Labor 3

KeyWords: Boston House Dataset, KNN, RandomForest, Klassifikation, Regression

Aufgabe 1:

In diesem Labor werden Sie sich mit KNNs und RandomForest beschäftigen. Dabei werden Sie den Boston Housing Dataset analysieren. Öffnen Sie hierzu die Datei: „RandomForestClassification.py“ (Doku : https://keras.io/datasets/#boston-housing-price-regression-dataset)

```
from keras.datasets import boston_housing
from sklearn.ensemble import RandomForestRegressor

# Try to understand the differences between training and testing dataset
(x_train, y_train), (x_test, y_test) = boston_housing.load_data()
```

Der Boston House-Dataset zeigt Eigenschaften von Häusern auf und weist diese Preisen zu. Normalerweise handelt es sich hier um einen klassischen Regressionsdatensatz. Für diesen Anwendungsfall teilen wir jedoch die Hauspreise in Klassen ein, in dem jede Ganzzahl eine Klasse darstellt.

```
y_train_class = [int(i) for i in y_train]
```

1) Warum gibt es Trainings und Test Daten? Was ist der Unterschied? Welchen Nutzen haben diese?
2) Um sich eine Verteilung der Zielklassen anzuschauen kann pyplot verwendet werden. Nutzen Sie pyplot um sich die Verteilung in y_train_class anzuschauen. Ist diese Verteilung vorteilhaft? Falls nicht, was könnte man dagegen tun?

```
clf = RandomForestClassifier(n_estimators=100, max_depth=2,
                             random_state=0)
```

3) Hier wird der Random-Forest trainiert. Verstehen Sie die einzelnen Parameter mit Hilfe der Dokumentation:
   https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

4) Um eine Vergleichbarkeit von Einstellungen zu ermöglichen, nutzt man sogenannte „Metriken“. Diese verändern bzw. Trainieren keinen Ansatz und dienen lediglich als Vergleichsbasis. Beschreiben Sie eine mathematische Formel um eine Zahl zu beschreiben die zeigt wie „gut“ das Ergebnis auf den Testdaten ist. Verwenden Sie hierzu die Numpy-Bibliothek

```
import numpy as np
# Write a function to describe the error
error = ...

print(error)
```

Tipp: Beschreiben Sie einen „Fehler“. Ein Fehler kann nicht negativ sein. Es soll nur eine Zahl herauskommen am Ende.

5) Um übersichtliche Skripte zu erhalten lohnt es sich Hilfsfunktionen und andere Klassen, Funktionen und Daten auszulagern. Erstellen Sie eine neue Datei mit dem Namen „utils.py“. Schreiben Sie dort eine Funktion mit dem Namen **metric ,** welche als Input-Parameter y_true und y_pred aufweist. Implementieren Sie ihre Formel aus 4) in diese Funktion

   y_true : Groundtruth gegeben vom Datensatz
   y_pred: Predizierte Daten

   Eine Implementierung könnte wie folgt aussehen:

```python
def metric(y_true,y_pred):
    """
        @brief Will calculate the error between y_true and y_pred
        @brief y_true Groundtruth
        @brief y_pred Predizierte Daten
    """

    # write your metric here
    error = ...

    return error
```

6) Importieren Sie nun die geschriebene Funktion aus der utils.py in die RandomForestClassification.py Datei. Ersetzen Sie die Berechnung durch den Funktionsaufruf und validieren Sie diese auf Richtigkeit.
7) Unterschiedliche Einstellungen können zu unterschiedlichen Ergebnissen führen. Es lohnt sich diese systematisch zu analysieren. Schreiben Sie hierzu eine Schleife in der Sie unterschiedliche Random-Forest Strukturen testen. Dabei sollen Sie über die max_depth von 1 bis 10 iterieren. Speichern Sie den berechneten Fehler pro tiefe. Stellen sie diesen auf einem Plot dar. Was fällt Ihnen auf?
8) **BonusAufgabe**: Ersetzen Sie nun den RandomForestClassifier durch den RandomForestRegressor. Verwenden Sie am besten eine Kopie der Datei RandomForestClassification.py
   https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html
   Benutzten Sie jetzt die y_train zum trainieren. Vergleichen Sie die Ergebnisse zwischen RandomForestRegressor und RandomForestClassifier mit einem Plot

Aufgabe 2

1) Schauen Sie sich die Doku an unter: https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html und ermitteln Sie wie sie ein KNN-Algorithmus in Python mit Hilfe von sklearn anwenden.
2) Erstellen Sie eine Kopie der Datei „RandomForestClassification.py“ mit dem Namen: „KNN.py“ und fügen Sie dort den KNeighborsClassifier hinzu.
3) In Ihrer Schleife aus Aufgabe 1, Unteraufgabe 7) fügen Sie nun den KNN hinzu und trainieren diesen ebenfalls. Über die Schleife iterieren Sie über die Nachbarn.
4) Plotten Sie die Ergebnisse von der RandomClassification und des KNNs in ein Plot und vergleichen Sie.