Dokumentation Random Forest Classifier

Es wurde ein Random Forest Classifier eingesetzt, um die Früchte anhand ihrer Eigenschaften korrekt einer der Klassen Apfel, Banane oder Traube zuzuordnen. Der Random Forest ist ein Ensembleverfahren, das aus einer Vielzahl von Entscheidungsbäumen besteht, welche mit einer zufälligen Stichprobe der Trainingsdaten trainiert wurden. Die finale Klassifikation ergibt sich anschließend durch Mehrheitsentscheid aller Bäume.

Datenvorbereitung

Bevor die Daten für das Training des Modells benutzt werden können, müssen diese aufbereitet werden. Dafür wird der Datensatz fruit_data.xlsx in Python mithilfe der Bibliothek pandas eingelesen. Der Datensatz enthält für jede Frucht die verschiedenen Eingangsmerkmale color, size und weight, sowie die Zielvariable fruit_type, die angibt um welche Frucht es sich handelt. Da die Merkmale nicht numerisch sind, wurden diese mittels dem LabelEncoder aus der Bibliothek sklearn.preprocessing in Ganzzahlen umgewandelt.

Um die Modellleistung objektiv bewerten zu können und sicherzustellen, dass das Modell den Datensatz nicht einfach nur auswendig lernt, wurde dieser in einen Trainingsdatensatz von 80% und einen Testdatensatz von 20% mit der Funktion train_test_split() unterteilt.

Modellbildung

Der Random Forest wurde aus der Bibliothek sklearn.ensemble importiert und mit den folgenden Hyperparametern initialisiert.

```
model = RandomForestClassifier(
    n_estimators = 400,
    max_depth = None,
    min_samples_split = 3,
    min_samples_leaf = 2,
    max_features = 'sqrt',
    bootstrap = True,
    random_state = 7
)
```

Diese Parameter wurden bewusst eingestellt und gewählt, um ein Gleichgewicht zwischen Modellkomplexität und Generalisierungsfähigkeit zu erreichen.

- n_estimators = 400: Das Modell besteht aus 400 Entscheidungsbäumen, da eine hohe Anzahl an Bäumen für ein stabileres Ergebnis sorgt, weil zufällige Schwankung einzelner Bäume besser ausgeglichen werden können.
- max_depth = None: Die Tiefe der einzelnen Bäume ist unbegrenzt. Dadurch kann jeder Baum die Trainingsdaten vollständig aufspalten. Dies kann jedoch zu Overfitting führen, weshalb die nachfolgenden Parameter dieses kontrollieren.
- min_samples_split = 3: Eine neue Aufspaltung eines Knotens erfolgt nur, wenn mindestens drei Datenpunkte vorhanden sind, was eine zu starke Fragmentierung verhindert.
- min_samples_leaf = 2: Jedes Endblatt muss mindestens zwei Stichproben enthalten, wodurch vermieden wird, dass Ausreißer einen zu großen Einfluss haben.
- max_features = 'sqrt': Für jede Aufspaltung wird eine zufällige Teilmenge der Merkmale betrachtet, was die Diversität der Bäume erhöht.
- bootstrap = True: Jeder Baum wird auf einer zufälligen Stichprobe aus den Trainingsdaten trainiert.

Training und Bewertung

Nach der Initialisierung wird das Modell mit den Trainingsdaten trainiert. Jeder Baum lernt dabei leicht unterschiedliche Entscheidungsregeln, da er nur eine Teilmenge der Daten und Merkmale sieht. Die Vorhersage erfolgt über den Mehrheitsentscheid aller Bäume.

Nach dem Training wurde das Modell auf den bisher nicht gesehenen Testdaten getestet. Die vorhergesagten Klassen werden dann mit den tatsächlichen Zielwerten verglichen und damit die Genauigkeit berechnet. In typischen Durchläufen kann das Modell gut zwischen den Früchten unterscheiden und erreicht eine Genauigkeit von 80-97,5%.

Fazit

Das Random Forest Modell wurde aufgrund der hohen Genauigkeit, Robustheit gegenüber Ausreißern und der geringen Neigung zum Overfitting gewählt. Es bietet im Vergleich zu einem einzelnen Entscheidungsbaum eine deutliche stabilere Klassifikation, da zufällige Fehler einzelner Modelle durch die Ensemble-Struktur ausgeglichen werde**n.**

Dokumentation Neuronales Netz

Im Anschluss an den Random-Forest-Klassifikator wurde ein weiteres Modell mit einem neuronalen Netz getestet und untersucht, ob dieses eine Verbesserung zum vorherigen darstellt. Das Modell wurde mit TensorFlow / Keras umgesetzt.

Datenvorbereitung

Die Datenaufbereitung ist nahezu analog wie bei dem vorherigen Modell. Jedoch muss bei dem neuronalen Netz noch die Eingangsmerkmale standardisiert, also auf eine gemeinsame Skala gebracht werden. Dies wird mit der Funktion StandardScaler, die jedes Merkmal so transformiert, dass es im Mittel 0 ist und eine Standardabweichung von 1 hat. Wenn dies nicht gemacht wird, würde das Netz dem Attribut Gewicht eine stärkere Gewichtung zuordnen als der z.B. der Farbe, da dessen Werte viel größer sind.

Netzarchitektur

Das neuronale Netz besteht aus mehreren Schichten

- Input Layer: Diese Schicht nimmt die drei Eingangsmerkmale weight, color, und size auf.
- Hidden Layer 1 und 2: Diese Schichten enthalten 32 bzw. 16 Neuronen, die mit der ReLU-Aktivierungsfunktion arbeiten, was dafür sorgt, auch nichtlineare Zusammenhänge erfassen zu können. Im Anschluss wird eine Dropout-Schicht eingeführt, wobei 20% der Neuronen deaktiviert werden, um Overfitting zu vermeiden.
- Output Layer: Die letzte Schicht enthält 3 Neuronen, um die drei Fruchtarten zu unterscheiden. Jedem Neuron wird eine Wahrscheinlichkeit zwischen 0 und 1 zugeordnet, womit dann die Fruchtart identifiziert werden kann.

Training

Trainiert wurde das Modell mit dem Adam-Optimizer und einer Lernrate von 0,00056, welche durch Testläufe ermittelt wurde. Sie hat eine gute Balance zwischen Lernfortschritt und Stabilität. Für die Verlustfunktion kommt die Funktion categorical_crossentropy zum Einsatz, die während des Trainings nach jeder Vorhersage die errechneten Wahrscheinlichkeiten der Klassen mit dem wahren One-Hot-Vektor vergleicht. Sie misst, wie weit die Wahrscheinlichkeiten vom wahren Label entfernt sind. Um Overfitting zu verhindern, wurde Early Stopping eingesetzt, was das

Training automatisch beendet, wenn sich die Validierungsgenauigkeit über mehrere Epochen nicht mehr verbessert.

Bewertung

Beim neuronalen Netz fällt die Genauigkeit geringer aus als bei dem Random Forest Modell. Es erreicht eine Genauigkeit zwischen 60-80%. Dies kann zum einen daran liegen, dass die Datenmenge zu klein ist, um ein Muster zu finden, was gut auf andere Daten generalisiert werden kann. Die geringe Anzahl an Eingangsmerkmalen grenzt die Lernmöglichkeiten des Netzes zusätzlich ein. Trotz des Dropouts und Early Stopping tritt Overfitting auf, also dass das Netz die Trainingsdaten genau lernt und daher schlecht auf neue Daten richtig zuordnet. Die Hyperparameter wurden zwar durch Test verbessert, aber auch hier kann das Modell mit mehr Zeit auch noch weiter verbessert werden.