Projektbericht

# Red Wine Quality
## Machine Learning Projekt

Teilnehmer:

Armin Shafiee
Fernando Moyano
Loay Ahmed Elalfy Abdelhafiz
Alexander Wemhoff

## Zusammenfassung von Fernando Moyano (Lessons Learned)

Mit diesem Projekt bekamen wir die Möglichkeit, die erlernten Methoden des maschinellen Lernens auf einen neuen Datensatz anzuwenden. Ich war froh, dass wir mit einem Datensatz gearbeitet haben, der keine anfängliche Reinigung erforderte. Als Wissenschaftler kenne ich den Aufwand für das Beschafung und Vorverarbeiten von Daten. Maschinelles Lernen hat jedoch seine einzigartigen Anforderungen, um Daten in die richtige Form zu bringen, und es war sehr aufschlussreich, die verschiedenen verfügbaren Methoden dafür zu sehen, einschließlich der Imputer, Encoder, und die verschiedene scaling Methoden.

Um dieses Abschlussprojekt durchzuführen, habe ich mich auf neuronale Netze konzentriert. Ein sehr interessanter Aspekt von NN-Algorithmen ist die Flexibilität in Form und Parametern der Netzwerke. Dies macht die Methode flexibel und sehr anspruchsvoll. Die Zahl der Hyperparameter-Kombinationen ist endlos. Ein Gefühl dafür zu bekommen, was am besten funktioniert, erfordert viel Erfahrung. In unserem Fall waren die NN-Ergebnisse nicht schlecht, aber nicht besser als andere, wie es bei Random Forests der Fall war.

Eine Schlussfolgerung aus dieser Übung war, dass es produktiver sein kann, mehr Zeit damit zu verbringen, eine Methode sehr gut anzupassen, als viele Methoden mit Standardparametern auszuprobieren. Die Wahl guter Parameter war bei NN wichtig, bei Support-Vektor-Maschinen weniger. Dies kann für diesen Datensatz spezifisch sein. Die Wahl der Methode wird wahrscheinlich an Bedeutung gewinnen, wenn man größere oder komplexere Datensätze erhält. Die Rechenzeit war bei einer NN-Grid-Search bis zu mehreren Minuten. Es war also hier kein limitierender Faktor, aber würde einer mit größeren Datensätzen oder viel mehr Parameterkombinationen werden.

Weitere Erkenntnisse waren, unter anderem:

- Die meisten Algorithmen lieferten nach der Normalisierung und Standardisierung der Daten bessere Ergebnisse. Ein PCA mit einem Verlust von 20 % der erklärten Variabilität verschlechterte die Scores etwas.
- Es ist einfach mit NN overfitting zu bekommen. Aber sehr große Freiheitsgrade in NN können immer noch zu guten Ergebnissen bei Testdaten führen (Datensatz abhängig?).
- Eine Kreuzvalidierung prüft auf genügend Stichproben jeder Target-Klasse.
- Ensemble methoden (ink. Random Forests) haben eine große auswirkung.

### Verteilung der Aufgaben

**Alle Teilnehmer**: Auswahl des Datensatzes und der Methoden, Datenbeschaffung, Data Preprocessing. Bericht Erstellung, Code Bearbeitung, Ensemble Voting.

**Armin Shafie**: Decision Trees, Random Forest

**Fernando Moyano**: Neural Networks, Support Vector Machine

**Loay Elalfy**: Logistic Regression, K-Nearest Neighbor, Cluster Analysis

**Alexander Wemhoff**: Bericht Bearbeitung

# Klärung der Aufgabenstellung

Zunächst soll anhand eines Beispiel Datensatzes untersucht werden, inwiefern eine Machine Learning Anwendung zur Einschätzung der Qualität von Weinen dienen kann.

### Definition von Qualitätskriterien

Als Qualitätskriterium wird ein Score benutzt, der die ‘accuracy’ von Voraussagen repräsentiert. Aber weil dieses Projekt eine Übung ist, gibt es keine klar definierten Qualitätskriterien.

### Lösungen die bereits existieren

Andere ML-Modelle basiert auf dem gleichen Datensatz. Wir schauen uns diese Lösungen zuerst nicht an.

### Verfahren

Für dieses Problem wenden wir überwachtes Lernen. Es ist ein Klassifikationsproblem und daher werden Klassifikationsmethoden eingesetzt.

# Daten Beschreibung

### Datenquellen

www.kaggle.com (Datensatz erstellt von [Cortez et al., 2009])

Der Datensatz besteht aus 12 Spalten und 1500 Zeilen mit Daten für Rotweine. Der Datensatz benötigt 98.58KB Speicherplatz. Downloadzeit und Speicherplatz sind keine einschränkenden Faktoren.

Der Download der Daten ist sehr einfach über folgenden Link möglich:

https://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/winequality-red.csv

Dazu wurden die Daten aus einem .csv File in ein pandas dataframe überführt.

Der Datensatz hat 11 features, die quantitativ für diverse Weine bestimmt wurden. Diese Features werden in der folgenden Tabelle beschrieben:

| Feature | Bedeutung |
|---|---|
| **fixed acidity** | Most acids involved with wine are fixed or nonvolatile (do not evaporate readily). |
| **volatile acidity** | The amount of acetic acid in wine, which at too high of levels can lead to an unpleasant, vinegar taste. |
| **citric acid** | Found in small quantities, citric acid can add 'freshness' and flavor to wines. |
| **residual sugar** | The amount of sugar remaining after fermentation stops, it's rare to find wines with less than 1 gram/liter. |
| **chlorides** | The amount of salt in the wine. |
| **free sulfur dioxide** | The free form of SO2 exists in equilibrium between molecular SO2 (as a dissolved gas) and bisulfite ion. |
| **density** | The density of water is close to that of water depending on the percent alcohol and sugar content. |
| **ph** | Describes how acidic or basic a wine is on a scale from 0 (very acidic) to 14 (very basic); most wines are between 3-4. |
| **sulphates** | A wine additive which can contribute to sulfur dioxide gas (S02) levels, which acts as an antimicrobial. |
| **alcohol sulfur dioxide** | The percent alcohol content of the wine. |

| | |
|---|---|
| **quality** | Output variable (based on sensory data, score between 0 and 10). |

Um einen Überblick über die enthaltenen features und Zeilen zu erlangen wurden die ersten 5 Zeilen des Datensatzes mit der Funktion .head() angezeigt

```
   fix_acid  volat_acid  citric_acid  ...  sulphates  alcohol  quality
0       7.4         0.70         0.00  ...       0.56      9.4        5
1       7.8         0.88         0.00  ...       0.68      9.8        5
2       7.8         0.76         0.04  ...       0.65      9.8        5
3      11.2         0.28         0.56  ...       0.58      9.8        6
4       7.4         0.70         0.00  ...       0.56      9.4        5
```

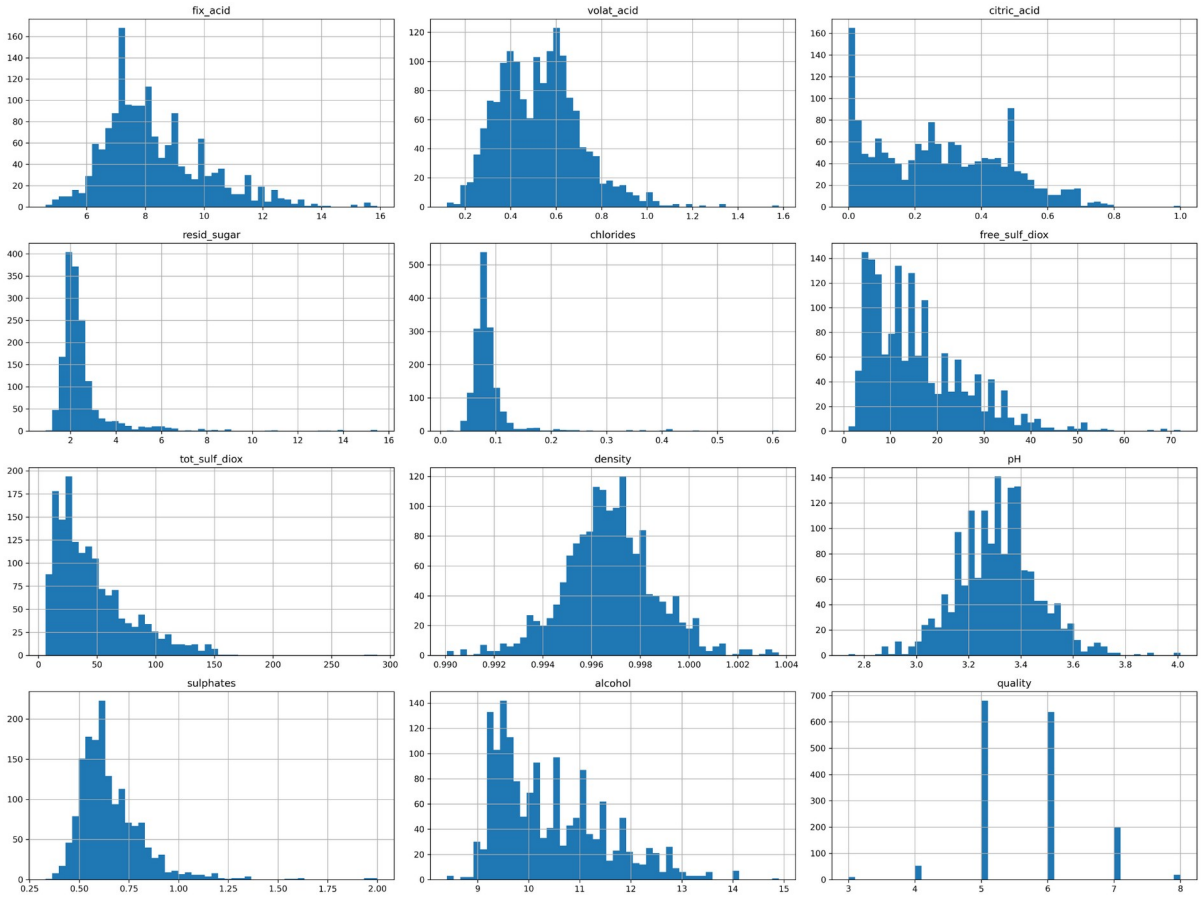
und mit .info() und .describe() zusammengefasst

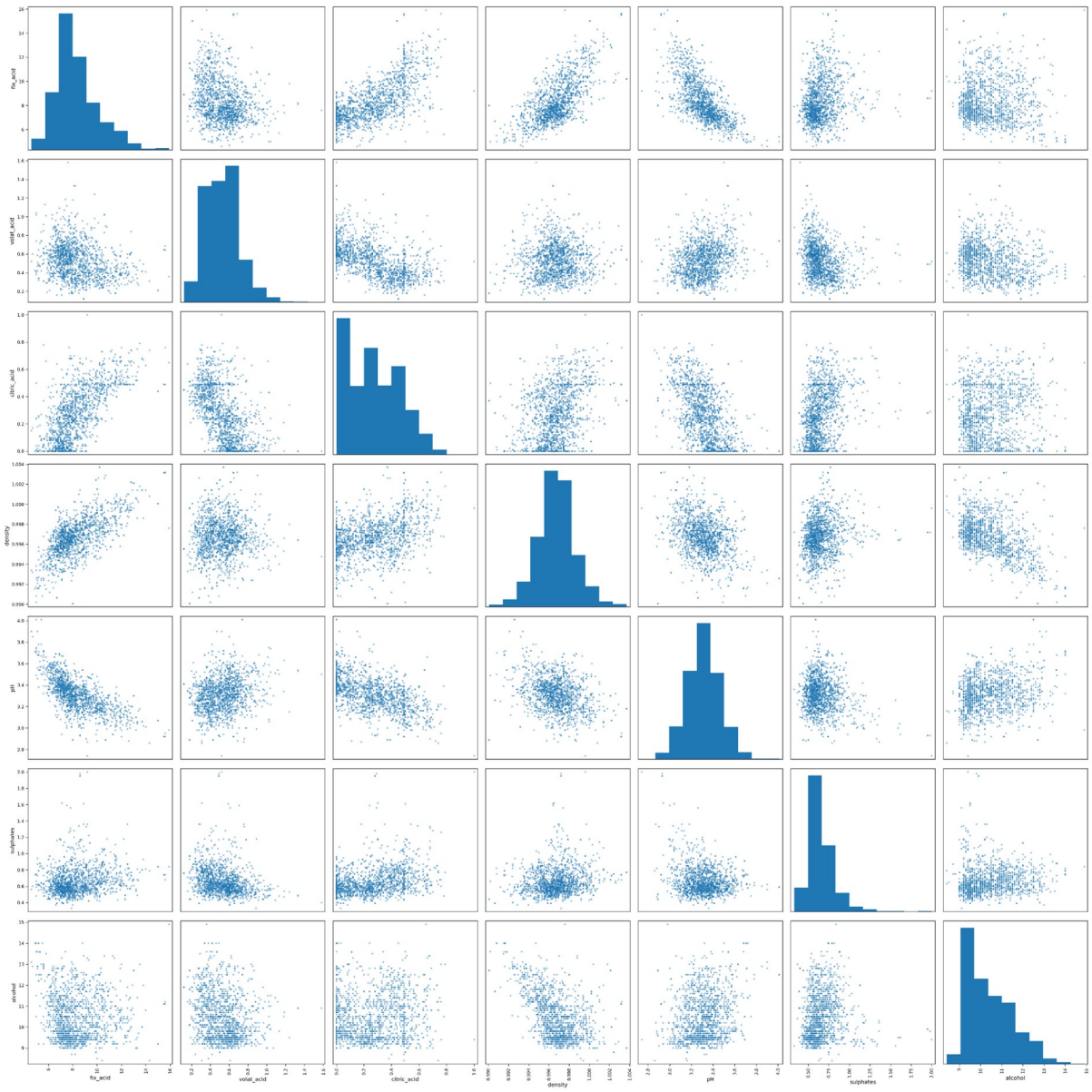
```
RangeIndex: 1599 entries, 0 to 1598
Data columns (total 12 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   fix_acid         1599 non-null   float64
 1   volat_acid       1599 non-null   float64
 2   citric_acid      1599 non-null   float64
 3   resid_sugar      1599 non-null   float64
 4   chlorides        1599 non-null   float64
 5   free_sulf_diox   1599 non-null   float64
 6   tot_sulf_diox    1599 non-null   float64
 7   density          1599 non-null   float64
 8   pH               1599 non-null   float64
 9   sulphates        1599 non-null   float64
 10  alcohol          1599 non-null   float64
 11  quality          1599 non-null   int64
```

```
          fix_acid   volat_acid  ...      alcohol      quality
count  1599.000000  1599.000000  ...  1599.000000  1599.000000
mean      8.319637     0.527821  ...    10.422983     5.636023
std       1.741096     0.179060  ...     1.065668     0.807569
min       4.600000     0.120000  ...     8.400000     3.000000
25%       7.100000     0.390000  ...     9.500000     5.000000
50%       7.900000     0.520000  ...    10.200000     6.000000
75%       9.200000     0.640000  ...    11.100000     6.000000
max      15.900000     1.580000  ...    14.900000     8.000000
```

Es wurde festgestellt, dass die Daten bereits in einer Form vorliegen, in der keine Werte über imputer ersetzt oder nachträglich bearbeitet werden müssen. Die Variablen sind alle der Type float und daher ist kein Encoding nötig.
Der Fokus lag also auf der geeigneten Skalierung der Daten. Dafür und einen besseren Überblick über die zu verarbeitenden Daten zu erlangen, wurden die Daten in Form von Plots visualisiert.
Hier werden Histogramm und Matrix-Korrelation plots gezeigt. Zur einfacheren Visualisierung haben wir im Matrixkorrelations eine Auswahl von Features gemacht, mit der diejenigen die

den stärksten Korrelationen zeigen.



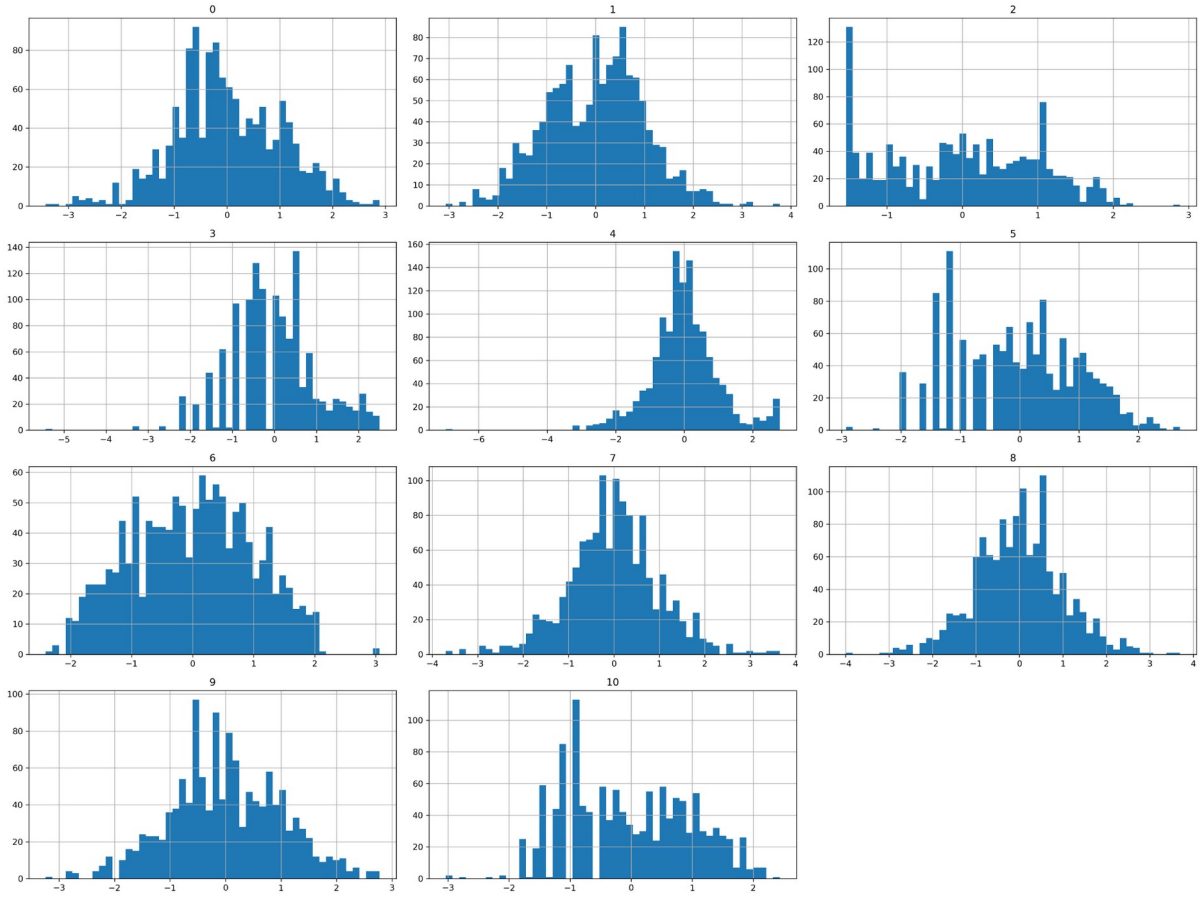



# Datentransformation

Alle Datentransformationen wurden nur für die Features gemacht. Das Target wurde nicht geändert.

## Skalierung

Da viele der Variablen eine Log-Normal Verteilung zeigten, haben wir uns für eine Normalisierungs entschieden. Dafür haben wir die PowerTransformer-Funktion von sklearn angewendet, um alle Features zu normalisieren. Dazu haben wir die Features mit der gleichen Funktion gleichzeitig standardisiert. Die neue Verteilungen nach skalierung wird unten gezeigt.




## Korrelationen und PCA

Die folgende Tabelle zeigt die Korrelationskoeffizienten zwischen allen Variablen.

| | fix_acid | volat_acid | citric_acid | resid_sugar | chlorides | free_sulf_diox | tot_sulf_diox | density | pH | sulphates | alcohol | quality |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| fix_acid | 1.00000 | -0.25613 | 0.67170 | 0.11478 | 0.09371 | -0.15379 | -0.11318 | 0.66805 | -0.68298 | 0.18301 | -0.06167 | 0.12405 |
| volat_acid | -0.25613 | 1.00000 | -0.55250 | 0.00192 | 0.06130 | -0.01050 | 0.07647 | 0.02203 | 0.23494 | -0.26099 | -0.20229 | -0.39056 |
| citric_acid | 0.67170 | -0.55250 | 1.00000 | 0.14358 | 0.20382 | -0.06098 | 0.03553 | 0.36495 | -0.54190 | 0.31277 | 0.10990 | 0.22637 |
| resid_sugar | 0.11478 | 0.00192 | 0.14358 | 1.00000 | 0.05561 | 0.18705 | 0.20303 | 0.35528 | -0.08565 | 0.00553 | 0.04208 | 0.01373 |
| chlorides | 0.09371 | 0.06130 | 0.20382 | 0.05561 | 1.00000 | 0.00556 | 0.04740 | 0.20063 | -0.26503 | 0.37126 | -0.22114 | -0.12891 |
| free_sulf_diox | -0.15379 | -0.01050 | -0.06098 | 0.18705 | 0.00556 | 1.00000 | 0.66767 | -0.02195 | 0.07038 | 0.05166 | -0.06941 | -0.05066 |
| tot_sulf_diox | -0.11318 | 0.07647 | 0.03553 | 0.20303 | 0.04740 | 0.66767 | 1.00000 | 0.07127 | -0.06649 | 0.04295 | -0.20565 | -0.18510 |
| density | 0.66805 | 0.02203 | 0.36495 | 0.35528 | 0.20063 | -0.02195 | 0.07127 | 1.00000 | -0.34170 | 0.14851 | -0.49618 | -0.17492 |
| pH | -0.68298 | 0.23494 | -0.54190 | -0.08565 | -0.26503 | 0.07038 | -0.06649 | -0.34170 | 1.00000 | -0.19665 | 0.20563 | -0.05773 |
| sulphates | 0.18301 | -0.26099 | 0.31277 | 0.00553 | 0.37126 | 0.05166 | 0.04295 | 0.14851 | -0.19665 | 1.00000 | 0.09359 | 0.25140 |
| alcohol | -0.06167 | -0.20229 | 0.10990 | 0.04208 | -0.22114 | -0.06941 | -0.20565 | -0.49618 | 0.20563 | 0.09359 | 1.00000 | 0.47617 |
| quality | 0.12405 | -0.39056 | 0.22637 | 0.01373 | -0.12891 | -0.05066 | -0.18510 | -0.17492 | -0.05773 | 0.25140 | 0.47617 | 1.00000 |

Die höchste Korrelation (0,67) bestand zwischen 'fixed acidity' und 'citric acid', gefolgt von 'density' mit 'fixed acidity (0.66)'. Da es keine sehr hohen Korrelationen gab, kann eine Dimensionsreduktion mittels PCA zu einer signifikanten Informationsverlust führen. Wir haben jedoch PCA mit genügend Komponenten angewendet, um ca. 80% der erklärten Variabilität zu behalten (6 components). Wir haben dann die Modelle mit und ohne PCA-Dimensionsreduzierung getestet.

Um die Transformation der Daten auszuführen wurde eine einheitliche Programmstruktur erzeugt, bei der die Daten über eine Pipeline einzelne Prozesse der Verarbeitung durchlaufen können.

Der Datensatz wurde in einen Trainings- und einen Testdatensatz aufgeteilt mit Hilfe des train_test_split Methode. Die Größe des Testdatensatz war 20%. Da einige Zielwerte in

geringer Zahl vorlagen, haben wir einen ‘stratified split’ verwendet, der auf die Zielvariable ‘quality’ angewendet wurde.
Der Trainingssatz und der Testsatz wurden in drei verschiedene Arrays reformiert: X_trn, X_trn_s und X_trn_sp, die untransformiert, skaliert und skaliert + PCA darstellen.

# Daten Analyse

Wir haben folgende Modelle verwendet:

- Logistic Regression
- K-Nearest Neighbor
- Support Vector Classifier
- Multilayer Perceptron
- Decision Tree Classifier
- Random Forest Classifier

Wir hatten im Voraus keine triftigen Gründe, einen bestimmten Satz von Klassifikatoren zu wählen. Wir haben daher eine breite Auswahl getroffen, um einen Vergleich der Ergebnisse zu ermöglichen. Die individuelle Anwendung wird für jeden unten beschrieben.
In allen Fällen haben wir die entsprechenden sklearn-Klassen verwendet. Als Metrik zur Berechnung des Scores wurde *accuracy* verwendet.

## Beschreibung der individuellen Modellen

### Logistic Regression

| Train | Score | Mean of cross validation score with cv=7 |
|---|---|---|
| X_trn | 0.595 | 0.589 |
| X_trn_s | 0.613 | 0.61 |
| X_trn_sp | 0.597 | 0.59 |

### K-Nearest-Neighbor

Der Algorithmus wurde auf die vier verschiedenen Transformationen der Trainingsdaten mit der Anzahl der Nachbarn als Variable zwischen 1 und 8 angewendet. Die scores sind:

| Train | neighbors | Score | Mean of crossvalidation score with cv=7 |
|---|---|---|---|
| X_trn | 1 | 1.00 | 0.54 |
| X_trn_s | 1 | 1.00 | 0.61 |
| X_trn_p | 1 | 1.00 | 0.54 |
| X_trn_sp | 1 | 1.00 | 0.61 |
| X_trn | 2 | 0.79 | 0.49 |
| X_trn_s | 2 | 0.83 | 0.55 |
| X_trn_p | 2 | 0.79 | 0.48 |
| X_trn_sp | 2 | 0.82 | 0.53 |
| X_trn | 3 | 0.74 | 0.47 |

| | | | |
|---|---|---|---|
| X_trn_s | 3 | 0.77 | 0.56 |
| X_trn_p | 3 | 0.74 | 0.47 |
| X_trn_sp | 3 | 0.77 | 0.54 |
| X_trn | 4 | 0.68 | 0.49 |
| X_trn_s | 4 | 0.72 | 0.58 |
| X_trn_p | 4 | 0.68 | 0.49 |
| X_trn_sp | 4 | 0.70 | 0.57 |
| X_trn | 5 | 0.64 | 0.48 |
| X_trn_s | 5 | 0.71 | 0.59 |
| X_trn_p | 5 | 0.64 | 0.48 |
| X_trn_sp | 5 | 0.70 | 0.57 |
| X_trn | 6 | 0.63 | 0.51 |
| X_trn_s | 6 | 0.68 | 0.58 |
| X_trn_p | 6 | 0.63 | 0.50 |
| X_trn_sp | 6 | 0.66 | 0.58 |
| X_trn | 7 | 0.61 | 0.50 |
| X_trn_s | 7 | 0.69 | 0.58 |
| X_trn_p | 7 | 0.61 | 0.50 |
| X_trn_sp | 7 | 0.66 | 0.57 |
| X_trn | 8 | 0.61 | 0.50 |
| X_trn_s | 8 | 0.66 | 0.58 |
| X_trn_p | 8 | 0.61 | 0.50 |
| X_trn_sp | 8 | 0.66 | 0.58 |

The best score (0.72) was for X_trn_s with 4 neighbors. The same value came from the grid search algorithm.

## Multilayer Perceptron (Neuronale Netze)

*https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html*
*https://scikit-learn.org/stable/modules/cross_validation.html#cross-validation*
*https://scikit-learn.org/stable/modules/model_evaluation.html#scoring-parameter*
*https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html*

Die Freiheitsgrade wurden über die folgende Formel berechnet:

```
DF = (F + 1) * nl1 + nl1 * nl2 + nl2 * nl3 + nl3 * T
```

F steht hierbei für die Anzahl der features (11 - standard, 6 - nach durchführen einer PCA),
T steht für die Anzahl an output Parametern (1),
nl1, nl2, nl3 = Neuronen in der jeweiligen Schicht
Um die Freiheitsgrade unterhalb der Anzahl an samples zu halten wurde mit einer maximalen Anzahl von 60 Neuronen gearbeitet.
Der ursprüngliche Test lief mit 3 hidden Layers ab, die jeweils 10 Neuronen beinhalteten.
Die Optimierung der ersten Durchläufe ist nicht konvergiert. Daher wurde die Anzahl der Iterationen von 200 auf 500 erhöht. Weiterhin wurde der Parameter (learning_rate_init) mit 0.01 festgelegt. (learning_rate) wurde auf 'adaptive' gesetzt. Der Einfluss des Parameters

random_state hat großen Einfluss auf das Ergebnis, die folgenden Ergebnisscores konnten mit Hilfe des Parameters random_state = 1 erzielt werden:

**Initial test scores (Multilayer Perceptron)**

| Train | Score |
|---|---|
| Initial test score X_trn: | 0.690 |
| Initial test score X_trn_s: | 0.724 |
| Initial test score X_trn_p: | 0.675 |
| Initial test score X_trn_sp: | 0.651 |
| Initial test cross validation score mean for X_trn: | 0.566 |
| Initial test cross validation score mean for X_trn_s: | 0.576 |
| Initial test cross validation score mean for X_trn_p: | 0.574 |
| Initial test cross validation score mean for X_trn_sp: | 0.553 |

Skalierung (Normierung und Standardisierung) hatte keinen großen Einfluss, auch wenn Skalierung als Methode für neuronale Netze empfohlen wird. Anschließend wurde die Analyse mit den skalierten Daten fortgesetzt (X_trn_s).

*Scaling (normalization and standardization) and PCA did not have a large impact. *However, because scaling is recommended for neural networks, we continued the *analysis with the scaled data (X_trn_s).

**Grid search zur Bestimmung der besten Parameterkombination:**

Es wurde ein grid search durchgeführt um die bestmöglichen Werte zu erhalten. Interessanterweise wurden trotz random_state_value verschiedene Ergebnisse ermittelt.

- Score nachdem die besten Parameter mit grid search bestimmt wurden: 0.996
- Gemittelter cross validation score mit best params von grid search: 0.605

**Scoring mittels Testdatensatz**
Abschließend wird die wirksamkeit der Algorithmen mit Hilfe der Testdatenmenge getestet. Die ursprüngliche MLP und der MLP nach grid search ergaben sich zu:

| Train | Score |
|---|---|
| mlp_init | 0.54 |
| mlp_gs | 0.628 |

Grid search hat sich als äußerst effektiv herausgestellt, es half das Modell zu verbessern. Der finale Score ist daher vergleichbar mit den Scores, die andere bereits ermittelten.

## Support Vector Machine

Support vector machines sollten nur dann benutzt werden, wenn Datensätze keine große Anzahl an features enthalten. Wir haben beschlossen, dass 11 features eine vertretbar kleine Menge an features darstellt. Es wurde der Support Vector classifier (SVC) Algorithmus angewendet. Unter den Standardeinstellungen ergaben sich die Scores zu:

| Train | Score |
|---|---|
| Initial test score für X_trn | 0.515 |
| Initial test score für X_trn_s | 0.689 |
| Initial test score für X_trn_sp | 0.644 |
| Cross validation score mean für X_trn | 0.507 |
| Cross validation score mean für X_trn_s | 0.613 |
| Cross validation score mean für X_trn_sp | 0.60 |

Grid search stellte sich hier als nicht zielführend heraus, deshalb wurde die default Option als ‘best model’ beibehalten.

## DecisionTreeClassifier & RandomForestClassifier

https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html
https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

### Vorgehensweise

- Stratifizierte Verteilung der Daten in *X_trn*, *X_tst*, *y_trn* und *y_tst* durch *train_test_split*
- Änderung der Verteilungsform der X-Daten in die Normalverteilung (Standardisierung der X-Daten) mittels *PowerTransformer*
  (*X_trn* ---> *X_trn_s*, *X_tst* ---> *X_tst_s*)
- Verwendung der *PCA*-Methode für die X-Daten
  (*X_trn* ---> *X_trn_p*, *X_tst* ---> *X_tst_p*)
- Verwendung von *PowerTransformer* und *PCA* zusammen via *Pipeline*
- Auswertung der Algorithmen *DecisionTreeClassifier* und *RandomForestClassifier* jeweils für die folgenden vier Fälle, wobei die besten Parameter für diese Algorithmen mithilfe von *GridSearchCV* festgelegt wurden

    1. Ohne PCA und Standardisierung (*X_trn*, *X_tst, y_trn, y_tst*)
    2. Ohne PCA, mit Standardisierung (*X_trn_s*, *X_tst_s, y_trn, y_tst*)
    3. Mit PCA, ohne Standardisierung (*X_trn_p*, *X_tst_p, y_trn, y_tst*)
    4. Mit PCA und Standardisierung (*X_trn_sp*, *X_tst_sp, y_trn, y_tst*)

### Ergebnisse von *DecisionTreeClassifier*:

1. The best tree parameters for X_trn: DecisionTreeClassifier(max_features=4, random_state=42)
2. The best tree parameters for X_trn_s: DecisionTreeClassifier(max_features=4, random_state=42)
3. The best tree parameters for X_trn_p: DecisionTreeClassifier(max_features=5, random_state=42)
4. The best tree parameters for X_trn_sp: DecisionTreeClassifier(max_features=4, random_state=42)

1. Mean tree training score for X_trn: 0.7639341932811892
   Tree test score for X_trn: **0.628125** ---> max. test score

2. Mean tree training score for X_trn_s: 0.766275579922082
   Tree test score for X_trn_s: 0.625

3. Mean tree training score for X_trn_p: 0.8167673282614714
   Tree test score for X_trn_p: 0.60625

4. Mean tree training score for X_trn_sp: 0.8150869091016784
   Tree test score for X_trn_sp: 0.59375

### Ergebnisse von *RandomForestClassifier*:

1. The best forest parameters for X_trn: RandomForestClassifier(max_features=3, random_state=42)
2. The best forest parameters for X_trn_s: RandomForestClassifier(max_features=5, n_estimators=150, random_state=42)
3. The best forest parameters for X_trn_p: RandomForestClassifier(max_features=3, random_state=42)
4. The best forest parameters for X_trn_sp: RandomForestClassifier(max_features=2, n_estimators=150, random_state=42)

1. Mean forest training score for X_trn: 0.6455313301246347
   Forest test score for X_trn: 0.678125

2. Mean forest training score for X_trn_s: 0.6428251622995245
   Forest test score for X_trn_s: **0.690625** ---> max. test score

3. Mean forest training score for X_trn_p: 0.6803065852838366
   Forest test score for X_trn_p: 0.646875

4. Mean forest training score for X_trn_sp: 0.66733266886931
   Forest test score for X_trn_sp: 0.6625

## Ensemble Voting

Wir haben den VotingClassifier() verwendet, um eine Ensemblevorhersage zu bekommen. Wir haben sowohl Hard- als auch Soft-Voting mit allen Estimators getestet.

# Test und Validierung

Testdaten wurden nur zur Validierung verwendet, d. h. es wurde kein Fit an diesen Daten durchgeführt. Die Scores der Modelle mit der besten Leistung für jeden einzelnen Estimator sind:

| Estimator | Accuracy |
|---|---|
| MLP: | 0.628 |
| KNN: | 0.606 |
| LogReg: | 0.6 |
| Decision Tree: | 0.625 |
| Random Forest: | 0.691 |
| SVC: | 0.594 |
| Hard voting classifier: | 0.672 |
| Soft voting classifier: | 0.684 |

# Schlussfolgerungen

Es ist keine leichte Aufgabe, die Weinqualität vorherzusagen! Dies mag nicht überraschen, da das Qualitätsmaß teilweise subjektiv ist. Wir konnten den Testdatensatz jedoch mit einer Genauigkeit von ca. 70 % voraussagen, was nicht sehr hoch ist, aber für diesen Fall vielleicht so viel wie zu erwarten ist.
Mit 1500 Zeilen was der Datensatz mittelgroß. So musste aufgrund der Datenmenge nicht auf bestimmte Modelltypen verzichtet werden. Wir haben erwartet, dass einige nicht ideal sein würden. Z.B. Neuronale Netze funktionieren am besten bei großen Datenmengen, während SVM bei großen Datensätzen Schwierigkeiten haben können. Die meisten Algorithmen lieferten jedoch eine ähnliche Leistung mit einer Accuracy um 0,60.
Deutlich bessere Accuracies wurden in zwei Fällen erzielt: durch die Verwendung von Voting-Klassifikatoren und mit Random Forests. Diese Methoden erhöhten die Leistung auf 0,68 bzw. 0,69 und stellten damit eine deutliche Verbesserung dar. Da Random Forests eine Art Ensemble-Methode sind, ist dies nicht verwunderlich.
Wir glauben nicht, dass mit diesem Datensatz viel höhere Vorhersageleistungen erzielt werden können. Einige Projekte melden Genauigkeitswerte von bis zu 80 % dieser Daten. Dies könnte also Raum für Verbesserungen sein. Rastersuchen oder Zufallssuchen könnten mit Random Forests an neuen Parametersätzen getestet werden. Komplexere Netzwerke könnten auch mit MLPs getestet werden.
Genauigkeiten bis ca. 90 % wurden für diese Daten gezeigt:
https://www.kaggle.com/vishalyo990/prediction-of-quality-of-wine
Aber in solchen Fällen wurden die Zielwerte jedoch auf nur 2 Fälle (guter Wein / schlechter Wein) geändert.
Ein Teil des Erfolgs bei der Erzielung hoher scores kann jedoch mit dem Zufall zusammenhängen, d. h. davon, wie die Test- und Train-datasets zufällig ausgewählt werden.