Projektbericht KI

Bayes-Netz

Seminararbeit

für den Studiengang

Informatik

an der

Dualen Hochschule

Baden-Württemberg

Stuttgart

von

Oliver Rudzinski, Dominik Stiller

Kurs	TINF17A
Fach	Künstliche Intelligenz

Oliver Rudzinski	
Matrikelnummer	5481330
Email	oliver.rudzinski@hpe.com

Dominik Stiller	
Matrikelnummer	4369179
Email	dominik.stiller@hpe.com

1 Einleitung

Modelle der echten Welt, wie sie in Anwendungen der Künstliche Intelligenz und wissensbasierten Systemen zu finden sind, können nie exakt die Realität abbilden. Durch Messfehler, Subjektivität oder schlichtweg fehlende Kenntnis von Parametern entsteht eine unscharfe Abbildung, die entsprechend behandelt werden muss. Neben der Evidenztheorie von Arthur P. Dempster bieten sich dafür vor allem probabilistische Methoden wie Bayes-Netze an. Diese kennen Fakten nicht nur in binärer Logik, sondern können Werte zwischen wahr und falsch mit statistischen Methoden modellieren.

Dieser Projektbericht beschreibt den Entwurf und die Umsetzung eines Programms zur Vorhersage eines Versicherungstarifs basierend auf Personendaten mithilfe eines Bayes-Netzes und bewertet die Ergebnisse anschließend.

2 Entwurf

Für diesen Programmentwurf ist ein Datensatz mit Angaben zu Personen gegeben, von denen 5 Spalten kategorisch (Geschlecht, Familienstand, Bildungsstand, Beruf, Immobilienbesitz) und 3 Spalten numerisch (Alter, Kinder, Jahresgehalt) sind. Der Datensatz enthält außerdem den vergebenen Versicherungstarif (Tarif A–B oder abgelehnt) zu diesen Personen. Zur Verwendung in einem Bayes-Netz müssen numerische Spalten durch *binning* in Kategorien eingeteilt werden, um darauf Wahrscheinlichkeitstabellen erstellen zu können. Die Wahl der Grenzen zwischen den Kategorien ist eine Abwägung zwischen Verallgemeinerung (wenige Kategorien) und Aussagekraft (viele Kategorien). Dafür haben wir die Daten visuell in einem Jupyter-Notebook (zu finden unter `data/exploration.ipynb`) untersucht und uns für folgende Einteilung entschieden:

- Alter: 0–20, 20–40, 40–80, 80–100
- Jahresgehalt (in Tausend): 0–20, 20–40, 40–60, 60–80, 80–100, 100–120, 120–140
- Kinder: 0, 1, 2, 3, 4, 5, 6, 7, 8

Die Verwendung des Bayes-Netzes soll in einer Klasse `InsuranceAdvisor` verpackt werden, um die Schnittstelle einfach zu halten und das domänenspezifische Wissen über die Versicherungsdaten zu kapseln. Die Klasse braucht folgende Methoden:

- `load_data`: Daten aus .csv-Datei laden, validieren und transformieren

- `fit`: Modell mit Trainingdaten trainieren
- `predict`: Vorhersage für Versicherungstarif anhand des Modells treffen
- `predict_probabilities`: Vorhersage für Wahrscheinlichkeit aller Versicherungstarife treffen
- `save_model`: Modell in Datei speichern
- `load_model`: Modell aus Datei laden

Dies entspricht auch der Schnittstelle für Classifier in `scikit-learn`, somit können Bewertungsmethoden aus diesem Package verwendet werden.

3 Implementierung

Zur Umsetzung wählen wir Python 3, da es hierfür eine Vielzahl an Bibliotheken gibt, um schnell eine funktionierende Lösung zu erstellen. Insbesondere verwenden wir das Package `pomegranate`, das eine Reihe an probabilistischen Modellen inklusive Bayes-Netzen enthält. `pomegranate` kann die Graph-Struktur und Conditional Probability Tables eines solchen Netzes von Beispieldaten lernen und für diverse Formen probabilistischer Inferenz (diagnostisch, kausal, interkausal, gemischt) anwenden. Dabei werden notwendige Bedingungen wie Azyklizität und P-Separierbarkeit eingehalten. Wir verwenden die A*-Variante[1]. Die Benutzung des Packages wird dann in der Klasse `InsuranceAdvisor` gekapselt.

Zusätzlich gibt es ein Skript, das den gesamten Datensatz in Trainings- und Testdaten zufällig im Verhältnis 70:30 aufteilt und in separaten Dateien im Ordner `data/` speichert. Dabei wird sichergestellt, dass in den Trainingsdaten jeder Wert für die Spalte Kinder einmal vorkommt. Andernfalls treten Probleme auf, wenn eine Kategorie dem Netz nicht bekannt ist. Außerdem werden die ersten 10 Testdaten ohne Versicherungstarif in einer weiteren Datei gespeichert, um Inferenz testen zu können.

[1] `https://github.com/jmschrei/pomegranate/blob/master/pomegranate/BayesianNetwork.pyx#L1558-L1646`

4 Verwendung

Zur Verwaltung von Abhängigkeiten und der Python-Umgebung verwenden wir `pipenv`, welches wie hier[2] beschrieben installiert werden kann. Dann kann die Umgebung per `pipenv install --skip-lock` eingerichtet und per `pipenv shell` im obersten Programmordner aktiviert werden. Unsere Lösung kann dann per Aufruf der folgenden Befehle in dieser Shell ausgeführt werden. Sie bietet drei Modi.

4.1 Training

Aufruf: `python insurance_advisor train [train-data]`
Beispiel: `python insurance_advisor train data/dataset_train.csv`
Daten: Features + Versicherungstarif
Mit diesem Befehl wird das Bayes-Netz aus den Trainingsdaten erlernt und im Ordner `model/` gespeichert. Die Datei mit Trainingsdaten muss dabei alle Spalten enthalten. Außerdem wird die Modell-ID ausgegeben, die in den anderen Modi verwendet werden kann.

4.2 Vorhersage

Aufruf: `python insurance_advisor predict [model-id] [predict-data]`
Beispiel: `python insurance_advisor predict default data/dataset_predict.csv`
Daten: Features
Mit diesem Befehl wird der Versicherungstarif für neue Daten vorhergesagt und mit einer Wahrscheinlichkeit als Tabelle ausgegeben. Dafür wird wieder die Modell-ID eines vorher trainierten Modells, wie etwa aus dem letzten Schritt, und der Pfad zu einer Datei mit Daten angegeben. Die Daten müssen alle Spalten enthalten, allerdings dürfen davon beliebige Einträge leer sein.

4.3 Evaluation

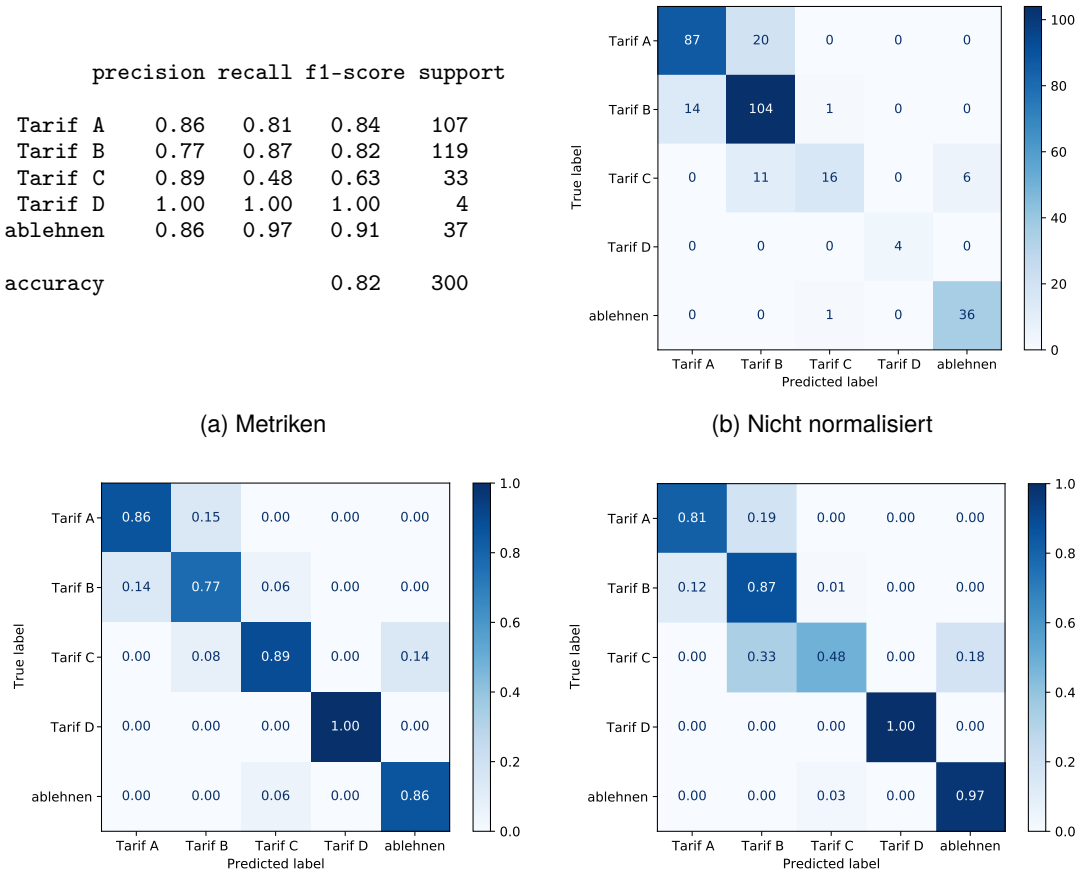
Aufruf: `python insurance_advisor evaluate [model-id] [test-data]`
Beispiel: `python insurance_advisor evaluate default data/dataset_test.csv`
Daten: Features + Versicherungstarif
Mit diesem Befehl werden Testdaten vorhergesagt und die vorhergesagten und wahren

[2] `https://pipenv.kennethreitz.org/en/latest/install/#installing-pipenv`

Versicherungstarife verglichen. um Klassifikationsmetriken und eine Confusion Matrix zu generieren. Dafür wird die Modell-ID eines vorher trainierten Modells und der Pfad zu einer Datei mit Testdaten angegeben.

5 Bewertung

Zur Bewertung der Klassifikation haben wir unser Modell mit den Testdaten aus dem Train/Test-Split mit dem Befehl aus Abschnitt 4.3 ausgeführt. Die Metriken sind in Abbildung 1a dargestellt. Außerdem sind die Ergebnisse in Form von drei Confusion Matrices abgebildet. Abbildung 1b ist eine übliche Confusion Matrix mit absoluten Angaben. Abbildung 1c und 1d zeigen die selben Daten wie 1b, allerdings pro Kategorie über die wahren Labels beziehungsweise über die vorhergesagten Labels normalisiert. Dadurch sind Pre-



	precision	recall	f1-score	support
Tarif A	0.86	0.81	0.84	107
Tarif B	0.77	0.87	0.82	119
Tarif C	0.89	0.48	0.63	33
Tarif D	1.00	1.00	1.00	4
ablehnen	0.86	0.97	0.91	37
accuracy			0.82	300

(a) Metriken

(b) Nicht normalisiert

(c) Normalisiert über wahres Label
Precision auf der Diagonalen

(d) Normalisiert über vorhergesagtes Label
Recall auf der Diagonalen

Abbildung 1: Klassifikationsergebnisse

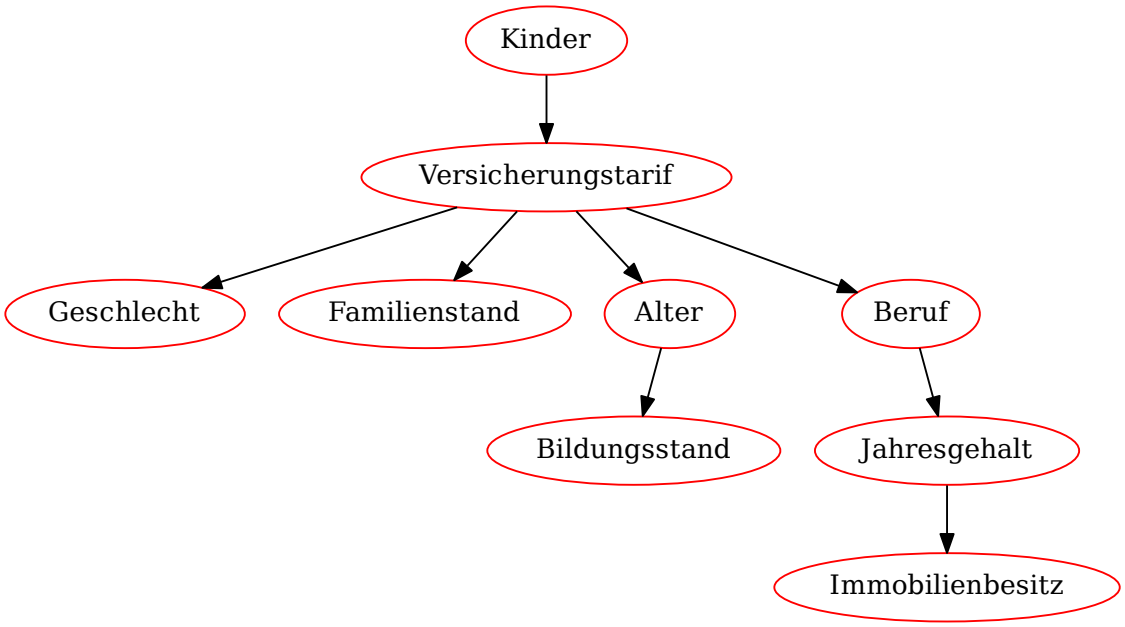


Abbildung 2: Grafische Darstellung des Bayes-Netzes

cision beziehungsweise Recall auf der Diagonalen ablesbar. Durch die Normalisierung lassen sich die Versicherungstarife untereinander vergleichen, obwohl unterschiedliche viele Beispiele für verschiedene Tarife vorliegen.

Wir erreichen eine Accuracy, also einen Anteil insgesamt korrekt klassifizierter Einträge, von 82%. Etwa in diesem Bereich bewegen sich auch Precision, Recall und F1-Scores aller Tarife. Nur Tarif C hat mit 48% einen deutlich schlechteren Recall, denn er wird in 33% aller Fälle als Tarif C klassifiziert. Auch zwischen Tarif A und B gibt es einige Verwechslungen. Tarif C, Tarif D und Ablehnungen sind in den Testdaten allerdings nur wenig vorhanden, daher ist die Signifikanz der Ergebnisse für diese Kategorien eher gering.

Der von `pomegranate` automatisch generierte Graph des Bayes-Netzes ist in Abbildung 2 dargestellt. Es ist deutlich zu sehen, dass der Versicherungstarif zentral liegt und direkt von fünf Features abhängt. Während Zusammenhänge wie Beruf $\rightarrow$ Jahresgehalt $\rightarrow$ Immobilienbesitz logisch plausibel sind, wäre man manuell wohl nicht auf diese Graph-Struktur gekommen.

6 Zusammenfassung

In diesem Bericht haben wir unseren Programmentwurf zur Vorhersage von Versicherungstarifen mit Bayes-Netzen vorgestellt. Dabei nutzen wir das Erlernen der Graph-Struktur und Wahrscheinlichkeitsverteilungen durch `pomegranate` und erreichen zufriedenstellende Ergebnisse. Das Programm ist über eine Kommandozeilen-Schnittstelle zu bedienen.

Das Modell könnte in einem realen Szenario als Empfehlung für einen menschlichen Berater dienen, allerdings sollte es nicht selbstständig für Entscheidungen verantwortlich sein. Außerdem sollte es auf Voreingenommenheit gegenüber diversen Personengruppen geprüft werden. Andere Methoden des maschinellen Lernens wie Random Forests, Gradient Boosted Trees oder Support Vector Machines könnten möglicherweise bessere Ergebnisse liefern und sollten als Alternativen in Erwägung gezogen werden.