ZHAW School of Engineering
Konzepte des maschinellen Lernens - KONZML

Elektrotechnik 2019

Projektarbeit zum Thema

Crop Recommendation with ML

Barnes Toby - barnetob

Seminar: KONZML
Fachsemester: Herbst 2021

Abgabedatum: 10.01.2022

Dozent: Weinmann Thomas Oskar (weto)

1 Einleitung

Technologien wie künstliche Intelligenz (KI), Robotik, das Internet der Dinge (IoT), Edge Computing, 5G und Blockchain haben alle das Potenzial, die Landwirtschaft effizienter, nachhaltiger und wettbewerbsfähiger zu machen. Die digitale Transformation der Landwirtschaft wird die Zusammenarbeit entlang der Wertschöpfungskette erleichtern, Landwirte unterstützen und Chancen für innovative KMUs bieten. Bei dieser Arbeit soll anhand eines Datensatzes [1] ein Machine Learning (ML) Algorithmus trainiert werden, um ein Vorhersagemodell zu erstellen, dass auf der Grundlage verschiedener Parameter, die am besten geeigneten Pflanzen für den Anbau auf einem Ackerfeld empfiehlt. Die Eingabeparameter bestehen aus den nachfolgenden Definitionen:

- **N** - Verhältnis des Stickstoffgehalts im Boden:
- **P** - Verhältnis des Phosphorgehalts im Boden
- **K** - Verhältnis des Kaliumgehalts im Boden
- **temperature** - Temperatur in Grad Celsius
- **humidity** - relative Feuchtigkeit in %
- **ph** - ph-Wert des Bodens
- **rainfall** - Niederschlag in mm

Es werden dabei 22 Pflanzen prognostiziert. (Reis, Mais, Kichererbse, Kidneybohnen, Taubenbohnen, Mottenbohnen, Mungbohne, Urdbohne, Linse, Granatapfel, Banane, Mango, Weintrauben, Wassermelone, Zuckermelone, Apfel, Orange, Papaya, Kokosnuss, Baumwolle, Jute, Kaffee)

2 ML – Pipeline

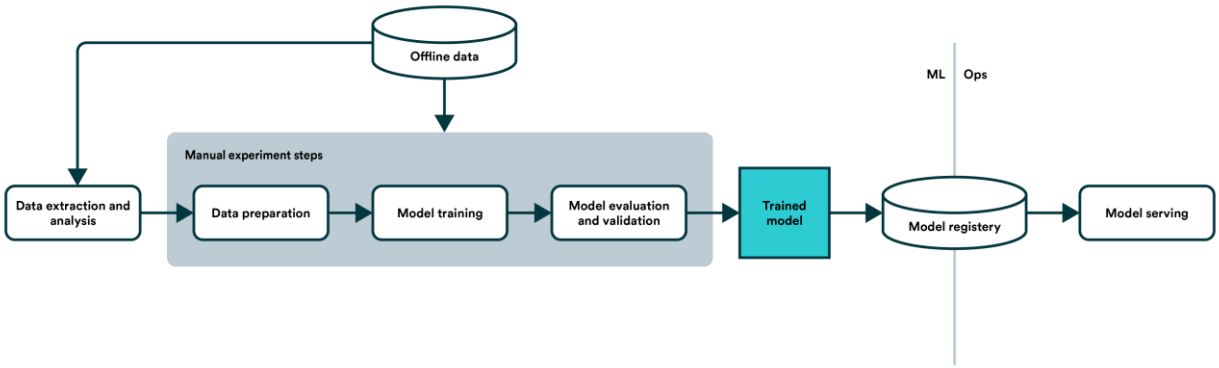


Abbildung 1 - Pipeline von einem ML-Model [2]

Dataset:

Der Datensatz wurde von Kaggle heruntergeladen und verwendet. Die Analyse dieses Datensatzes wurde im Python Skript PRE_crop näher angeschaut.

Data preparation:

Da der Datensatz gute Ergebnisse mit der Brute Force Methode beim Trainieren eines ML Netztwerkes erzeugt (siehe Kapitel 3.2) wurde entschieden, dass beim Datensatz keine zusätzliche Verarbeitung benötigt.

Model training

Das Trainieren des ML Modells befindet sich im Python Skript ML_crop. Dazu wurden zwei Modelle für die Evaluation und Deployment trainiert.

Model & validierung & evaluation:

Für die Evaluation wurde die Confusion Matrix verwendet welche ein Testdatensatzes benützt um die Klassifikation der Kategorien des Datasets zu überprüfen. Die Testdaten bestehen aus 30 % vom Datensatz, dazu wurden 70 % des Datensatzes verwendet um das ML Modells zu trainieren.

3 Vorverarbeitungsschritte

3.1 Vorevaluierung

3.1.1 Scatter Plots

Für die Vorevaluierung wurden einige Plots erstellt, um eine Visualisierung der Daten zu erhalten und um einen Eindruck zu erhalten, ob genug Abstraktion zwischen den Klassen vorhanden sein könnte. Um alle Variationen zu betrachten, wurde eine Methode namens Pairplot angewendet, dieser Plot befindet sich im Anhang.

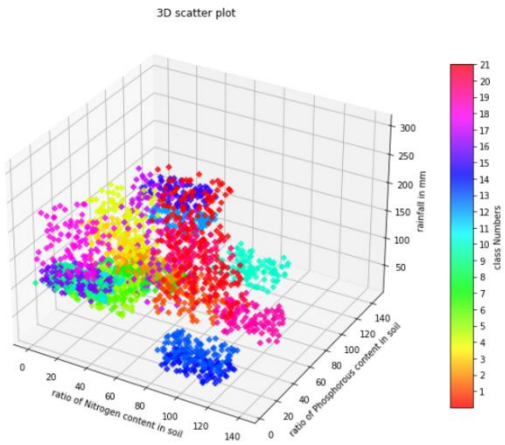
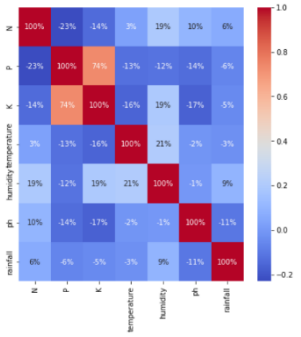


Abbildung 2 - 3D-Scatter Plot

Zusätzlich wurde als erstes jedoch einen 3D-Scatter Plot erstellt (siehe Abbildung 2). Es zeigt die Abhängigkeit zwischen dem Verhältnis Stickstoffgehalts, Phosphorgehalts und Niederschlag in mm im Erntefeld, zusätzlich werden mit der farblichen Visualisierung die Pflanzen dargestellt. Es wurde festgestellt, dass es eine gewisse Abstrahierung zwischen den verschiedenen Klassen gibt, ins besonders die Klasse 10 (türkis) kann gut von den anderen Klassen unterschieden werden. Da es jedoch sieben Eingabeparametern sind, die benützt wurden, um den Datensatz zu erzeugen, kann davon ausgegangen werden, dass es genug weitere Abstrahierung zwischen den Klassen vorhanden sein wird.

3.1.2 Korrelationmatrix



Eine Korrelationsmatrix ist eine Tabelle mit Korrelationskoeffizienten zwischen Variablen. Jede Zelle in der Tabelle zeigt die Korrelation zwischen zwei Variablen an. Der Sinn und Zweck dieser Matrix ist es, dass Muster in den Daten erkannt werden. Beim verwendeten Datensatz wurde in der generierten Korrelationsmatrix (siehe Abbildung 4) erkannt, dass der Phosphorgehalt und der Kaliumgehalt im Boden stark korrelieren. Jedoch gibt es kaum Korrelation zwischen den anderen Parametern.

Abbildung 3 - Korrelationmatrix

3.2 Modellselektion

```
Ridge -> Accuracy score: 75.00 %
RBF SVM -> Accuracy score: 98.03 %
Nearest Neighbors -> Accuracy score: 98.33 %
Naive Bays -> Accuracy score: 99.55 %
Decision Tree -> Accuracy score: 98.33 %
Random Forest -> Accuracy score: 99.85 %
Ada Boost -> Accuracy score: 15.61 %
QuadraticDiscriminantAnalysis -> Accuracy score: 99.09 %
```

Abbildung 4 - ML Modelle selektionieren

Um eine oder zwei ML Modelle anzuwenden, wurde mit einer Brute Force Methode bestimmt, welche Modelle schon, ohne irgendwelche Parameter zu übergeben, den besten Score hat. Man hätte einen Algorithmus wie den Z-Score auf den Datensatz anwenden können, da aber die ML Modelle ein erstaunlich gutes Ergebnis erbringt, wurde bestimmt, dass es keine Daten Vorverarbeitung des Datensatzes benötigt. Es wurde bestimmt, dass der Gaussian Naive Bayes und der Random Forest Algorithmus näher für die Umsetzung des Projektes angeschaut wird.

4 Umsetzung

4.1 ML – Model

4.1.1 Gaussian Naive Bays – Algorithmus

Es handelt sich um eine Klassifizierungstechnik, die auf dem Bayes'schen Theorem basiert, wobei von der Unabhängigkeit der Prädiktoren ausgegangen wird. Das Naive Bayes-Modell ist einfach zu erstellen und besonders nützlich für sehr grosse Datensätze. Neben seiner Einfachheit ist Naive Bayes dafür bekannt, dass es selbst hoch entwickelte Klassifizierungsmethoden übertrifft. [3]

4.1.2 Random Forest – Algorithmus

Random Forest ist ein Supervised Lernalgorithmus. Der Tree, den er aufbaut, ist ein Ensemble von Entscheidungsbäumen, die in der Regel mit der "Bagging"-Methode trainiert werden. Einfach ausgedrückt: Random Forest erstellt mehrere Entscheidungsbäume und fügt sie zusammen, um eine genauere und stabilere Vorhersage zu erhalten.
Ein grosser Vorteil von Random Forest ist, dass er sowohl für Klassifizierungs- als auch für Regressionsprobleme verwendet werden kann, die den Grossteil der aktuellen maschinellen Lernsysteme ausmachen. [4]

4.2 Evaluation mit Confusion Matrix

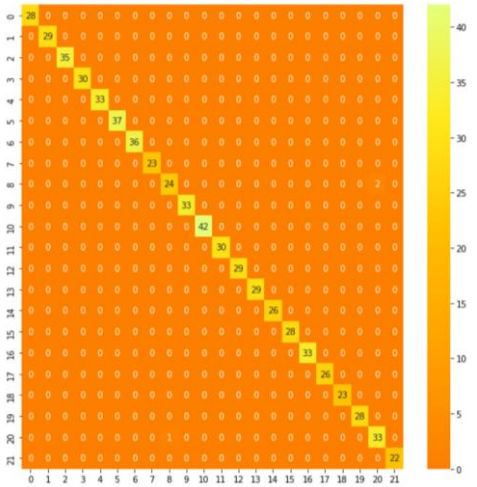


Abbildung 5 - Gaussian NB Evaluation

Mit dem Gaussian Naive Bayes konnte mit der Validierung der Testdaten eine Genauigkeit von 99.545 % erreicht werden. Bei der Validierung mit den Testdaten wurde festgestellt, dass der Algorithmus nur eine Klasse falsch prognostiziert, jedoch kam dies 2-mal vor. In diesem Fall wurde die Klasse 9 bzw. die Pflanze Linse vorhergesagt, jedoch in Wahrheit war es die Klasse 21 bzw. die Pflanze Jute. Da der Datensatz aus 7 Eingabeparametern besteht, wurde auf die Berechnung des Priors verzichtet. Dadurch kann nicht ausgesagt werden, ob man die Genauigkeit mit einem Prior erhöht hätte werden können.

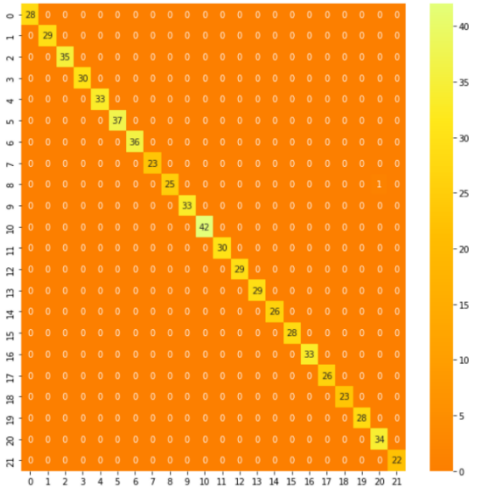


Abbildung 6 - Random Forest Evaluation

Mit dem ML-Model Random Forest konnte eine Genauigkeit von 99.848 % erreicht werden. Bei der Validierung mit den Testdaten wurde festgestellt, dass wie beim Gaussian Naive Bayes Algorithmus er dieselbe Klasse falsch prognostiziert, jedoch kam dies nur einmal vor, dies macht die Genauigkeit des Random Forest Algorithmus höher. Trotz mitgegeben Parametern wie max_feature oder class_weight konnte die Genauigkeit nicht auf 100 % erhöht werden.

4.3 Random Forest Decision Tree

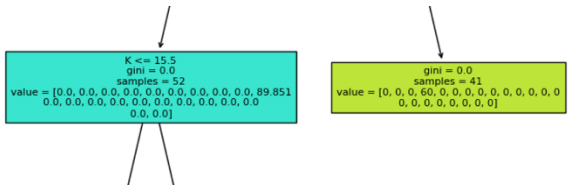


Abbildung 7 - Decision Tree Visualisierung

Der Decision Tree des Random Forest Algorithmus wurde mit einer Python Bibliothek visualisiert. Die Abbildung 7 zeigt einen kleinen Ausschnitt des Tree's. Der Tree besteht aus vierzehn Layern und einigen Zweigen.
Die komplette Darstellung befindet sich im Anhang.

5 Fazit

Die beiden ML-Algorithmen Gaussian Naive Bayes und Random Forest wurden in diesem Projekt implementiert, analysiert und verglichen. Fest steht, dass der Random Forest Algorithmus eine höhere Genauigkeit aufweist als der Gaussian Naive Bayes Algorithmus. Beide Algorithmen weisen jedoch eine sehr hohe Genauigkeit auf, welche auf den Datensatz zurückzuführen ist. Dieser weist eine hohe Genauigkeit mit wenig Messrauschen auf, dadurch konnte die hohe Genauigkeit erreicht werden. Dadurch musste keine Vorverarbeitung der Daten vorgenommen werden. Durch die Analyse der Testdaten durch die beiden verwendeten Algorithmen wurde festgestellt, dass die Klassen Linse und Jute gewisse Ähnlichkeiten in den Parametern aufweist, da beide Algorithmen genau bei dieser Label-Klassifikation eine oder mehrere Eingaben falsch schätzt.

6 Diskussion

Um eine Genauigkeit von 100 % zu erreichen, könnte man darüber diskutieren, ob man evt. weitere Eingabeparameter wie Bodendichte, elektrische Leitfähigkeit und oder Lichtintensität hinzufügen könnte. Jedoch erreichen die beiden Algorithmen eine genug hohe Genauigkeit, um einer der beiden in eine real time Applikation zu implementieren.

7 Quellenverzeichnis & Abbildungsverzeichnis

[1] Atharva Ingle (2020): Crop Recommendation Dataset,
Maximize agricultural yield by recommending appropriate crops [online]
https://www.kaggle.com/atharvaingle/crop-recommendation-dataset [Stand: 31.12.2021]

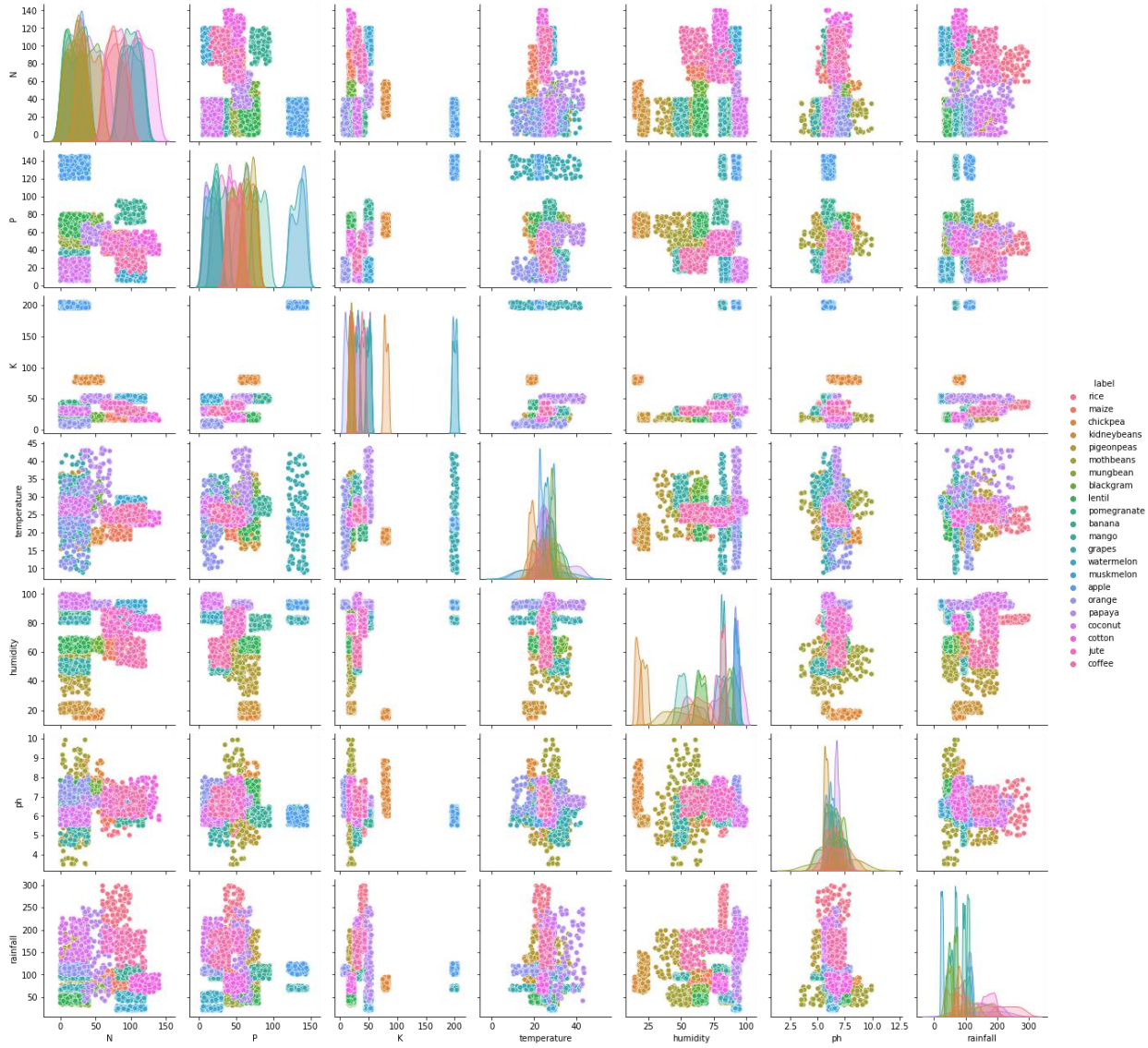
[2] Pragati Baheti (2021) A Simple Guide to Data Preprocessing in Machine Learning [online] https://www.v7labs.com/blog/data-preprocessing-guide [Stand: 31.12.2021]

[3] Sunil Ray (2017): 6 Easy Steps to Learn Naive Bayes Algorithm with codes in Python and R [online] https://www.analyticsvidhya.com/blog/2017/09/naive-bayes-explained/ [Stand: 31.12.2021]

[4] Niklas Donges (2021): A Complete Guide to the Random Forest Algorithm [online] https://builtin.com/data-science/random-forest-algorithm [Stand: 31.12.2021]

8 Anhang

8.1 Pairplot



8.2 Random Forest Decision Tree Visualisierung