h_da
HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES
fbmn
FACHBEREICH MATHEMATIK
UND NATURWISSENSCHAFTEN

Gruppe 14
Tamara Döß (767956)
Tobias Rohrer (???)

Data Mining 1

Booklet von Gruppe 14

Inhaltsverzeichnis

Zusammenfassung

1 Aufgabe 1: Entscheidungsbäume

1.1 Feature Engineering

1.2 Analyse der Zielvariable

Wie Abbildung 1 entnommen werden kann ist Ausprägung der Zielvariable ungleich auf beide Klassen verteilt. Die Klassifizierungsgenauigkeit eines Modells muss demnach unter Berücksichtigung der sogenannten *Null Accuracy* bewertet werden. Unter *Null Accuracy* versteht man die Genauigkeit eines Modells, dass unabhängig von allen Eingaben immer die am häufigsten auftretende Klasse vorhersagt. In unserem Fall würde ein Modell, welches immer Regen vorhersagt, eine Klassifizierungsgenauigkeit von 79,39% erreichen. Das Ziel der nachfolgenden Schritte ist also ein Modell mit einer besseren Klassifizierungsgenauigkeit als die *Null Accuracy* aufzubauen.

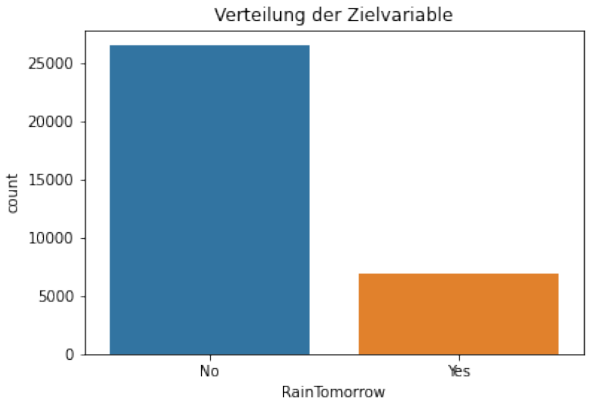

Abbildung 1: Verteilung der Zielvariable

1.2.1 Fehlende Werte

Der zu untersuchende Datensatz beinhaltet fehlende Werte. Diese müssen für eine weitere Verarbeitung bereinigt werden. Im Folgenden werden Methoden beschrieben, wie mit den fehlenden Werte umgegangen wurde:

Fehlende Zielvariable
Im ersten Schritt wurden 834 Beobachtungen, welche keinen Wert für die Zielvariable *RainTomorrow* aufweisen, aus dem Datensatz entfernt. Damit wurde die Anzahl an Beobachtungen auf 33402 reduziert.

Spalten mit fehlenden Werten
In einem nächsten Schritt werden die Spalten aus dem Datensatz entfernt, in denen mehr als 40% der beinhaltenden Variablen fehlen. Namentlich wurden somit die Spalten *Evaporation, Sunshine, Cloud9am* sowie *Cloud3pm* aus dem Datensatz entfernt. Der Schwellwert von 40% wurde empirisch festgelegt und hat zu den besten Klassifizierungsergebnissen geführt.

Beobachtungen mit fehlenden Werten
Des Weiteren werden Beobachtungen aus dem Datensatz entfernt, von denen mehr als 50%

der Variablen fehlen. Durch diesem Schritt wurden 55 Beobachtungen aus dem Datensatz entfernt.

Imputation

Durch die zuvor beschriebenen Methoden ist der Datensatz noch immer nicht frei von fehlenden Werten. Um diese zu ersetzen werden für kategorische und numerische Variablen verschiedene Strategien zur Imputation verfolgt. Fehlende numerische Werte werden mit dem Median der jeweiligen Variable ersetzt. Der Median wurde gewählt, da dieser im Vergleich zum Mittelwert robuster gegenüber Ausreißern ist. Für kategorielle Variablen hingegen wird der am häufigsten vorkommende Wert verwendet. Wichtig bei der Ermittlung des Medians bzw. des häufigsten Wertes ist, dass dieser ausschließlich mit Hilfe der Trainingsdaten ermittelt wird. Es muss davon ausgegangen werden, dass die Testdaten nicht bekannt sind. Die Ermittlung auf Basis des gesamten Datensatzes, inklusive der Testdaten, würde zu *Data-Leakage* führen und ist zu vermeiden. Die auf Basis der Trainingsdaten ermittelten Werte für die Imputation werden auf die Trainings- und Testdaten angewendet.

1.2.2 Merkmalsaufteilung

Das Feld *Datum* wurde in die Merkmale *Year, Month* und *Day* aufgeteilt.

1.2.3 Merkmalserstellung

Eine weit verbreitete Technik des Feature Engineerings ist die Erstellung zusätzlicher Merkmalen. Somit wurde die Variable *MinMaxDiff* erstellt, welche die Differenz zwischen der minimalen und der maximalen Tages-Temperatur angibt. In gleicher Weise wurden die Variablen *PressureDiff, HumidityDiff und WindSpeedDiff* erstellt.

1.2.4 Diskretisierung

Die Diskretisierung eines Merkmals kann eine Überanpassung bei der Erstellung von Modellen verhindern, indem der Wertebereich des Merkmals minimiert und somit generalisiert wird. Es sollte beachtet werden, dass der Informationsverlust durch die Diskretisierung nicht zu hoch ausfällt.

Zu Testzwecken wird das neu erstellte Merkmal *Month* in zwei weiteren Merkmalen diskretisiert. Zum einen wird das kategorische Merkmal *Season* erstellt, welches angibt, in welcher Jahreszeit der Monat liegt. Zum anderen gibt das Boolean Merkmal *RainyMonth* an, ob es sich um einen in Erwartung regenreichen Monat handelt.

1.2.5 Kodierung kategorischer Werte

Um kategorische Werte für weitere Analysen verwenden zu können, müssen diese in numerische Werte umkodiert werden. Hierbei wurden die folgenden Strategien Angewendet:

Binäre Kodierung

Die Zielvariable *RainTomorrow*, sowie die Variable *RainToday* liegen mit den binären Ausprägungen *Yes* und *No* vor. Die binären Werte wurden in eine numerische Darstellung umgewandelt

One-Hot-Kodierung

Das neu diskretisierte Merkmal *Season* wird mittels *One-Hot Encoding* in drei Boolean

Spalten aufgeteilt. Eine einfache Zuteilung eines Zahlenwertes pro auftretender Variablenausprägung hat den Nachteil, dass dadurch eine metrische numerische Variable für ein gegebenenfalls nicht metrisch interpretierbares Merkmal entsteht. Für Entscheidungsbäume wäre dies kein Problem, jedoch sollte das Feature Engineering weitgehend unabhängig von dem verwendeten Klassifizierungsalgorithmus durchgeführt werden.

Ziel-Kodierung

Mit Hilfe des *Target Encodings* werden die Merkmale *Location, WindGustDir, WindDir9am* und *WindDir3pm* verarbeitet. Dabei werden den jeweiligen Merkmalsausprägungen ihr Einfluss auf die Zielvariable zugeordent, also die bedingte Wahrscheinlichkeit, dass unter der Bedingung, dass z.B. Merkmal *Location* für die Zielvariable (*Target*) den Wert 1 (in unsrem Fall "Yes") annimmt. Zum Beispiel wird die Merkmalsausprägung "Perth"mit dem Wert 0.2 kodiert, wenn 20% der jeweiligen Beobachtungen für die Zielvariable den Wert "Yesännehmen. Somit wird keine zufällige Zuteilung von nummerischen Werten verteilt, sondern direkt einen Verbindung zu Zielvariable hergestellt. Ein Nachteil des Target Encoding ist, dass genau durch diese Verbindung zur Zielvariable die Wahrscheinlichkeit des Overfittings steigt (vgl. https://towardsdatascience.com/why-you-should-try-mean-encoding-17057262cd0 keine Wissenschaftliche Quelle).

1.2.6 Bereinigung von Ausreißern

Ausreißer können die Performance eines Modells mindern, indem sie als Hebelwerte agieren und somit die Schätzungen der Zielvariable verzerren. Aus diesem Grund werden die Merkmale des Datensatz hinsichtlich ihrer Ausreißer begutachtet. Es wird ersichtlich, dass nur die Merkmale *Rainfall* und *WindGustSpeed* abweichende Werte aufweisen. Das Enfernen dieser Werte aus dem Datensatz führt jedoch zu einer schlechteren Performance der im folgenden Abschnitt besprochenen Entscheidungsbäume. Deshalb werden die Beobachtungen nicht aus dem Datensatz entfernt. Zudem zählen die beiden Merkmale zu denjenige Merkmalen, die durch die Variablenselektion aussortiert und somit für weitere Modelle nicht mehr betrachtet werden.

1.2.7 Variablenselektion

Nach dem durchgeführten Feature Engineering verfügt der Datensatz über 28 Einflussvariablen. Durch eine Variablenselektion wird diese Anzahl verringert. Das hat den Vorteil, dass Modelle besser interpretierbar sind. Zudem steigert es die Performance von Modellen, indem nicht relevante Merkmale ausgeschlossen werden, wodurch auch die Wahrscheinlichkeit des Overfittings sinkt.

Für den betrachteten Datensatz wird eine univariate Variablenselektion mit dem Modul *SelectKBest* der Bibliothek *sklearn* durchgeführt. Dabei werden die *k=5* Merkmale ausgewählt, die den höchsten Wert der F-Statistik aufweisen. Dieser Wert gibt an, ob ein einzelnes Merkmal eine signifikante Verbesserung des Modells erzielt. Dazu wird ein Modell ohne dieser Variable mit einem Modell mit der beobachteten Variable verglichen. Als Ergebnis wurden die Merkmale *WindGustDir, Humidity9am, Humidity3pm, RainToday* und *MinMaxDiff* ausgewählt.

1.3 Entscheidungsbäume

Für die Erstelltung der Entscheidungsbäume wurde der Datensatz in eine Trainings- und eine Testmenge eingeteilt. Dabei beträgt das Verhältnis 8:2. 80% der Daten sind genügend, um ein

gutes Modell zu trainieren. 20% sind eine ausreichende Menge, um einen genauen Testfehler und die Modellgüte zu bestimmen.

1.3.1 Default-Einstellungen

Quelle: https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html
Die Entscheidungsbäume werden mit dem Modul *tree.DecistionTreeClassifier* erstellt. Im ersten Aufgabenteil werden dazu die Default-Einstellungen des Moduls genutzt.

Diese geben an, dass ein Baum immer maximal gebaut wird ($max\_depth = None$). Das heißt, so lange mehr als ein Objekt in einem Knoten vorliegt, wird dieser Knoten weiter aufgespalten ($min\_samples\_split = 2$). Das Ergebnis ist überangepasst, da jeder einzelne Datenpunkt ein eigenes Blatt im Baum bekommt ($min\_samples\_leaf = 1$). Dieser Baum kann somit nicht auf unbekannte Daten generalisiert werden und der Testfehler fällt sehr hoch aus. Die Splits werden mit dem Gini-Wert durchgeführt und nicht mit der Entropie ($citerion$=„*gini*“). Zudem werden für jedes Spalten alle Merkmale einbezogen, um den besten *Split* zu finden ($max\_features = None$). In Abbildung 2 wird der mit den Default-Einstellungen erzeugte Baum dargestellt. Es ist leicht zu erkennen, dass dieser Baum zu viele Knoten und Blätter enthält

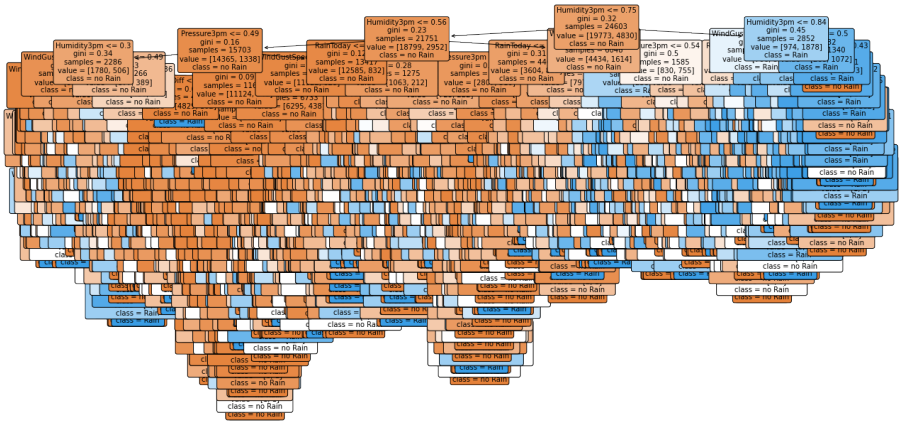

Abbildung 2: Entschidungsbaum mit Default-Einstellungen

und somit eine Überanpassung darstellt. Diese Überanpassung kann auch an dem Trainings- und Testfehler abgelesen werden. Der Trainingsfehler beträgt für den gebildeten Entscheidungsbaum 0.0005 und der Testfehler 0.2340. Der Testfehler ist damit höher als bei einem Modell, das zufällig entscheidet. Dem kann mit verschiedenen Einstellungen entgegengewirkt werden.

1.3.2 Variationen

Um einen übersichtlichen und brauchbaren Entscheidungsbam erzeugen zu können werden verschiedenen Einstellungen für die Parameter *max_ depth, min_ ipurity_ decrease* und *criterion* angewandt. Im Folgenden werden die Entscheidungsbäume dargestellt, deren Parameter mit Hilfe des **GridSearch Verfahrens** festgelegt wurden. Bei der Visualisierung liegt der Fokus auf der Struktur des Baumes und nicht darauf, dass die einzelnen Knoten identifiziert werden können.

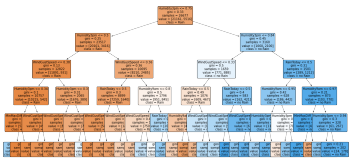

Abbildung 3: Maximale Tiefe von 5

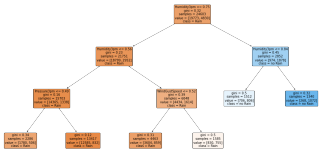

Abbildung 4: Minimale Unschärfe Reduktion von 0.003

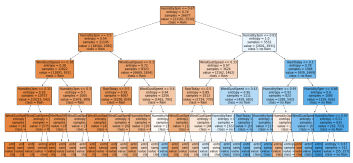

Abbildung 5: Entropie als Entscheidungskriterium

Maximale Tiefe

In Abbildung 3 wurde eine maximale Tiefe von **fünf** angegeben. Trotz der geringen Tiefe entstehen schon zu viele Blatt KNoten für eine übersichtliche Visualisierung. Im Gegensatz zu den anderen Variationen ist dieser Baum **symmetrisch**, da jeder Ast bis zur gleichen Tiefe verfolgt wird. Dieser Baum entspricht den ersten fünf Stufen des vorher erstellten Default-Entscheidungsbaum, da der *max_depth*-Parameter den Baum einfach an der gegebenen Stufe abschneidet.

Unschärfe-Reduktion

In Abbildung 4 wurde mit dem Parameter *min_impurity_decrease* festgelegt, wie viel ein weiterer Knoten die **Unschärfe** des Entscheidungsbaumes reduzieren muss, um ausgebildet zu werden. Dabei wird jeder zu bildende Knoten einzeln bewertet, sodass es zu einem nicht symmetrischen Baum kommen kann, da nicht jeder Ast bis in die gleiche Tiefe verfolgt wird.

Entscheidungskriterium

In Abbildungen 5 wurde zusätzlich zur maximalen Tiefe von fünf das Entscheidungskriterium angepasst. Hier wird mit Hilfe der Entropie statt des Gini-Wertes entschieden, welcher Split durchgeführt wird. Die Struktur des Baumes wird dadurch nicht geändert, jedoch können die Knoten voneinander abweichen. Die Entropie erzeugt bei sonst gleich bleibenden Parametern einen Entscheidungsbaum, der einen kleineren Testfehler aufweist.

1.3.3 Minimal Cost-Complexity-Pruning

2 Aufgabe 2: Neuronale Netze

2.1 Unterkapitel 1

2.1.1 Unterkapitel 1.1

2.2 Neuronale Netze mit TensorFlow

Im Folgenden wird die Implementierung eines voll vernetzen Neuronalen Netzwerks mit Hilfe der *TensorFlow* Bibliothek beschrieben **?**. Unterstützend wurde für den Aufbau des neuronalen Netzes die *Keras*-API verwendet, die seit *TensorFlow* Version 2 standardmäßig integriert ist. Um die Daten für das Training des Neuronalen Netzes effizient vorzubereiten wurde die *Dataset*-API verwendet. Mit Hilfe der *Dataset*-API wurde das zufällige Mischen der Trainingsdaten, sowie die Bereitstellung in Form von *Mini-Batches* implementiert. Die Verwendung der *Dataset*-API hat noch zusätzlich den Vorteil, dass die Ausführung von Vorbereitungsschritten der Daten perfekt mit dem Training des Neuronalen Netzes koordiniert und parallelisiert werden können.

2.2.1 Hyperparameter Suche

Um ein best Mögliches Modell für die vorliegenden Daten zu finden, wurde eine Hyperparameter-Suche implementiert.

3 Aufgabe 2: Ensemblemethoden

3.1 Unterkapitel 1

3.1.1 Unterkapitel 1.1

4 Aufgabe 4: Support Vector Machines

4.1 Unterkapitel 1

4.1.1 Unterkapitel 1.1

Anhang

Quellcode zu Aufgabe 1

Quellcode zu Aufgabe 2

Quellcode zu Aufgabe 3

Quellcode zu Aufgabe 4