für verschieden Werte von `max_depth`, der maximalen Tiefe des Baums bzw. der Bäume der jeweiligen Klassifizierungsmethode. Variieren Sie dabei den Wert von `max_depth` im Intervall $[1, 20]$. Tragen Sie ermittelten Werte gegen die Baumtiefe für beide Algorithmen auf. Was fällt Ihnen auf? Welches Verhalten lässt sich vor allem bei tiefen Entscheidungsbäumen beobachten? Geben Sie außerdem für den Random Forest Classifier (trainiert ohne `max_depth`-Beschränkung) die 10 wichtigsten Features (basierend auf der `gini importance`), die die Methode zur Klassifizierung heranzieht. (Tipp: werfen Sie einen Blick auf die Dokumentation des `RandomForestClassifier`)
Hinweis: Die Kreuzvalidierung könnte bei den RandomForestClassifier relativ lange dauern. Überschreitet die Berechnung zehn Minuten, reduzieren Sie die Anzahl der Folds oder reduzieren Sie die Anzahl der Bäume der Methode. Die Anzahl der Bäume sollte > 10 sein.

Bitte lösen Sie die Fragen und die Programmieraufgaben bis zum **18. Dezember 2023**. Ihren Python-Code, sowie Visualisierungen können Sie in Form eines Jupyter Notebooks oder PDFs hochladen.