Effiziente Klassifikation: Methoden und Ergebnisse

Ein datenwissenschaftliches Projekt zur Typisierung von Antworten

Erstellt von:

Frédéric C. Kurbel, Abdul J. Safi

Studiengang: Computational and Data Science

Modul: Einführung in Data Science

Professor: Corsin Capol

Abgabedatum: 12.12.2024

Fachhochschule Graubünden

Inhaltsverzeichnis

1 Einleitung

Dieses Projekt untersucht die Klassifikation von Antworttypen basierend auf einem gegebenen Datensatz. Der Workflow umfasst die Datenvorverarbeitung, Feature-Auswahl, den Einsatz verschiedener Klassifikationsmodelle sowie die Optimierung der Modellleistung. Im Mittelpunkt steht der Vergleich der Ergebnisse von Random Forest, XGBoost, LightGBM und logistischer Regression, unter Anwendung von Techniken wie Hyperparameter-Optimierung und SMOTE.
Der vollständige Quellcode sowie eine detaillierte Dokumentation des Projekts sind im zugehörigen GitHub-Repository verfügbar:

`https://github.com/Fredeys/1_Semester_MathE_Learning_Classification`

2 Datenbeschreibung

Der verwendete Datensatz stammt aus dem *UCI Machine Learning Repository* und trägt den Titel **Dataset for Assessing Mathematics Learning in Higher Education**. Er wurde im Rahmen des *MathE-Projekts* entwickelt und umfasst **9.546 Antworten** zu mathematischen Themen, die in der Hochschulbildung gelehrt werden. Die Daten wurden zwischen Februar 2019 und Dezember 2023 gesammelt.

2.1 Spaltenbeschreibung und Datentypen

- **Student ID (int64):** Eindeutige ID für jeden Studenten.
- **Student Country (object):** Herkunftsland der Studenten.
- **Question ID (int64):** Eindeutige ID für jede Frage.
- **Type of Answer (int64):** Klassifikationsziel – korrekt oder inkorrekt.
- **Question Level (object):** Schwierigkeitsgrad der Frage – *basic* oder *advanced*.
- **Topic (object):** Übergeordnetes mathematisches Thema.

- **Subtopic (object):** Spezifisches Unterthema.
- **Keywords (object):** Schlüsselwörter, die mit der Frage verbunden sind.

2.2 Fehlende Werte

Es wurden keine fehlenden Werte in den Spalten festgestellt.

2.3 Besonderheiten des Datensatzes

- Die Spalten *Student Country*, *Question Level*, *Topic*, *Subtopic* und *Keywords* enthalten kategorische Werte, die numerisch konvertiert werden mussten.
- Der ursprüngliche Datensatz enthielt **2.083 doppelte Zeilen**, die nicht vor der Modellierung entfernt wurden, da sonst ein erheblicher Anteil der Daten verloren geht.

2.4 Verfügbarkeit

Weitere Informationen und der ursprüngliche Datensatz sind öffentlich im *UCI Machine Learning Repository* verfügbar: `https://archive.ics.uci.edu/dataset/1031/dataset+for+assessing+mathematics+learning+in+higher+education`. Der verarbeitete Datensatz `processed_dataset.csv` wurde für diese Arbeit genutzt und ist bereit für maschinelle Lernverfahren.

3 Methodik

3.1 Modellierung und Optimierung von Klassifikatoren

3.1.1 Vorbereitung der Daten

- **Datensatz:** Der Datensatz `MathE dataset.csv` enthielt 9.546 Einträge mit 8 Spalten, die durch Bag-of-Words-Transformation auf 226 erweitert wurden.

- **Kombination der Spalten *Subtopic* und *Keywords*:** Die Textfelder wurden kombiniert, um eine umfassendere Darstellung der Textdaten zu gewährleisten.
- **Bag-of-Words-Methode:** Die kombinierten Textdaten wurden numerisch repräsentiert, was zu einer erheblichen Erhöhung der Dimensionalität führte.
- **Kodierung:** Kategorische Variablen (*Student Country*, *Question Level*, *Topic*) wurden numerisch kodiert.
- **Datenaufteilung:** 80% der Daten wurden für das Training und 20% für Tests verwendet.

3.1.2 Modellvergleich und Optimierung

Random Forest

- **Baseline-Modell:** Erste Trainingsgenauigkeit von **59.3%**.
- **Hyperparameter-Tuning mit GridSearchCV:**
  - Beste Parameter:
    - *max_depth*: None
    - *min_samples_leaf*: 2
    - *min_samples_split*: 10
    - *n_estimators*: 100
  - Bestes F1-Score (Macro): **0.6129**.
- **Optimiertes Modell:** Nach Training mit optimierten Parametern erzielte der Random Forest eine Genauigkeit von **61.6%**.

Logistische Regression

- **Baseline-Modell:** Trainingsgenauigkeit von **54.8%**.
- **Optimierungen:**

  - *Feature-Skalierung*: Standardisierung der Features.
  - *SMOTE*: Balancierung der Klassenverteilung.
  - *Klassen-Gewichtung*: Berücksichtigung ungleicher Klassenhäufigkeiten.
- **Ergebnisse:** Optimierte Genauigkeit von **54.8%** mit verbessertem Recall für unterrepräsentierte Klassen.

XGBoost

- **Baseline-Modell:** Erste Genauigkeit von **64.5%**.
- **Optimierungen:**
  - *Randomized Search*: Beste Parameter:
    - *subsample*: 1.0
    - *n_estimators*: 200
    - *max_depth*: 5
    - *learning_rate*: 0.2
    - *colsample_bytree*: 0.6
  - *Feature-Reduktion*: Auswahl der wichtigsten Features.
  - *SMOTE*: Verbesserung der Klassenbalance.
  - *Kombinierte Optimierungen*: Klassen-Gewichtung und Feature-Reduktion.
- **Ergebnisse:** Genauigkeit von **65.7%** nach kombinierten Optimierungen.

LightGBM

- **Baseline-Modell:** Erste Genauigkeit von **64.0%**.
- **Optimierungen:**
  - *Hyperparameter-Tuning mit GridSearchCV*:
    - Beste Parameter:
      - *learning_rate*: 0.1

        - *min_child_samples*: 50
        - *n_estimators*: 200
        - *num_leaves*: 50
    - *Feature-Reduktion*: Schwellenwert von 8 für Feature-Importances führte zu 45 relevanten Features.

- **Ergebnisse:**
  - Optimierte Genauigkeit mit allen Features: **64.0%**.
  - Genauigkeit nach Feature-Reduktion: **65.0%**.
  - Durchschnittliche Genauigkeit aus 5-facher Kreuzvalidierung: **57.0%**.

3.2 Ergebnisse und Erkenntnisse

- **Beste Leistung:** XGBoost und LightGBM erreichten die höchste Genauigkeit (**65.7%** und **65.0%**).
- **Feature-Engineering:** Die Reduktion der Features führte zu stabileren und besseren Ergebnissen, insbesondere bei LightGBM und XGBoost.
- **Datenbalancierung:** SMOTE verbesserte Recall und F1-Score für unterrepräsentierte Klassen, jedoch teilweise auf Kosten der Gesamtgenauigkeit.
- **Hyperparameter-Tuning:** Besonders effektiv bei XGBoost und LightGBM, da eine präzise Feinabstimmung der Modelle möglich war.

3.3 Zusammenfassung

Die beste Modellleistung wurde mit dem **XGBoost-Klassifikator** erreicht, der durch eine Kombination aus **Feature-Auswahl**, **Datenbalance** und **Hyperparameter-Tuning** optimiert wurde. Diese Techniken haben entscheidend dazu beigetragen, eine hohe Genauigkeit zu erzielen, während die Balance zwischen **Präzision** und **Generalisierbarkeit** gewahrt blieb.

4 Fazit

Dieses Projekt zeigt deutlich, dass **Datenvorbereitung**, **Feature-Auswahl** und **Modelloptimierung** entscheidende Schritte sind, um die Leistung von Klassifikationsmodellen nachhaltig zu verbessern. Die Ergebnisse unterstreichen, dass unterschiedliche Algorithmen spezifische Stärken in Bezug auf verschiedene Datenstrukturen aufweisen, was die Relevanz einer modellabhängigen Strategie hervorhebt.

Besonders der **XGBoost-Klassifikator** hat sich als robust und effizient erwiesen. Durch die Kombination mit Techniken wie **SMOTE** zur Datenbalance und einer gezielten Hyperparameter-Optimierung konnte sowohl die Vorhersagequalität als auch die Stabilität des Modells gesteigert werden.

Ausblick

Um die Ergebnisse weiter zu verbessern und die Übertragbarkeit der Modelle zu prüfen, bieten sich folgende Ansätze an:

- **Integration zusätzlicher Merkmale:** Die Aufnahme weiterer relevanter Features könnte zusätzliche Informationen liefern und die Modellgenauigkeit erhöhen.
- **Erweiterung der Algorithmenvielfalt:** Die Evaluierung alternativer Methoden wie neuronaler Netze oder Ensemble-Ansätze kann neue Erkenntnisse und potenzielle Leistungssteigerungen ermöglichen.
- **Transferierbarkeit und Generalisierbarkeit:** Die Validierung der Modelle auf externen Datensätzen würde ihre Anwendbarkeit auf andere Domänen überprüfen und die Robustheit erhöhen.
- **Automatisierung des Optimierungsprozesses:** Der Einsatz von AutoML-Frameworks könnte die Effizienz und Genauigkeit durch automatisierte Optimierungsschritte weiter steigern.

Quellenverzeichnis

[1] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 2623–2631, 2019.

[2] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[3] Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. Smote: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16:321–357, 2002.

[4] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 785–794, 2016.

[5] D. Dua and C. Graff. Uci machine learning repository [dataset for assessing mathematics learning in higher education], 2019.

[6] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. In *Advances in Neural Information Processing Systems*, volume 30, pages 3146–3154, 2017.

[7] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, and Édouard Duchesnay. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.