Klassifizierungsprojekt für Pinguine mittels Random Forest

Projektübersicht

Dieses Projekt zielt darauf ab, Klassifizierungsmodelle zu erstellen, um Pinguinarten anhand von physischen Messungen aus dem Palmer-Pinguin-Datensatz zu identifizieren. Es unterstützt sowohl Random Forest- als auch Decision Tree-Klassifikatoren, was einen Vergleich zwischen Ensemble-Methoden und interpretierbaren Einzelbaum-Ansätzen ermöglicht.

Inhalt

Datenbeschreibung

Der Datensatz enthält die folgenden Merkmale:

- **species**: Zielvariable, die die Pinguinart repräsentiert (Adelie, Gentoo, Chinstrap)
- **island**: Die Insel, auf der der Pinguin lebt (Biscoe, Dream, Torgersen)
- **culmen_length_mm**: Schnabellänge in Millimetern
- **culmen_depth_mm**: Schnabelhöhe in Millimetern
- **flipper_length_mm**: Flossenlänge in Millimetern
- **body_mass_g**: Körpermasse in Gramm
- **sex**: Geschlecht des Pinguins

Datenvorverarbeitung

- Fehlende Werte werden entfernt.
- Kategorische Variablen island und sex werden mittels One-Hot-Encoding in numerische Merkmale umgewandelt.
- Die Originaldatendateien bleiben unverändert; die Vorverarbeitung erfolgt in einer dedizierten Funktion innerhalb des Moduls one_hot_encoder.py.

Originaldaten:

species	island	culmen_length_mm	culmen_depth_mm	flipper_length_mm	body_mass_g	sex
Adelie	Torgersen	39.1	18.7	181.0	3750.0	MALE
Adelie	Torgersen	39.5	17.4	186.0	3800.0	FEMALE
Adelie	Torgersen	40.3	18.0	195.0	3250.0	FEMALE
Adelie	Torgersen	NaN	NaN	NaN	NaN	NaN
Adelie	Torgersen	36.7	19.3	193.0	3450.0	FEMALE

Bereinigte und kodierte Daten:

Merkmals-Tabelle (Features):

Index	culmen_length_mm	culmen_depth_mm	flipper_length_mm	body_mass_g	island_Biscoe	island_Dream	island_Torgersen	sex_.	sex_FEMALE	sex_MALE
0	39.1	18.7	181.0	3750.0	False	False	True	False	False	True
1	39.5	17.4	186.0	3800.0	False	False	True	False	True	False
2	40.3	18.0	195.0	3250.0	False	False	True	False	True	False
4	36.7	19.3	193.0	3450.0	False	False	True	False	True	False
5	39.3	20.6	190.0	3650.0	False	False	True	False	False	True

Ziel-Tabelle (Target):

Index	species
0	Adelie
1	Adelie
2	Adelie
4	Adelie
5	Adelie

Modulare Projektstruktur

- one_hot_encoder.py: Enthält die Funktion preprocess_penguin_data(filepath), die den Datensatz lädt, bereinigt und kodiert.
- data_inspector.py: Lädt die verarbeiteten Daten mithilfe des Encoders und stellt Dienstprogramme zur Dateninspektion bereit (z. B. Anzeige der ersten Zeilen).
- random_forest.py: Implementiert Modelltraining, Vorhersage, Evaluierung und Visualisierung mittels Random Forest.
- decision_tree.py: Implementiert Modelltraining, Vorhersage, Evaluierung und Visualisierung mittels Decision Tree.

Machine-Learning-Workflow

1. Laden und Vorverarbeiten der Daten mit preprocess_penguin_data.
2. Aufteilen der Daten in Trainings- und Testsets.
3. Trainieren eines Klassifikators (Random Forest oder Decision Tree) mit dem entsprechenden Skript.
4. Vorhersage auf den Testdaten.
5. Evaluierung der Leistung mittels Genauigkeit (Accuracy), Präzision (Precision), Trefferquote (Recall), F1-Score und Visualisierung der Ergebnisse mit einer Konfusionsmatrix.

Klassifikator-Optionen

- Random Forest (random_forest.py): Eine Ensemble-Methode, die die Vorhersagegenauigkeit und Robustheit verbessert.
- Decision Tree (decision_tree.py): Ein einfaches, interpretierbares Modell, das eine einzelne Baumstruktur verwendet.

Beide Klassifikatoren werden unterstützt; wählen Sie durch Ausführen des entsprechenden Skripts aus, welcher trainiert und evaluiert werden soll.

Modellleistung

- Random Forest erreicht eine hohe Genauigkeit (~99%).
- Decision Tree bietet interpretierbare Ergebnisse bei leicht geringerer Genauigkeit.
- Ausgewogene Klassifizierungsleistung über alle Pinguinarten hinweg bei beiden Modellen.

Visualisierung

Visualisierung der Konfusionsmatrix mittels sklearn.metrics.ConfusionMatrixDisplay, um wahre vs. vorhergesagte Klassifizierungen anzuzeigen.

Installation und Anforderungen

- Python 3.12.4 (empfohlen)

Erforderliche Python-Bibliotheken (Installation über pip):

- pandas
- numpy
- scikit-learn
- matplotlib

Wie man das Modell einrichtet

1. Erstellen einer virtuellen Umgebung (Virtual Env)

```
python -3.12 -m venv venv
```

2. Aktivieren der venv

```
.\venv\Scripts\activate
```

3. Erforderliche Pakete installieren

```
pip install -r requirements.txt
```

4. Streamlit-App starten

```
streamlit run .\Live_Demo.py
```

Mitwirkende

- Bendix Greiner
- Maurice Baumann
- Pascal Grimm