# Ziel des Projektes

Ziel des Projektes ist es, einen zukünftigen Rotwein qualitativ anhand seiner Inhaltsstoffe/-werte einzuordnen. Bisher wurden die Qualitätsbeurteilungen durch Weintester ausgeführt. Mit einem Machine Learning Algorithmus soll das in Zukunft durchgeführt werden.

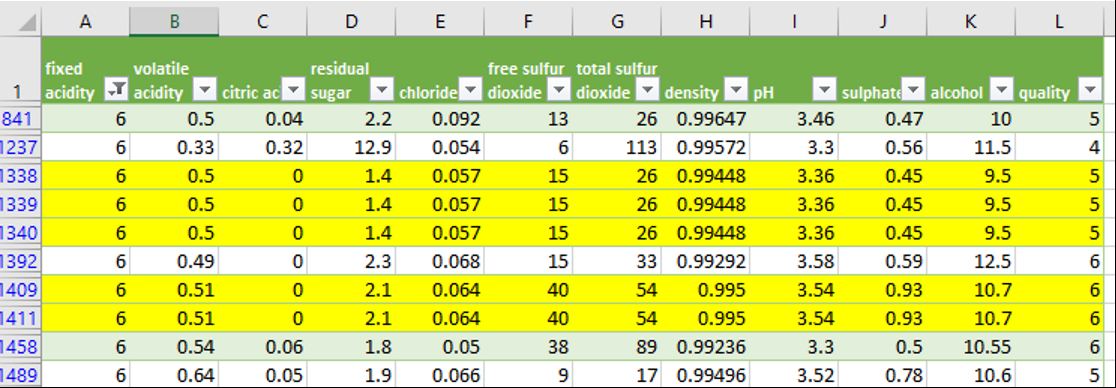
# Beschreibung der Daten

In der Tabelle sind bereits Weinkriterien und Qualitätsbewertungen für Weine aufgeführt. Inwieweit diese Kriterien die Qualität bestimmen, sind teilweise klar bzw. unklar. Eine Analyse der Daten kann helfen, hier mehr Klarheit zu schaffen. Folgende Kriterien sind im Datensatz aufgeführt:

|  |  |
| --- | --- |
| **Kriterium / Feature** | **Inwieweit die Qualität eines Weines dadurch bestimmt ist** |
| 1 - Fixed acidity = Fester Säuregehalt (bleibt im Wein) | Unklar, wann gut oder schlecht |
| 2 - Voltile Acidity = flüchtige Säure (Duft und Geruch) | Unklar, wann gut oder schlecht |
| 3 - Citric acid = Zitronensäure | Unklar, wann gut oder schlecht |
| 4 - Residual Sugar = Restzuckergehalt | Unklar, wann gut oder schlecht |
| 5 - Chlorides  = Chloride | Annahme: je weniger, um so besser |
| 6 - Free sufur dioxide = Schwefeldioxid | Annahme: je weniger, um so besser |
| 7 - Total sulfur dioxide = Gesamtschwefel | Annahme: je weniger, um so besser |
| 8 - Density = Dichte (Anteil an Inhaltsstoffen) | Annahme: je mehr, um so besser |
| 9 - pH = Wert (korreliert mit Säure) | Mittelwert bei Säure ist besser (Mttelwert ~ 3 bis 4) |
| 10 - Sulphates = Sulfate (Konservierung) | Annahme: je weniger, um so besser |
| 11 - Alcohol = Alkoholgehalt | Annahme: 12-15% ist gut für trockenen Wein, bei süßen auch unter 10%, gesetzlicher Mindestalkohol liegt bei 7-8% |
| 12 - Quality = Qualität (Zielspalte) | **Zielwert. Qualitätswert** wurde durch Weintester bestimmt. |
| 13 -Farbe | Rotwein und Weißwein. Daten auf Rotweine beschränkt. |

# Rotwein Datensatz

Der Datensatz besteht aus 1599 Rotweinen (Zeilen), wobei 240 Duplikate (15%) im Datensatz vorliegen. Duplikate können entstehen, weil es sein kann, dass diese Weine von unterschiedlichen Winzern hergestellt, aus ähnlichen Regionen ider auch durch verscheidene Jahrgänge bestimmt sein könnten. Somit kann es trotz der im Datensatz vorliegenden identischen Werte, andere nicht erfasste unterschiedliche Eigenschaften gegeben haben, die später entfernt wurden (Überlegung hier ist, dass DSGVO Vorgaben und Datenbereinigung am Datensatz stattgefunden haben könnten).   
Beispiele für nicht erfasste Eigenschaften wären der Standort, Aromastoffe, Farbstoffe oder auch Gerbstoffe. Die nachfolge Tabelle zeigt Beispiele für Duplikate, aus dem Datensatz:



# Trainingsdaten und Testdaten

Die Daten bestehen aus 1599 Datenzeilen, wobei jede Zeile einem Wein mit 11 Kriterien (Features) und einer Qualitätsbeurteilung (Target) zugeordnet ist.

Bei der Sichtung der Daten wurden keine fehlenden (NaN) Werte gefunden.

**Der Aufbau der Zeilenwerte sieht wie folget aus:**

|  |  |
| --- | --- |
| **Kriterium / Feature** | **Datentyp** |
| 1 - Fixed acidity | float |
| 2 - Voltile Acidity | float |
| 3 - Citric acid | float |
| 4 - Residual Sugar | float |
| 5 - Chlorides | float |
| 6 - Free sufur dioxide | float |
| 7 - Total sulfur dioxide | float |
| 8 - Density | float |
| 9 - pH | float |
| 10 - Sulphates | float |
| 11 - Alcohol | float |
| 12 - Quality | integer |

Die Qualitätsspalte sind Klassen, welche in Reihenfolge ihrer Qualität von 3-8 durchnumeriert sind. Somit hat die Bezeichnung der Klassen durch ihre aufgesteigenden Wert eine Bedeutung und man könnte sich auch eine Regression vorstellen.

Allerdings sind es keine kontinuierlichen Werte, sondern subjektiv gewählte Werte. Es gibt somit keine Zwischenwerte, und es wäre schwer die Zwischenwerte zu interpetieren. Man müsste einen ordinalEncoder auf die Werte anwenden, um sie zu interpretieren.

Eine Klassifikation ist einfacher, da es in dem Fall keine Zwischenwerte auftreten und es direkt interpretiert werden kann.

Auffällig an den Daten ist, dass überwiegend Inhaltsstoffe aufgeführt sind, die man nicht im Wein haben möchte, wie Sulphat, Schwefeldioxid und Chlorid. Von den „positiven“ Inhaltstoffen sind eigentlich nur Alkohol, Zitronensäure und je nach Geschmack Zucker vorhanden.

Informationen, die man eigentlich haben wollen würde, um die Qualität der Weine zu beurteilen (wie Jahrgang, Standort, Sonnenstunden, oder auch fast alle Geschmackstoffe), fehlen. Dies weist darauf hin, dass es schwierig wird mit den Daten etwas über die Qualität des Weine auszusagen. Auch ist die Einteilung von Wein nach Qualität sehr subjektiv. So gibt es Menschen, welche süßen Wein bevorzugen, während andere Menschen trockenen Wein mögen.

Die Daten wurden vorab in einen Trainings- und Testdatensatz aufgeteilt. Da der Datensatz nicht sehr groß ist, wurde der Datensatz nach dem Pareto-Prinzip (80/20) aufgeteilt.

Der Zufalsswert (Random Seed) wurde auf 42 festgesetzt.

## Verteilung der Qualitätkriterien

Die Weine sind in drei wesentlichen Qualitätsstufen (“Good”, “Acceptable”, “Poor”) eingeteilt, die sich folgendermaßen aufteilen. Diese Darstellung zeigt eine zweite Schwierigkeit des Datensatze die Anzahl der Weine in den unterschiedlichen Klassen ist völlig ungleich. Damit kommen viele Algorithmen wie Bayes nicht zurecht.

Eigentlich bräuchte man ausgeglichene Verhältnisse, um gute Vorhersagen zu machen.

Ein Bild, das Text, Screenshot, Schrift, Diagramm enthält.

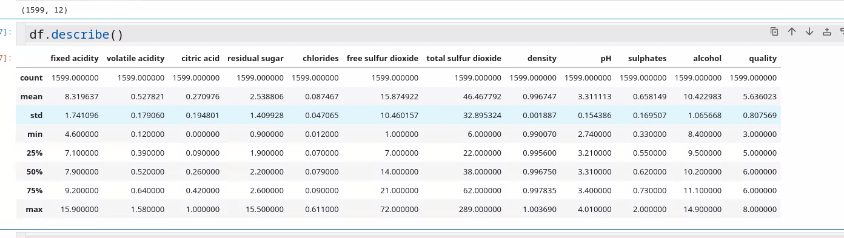
Automatisch generierte Beschreibung

# Feature Korrelationen

Zunächst wurden alle Features auf Datenintegrität (Vorhandensein, Ausreißer, Werteumfang) gepüft, um in einem zweiten Schritt die einzelnen Features zueinander auf ihre Korrelation zu prüfen.

Alle Spalten haben vergleichbare statistische Eigenschaften außer:

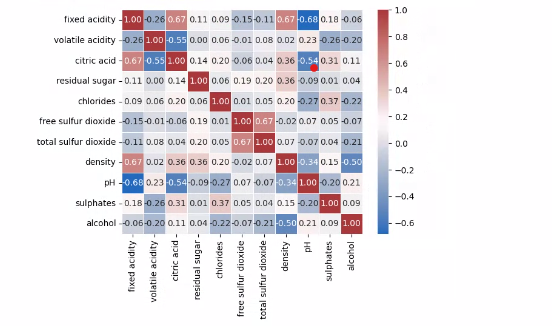
* “**total sulfur dioxide”**
  + Mittelwerte = 46.46
  + Abweichung = 32.9
  + Median = 38



## Korrelation der Features

Die Korrelation der Werte zueinander. Je höher die absolute Zahl, desto größer ist die Korrelation.  
“Starke Korrelation”, definiert ab 0.5 (Absolutwert):  
(1) fixed acidity & pH; (2) volatile acidity & citric acid; (3) residual sugar & density; (4) chlorides & sulphates; (5) free sulfur dioxide & total sulfur dioxide; (7) sulphate & chlorides ; (8) alcohol & density;

Es finden sich 8 starke Korrelationen, dabei werden diese, definiert ab 0.5 (Absolutwert):  
(1) fixed acidity & pH; (2) volatile acidity & citric acid; (3) residual sugar & density; (4) chlorides & sulphates; (5) free sulfur dioxide & total sulfur dioxide; (7) sulphate & chlorides ; (8) alcohol & density.



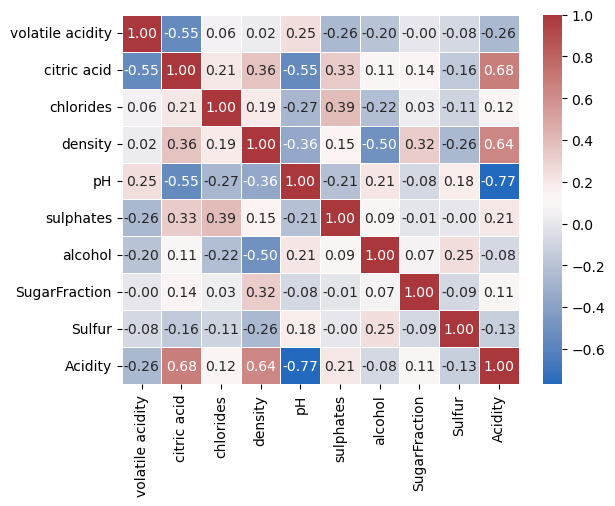
In der Hoffnung die Analyse zu vereinfachen, fassen wir Daten, welche stark korrellieren zusammen. Dabei nehmen wir dabei die Spalten, welche 0.67 in der Korrelationsmatrix aufweisen. Dazu gehören “fixed acididity pH”, “volatile acidity citric acid, sulfur dixodie und total sulfor dioxidie. Bei diesen ist es auch aus chemischer Sicht völlig einsehbar, dass diese korrellieren. So ist der pH-Wert eine Messung des Säuregehalts, beziehungswiese der Basizität, welche hier nicht relevant ist. Das der Säuregehalt mit der Zitronensäure zusammenängt ist ebenso nachvollziehbar, ebenso die beiden Schwefeldioxid-Spalten.

Zusätzlich gibt es starke Korrelationen bei der density and residual sugar. Das ist chemisch gesehen relativ nachvollziehbar, da der Zucker die Dichte beeinflusst, allerdings wird die Dichte nicht nur von dem Zucker bestimmt, so dass sich hier die Befürchtung ergibt, dass ein zusammenfassen der Spalten dazu führt, dass Informationen verloren gehen.

Aus diesem Grund wird der Zuckergehalt durch die Dichte geteilt, um den Zuckeranteil an der Dichte herauszufinden. Dies könnte neue Information geben. Die Zuckersplate wird fallengeslassen, weil sich sonst eine Korrelation von 1 mit der neuen Spalte ergeben würde. Die Dichtespalte dagegen wird weitergenutzt, weil sonst Informationen verloren gehen würden.

Die Zusammenfassung der Sulfordioxide -Spalten ist durch eine Division einfach möglich, da die Menge an freiem Sulfordioxid eine Teilmenge des totalen Sulfordioxids ist. Somit riskieren wir in dem Fall nicht durch 0 zuteilen. Dies ist jedoch bei der Zitronensäure und fixed acididy der Fall, also können diese Spalten nicht durch eine Division zusammengefasst werden .

Die fixed acidity wird mit dem pH zusammemgefasst, da die gebundene Säure den pH-Wert beeinflusst. Fixed acidity wird fallengelassen da es keine weitere Infomationen gibt. Dagegegen wird der pH-Wert zusätzlich auch durch die Zitronensäure bestimmt. Somit bleibt diese Spalte zunächst bestehen, auch wenn sich jetzt eine sehr hohe Korrelation mit der neuen Spalte ergibt. Die Korrelation zwischen der fixed-acididy und der Zitronensäure ist dafür nicht mehr vorhanden.



Eine starke Korrelation ist allerding noch vorhanden, und zwar zwischen Zitronensäure und Acidity. Diese ist zu erwarten, weil die Zitronensäure beeinflusst der Wein ist. Jedoch ist Zitronensäure einer der wenigen angegeben gewünschten Geschmäcker, somit ist zu erwarten, dass es einen starken Einfluss auf die Weinqualität hat. Somit wurde diese Spalte nicht gelöscht, obwohl die starke Korrelation vorhanden ist.

**Löschung der Duplikate**

Anschließend wurde über die Löschung der Duplikate diskutiert. Dabei ergab sich die Frage, ob diese vor oder nach der Teilung in der Train- und Trainingsdatensatzes. Es gab zwei Ansätze zum einen die Duplikate am Anfang zu löschen, oder die Duplikate in der Testmenge zu löschen, während sie in der Trainingsmenge verbleiben. Das Argument für das Behalten der Duplikate in der Trainingsmenge besteht, darin dass sie in der Trainingsmenge nicht stören. Dagegen würden in der Testmenge Duplikate in Falle Fehler bei den Duplikaten zu Verstärkung der Fehler führen.

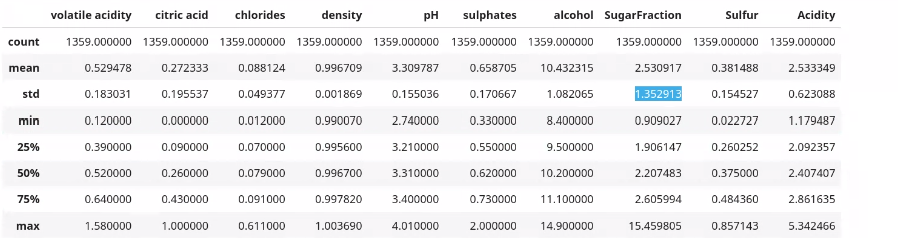
Das Behalten der Duplikate würde dazu führen, dass der Algorithmus in der Testmenge auf Daten trifft, die er schon gelernt hat. Es wäre ein Informations-Leakage und würde zusätzlich zu Overfitting führen. Aus diesem Grund wurden die Duplikate gelöscht.

**Ausreißer**

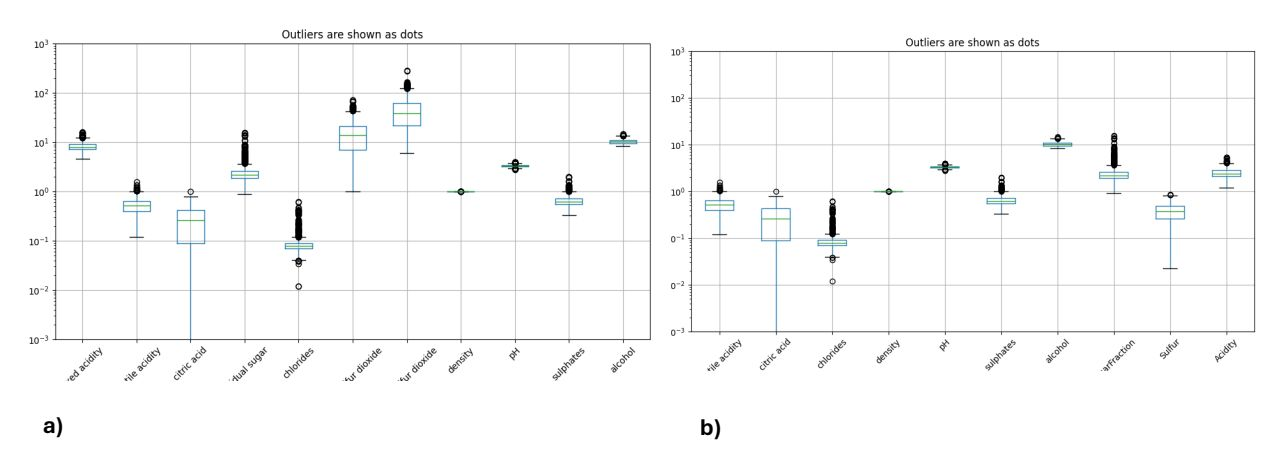
Es wurde auch überlegt, ob es nötig ist Ausreißer zu entfernen, dafür wurde auf die statische Zusammenfassung geguckt. Es wurde argumentiert, dass sich Ausreißer durch eine große Standardabweichung bemerkbar machen würden. Die statische Zusammenfassung zeigte nur im Fall der totalen Schwefeldioxidkonzentration eine hohe Standardabweichung.



Dies war eine der Spalten, welche verändert worden war, in dem es mit der flüchtigen Schwefeldioxidkonzentration zusammenfasst wurde, aus dem Grund wurde die statistische Zusammenfassung mit den neuen Daten wiederholt. In dem Fall war diese starke Standardabweichung verschwunden. Alle Standardabweichungen waren unter 1.5. Dies deutet darauf hin, dass es keine Probleme mit Ausreißern geben sollte.



Zur genaueren Untersuchung bezüglich Ausreißer wurde ein Boxplot erstellt. Die Ausreißer sind dabei einzelne Punkte. Der Boxplot wies darauf hin, dass es immer noch einige Ausreißer vorhanden sind, insbesondere bei Chlorid.

Das lässt darauf schließen, dass der Standardskaler nicht ideal ist. Somit wurde er zwar im zunächst verwendet, aber gleichzeitig bedacht, dass zur Verbesserung der RobustScaler hilfreich wäre. Auch gibt es bei verschiedenen Algorithmen Varianten, welche mit Ausreißern besser klarkommen und entsprechend genutzt werden sollten. Diese können auch im zweiten Schritt berücksichtigt werden. Auch sollte Support Vector Machines (SVM) damit besser klarkommen als andere Algorithmen. Die Ausreißer sind existieren sowohl in den Originaldaten (a) als auch bei den geänderten Spalten b). 

**Erstellte Dateien zur Analyse**

Nach der Entfernung der Duplikate wurde die Daten gespeichert. Diese Datei sollte für den ersten nichtoptimierten Schritt verwendet werden.

In einer zweiten Datei wurden Daten mit den zusammen gefassten Spalten gespeichert. Dabei wurde überlegt die Spalten nur in der Trainingsdaten geändert werden müssen, oder auch in Trainingsdaten. Wenn es in beiden verändert wird, würde dies bedeuten, dass auch in neuen Daten die Spalten verändert werden müssen. Um dies zu vereinfachen wurde überlegt, ob man eine Klasse schreiben könnte, welches das automatisch erledigt. Um diese Klasse zu schreiben, wurde überlegt, ob der Imputer als mögliches Beispiel dienen kann. Wobei die Klasse so angepasst werden würde, dass nicht Werte ersetzt, sondern anstelle dessen dividiert wird. Dies wurde als Verbesserung in den zweiten Schritt verschoben. Wobei auch überlegt wurde, ob eine Funktion reichen würde, da eigentlich kein fit durchgeführt werden muss, sondern nur ein transform. Für das Schreiben einer Klasse blieb allerdings nicht genug Zeit

**Algorithmen und Beobachtungen bezüglich der Datenvorbereitung**

Nachdem Datenvorbereitung abgeschlossen war, wurden de Algorithmen wurden aufgeteilt: SVM= Sayedeh, Christian = DecisionTree, Peter = logisticRegression, Cordula = Bayes

Es fiel auf, dass die Duplikate falsch gelöscht wurden, es wurden alle Originale mit deren Duplikate gelöscht, so dass zu viele Daten verloren gingen. Dies wurde so geändert, dass zumindest eine Variante enthalten bleibt.

Es zeigte sich auch, dass die Skalierung keinen Einfluss auf die Feature Engineering hat. Somit kann die Skalierung auch später durchgeführt werden. Die Skalierung mit dem StandardScaler sorgte auch dafür, dass Bayes nicht durchgeführt werden konnte, da die Skalierung negative Werte eingefügt hatte. Dadurch, dass die Skalierung später durchgeführt werden kann, kann man bei den verschiedenen Algorithmen unterschiedliche Skalierungen durchführen.

Beim Durchführen des Cross-Val-Scores beim Decision Tree und logistic regression, tauchten beim Cross-Val-Score die Warnung auf, dass in der kleinsten Klasse (Qualitätstufe) nur zwei Daten wären. Dies deutet darauf hin, dass nicht genug Daten der besten, oder auch der schlechtesten Qualitätsstufen in die Train-/Testdaten gekommen sind. Aus dem Grund wären die CV-Scores wertlos. Dies deutet daraufhin, dass die Stratifizierung nötig wäre.

Dafür wurden mehrere Ansätze getestet zum einen Stratified ShuffleSplit, StratifiedKFold, als auch die Stratifizierung direkt bei der Aufspaltung der Daten mit Train -Test-Split.

Stratified ShuffleSplit funktionierte gut zum Aufteilen der Test- und Trainingsdaten, beseitigte aber nicht alle Probleme. Auch kann man dies nicht nutzen umd damit die Cross-Validation durchzuführen. Dafür benötigt man StratifiedKfold. Bei einigen Algorithmen stelle sich heraus, dass es besser ist, direkt bei train-test-split zu stratifizieren, anstelle von Stratified ShuffleSplit.

Im Endeffekt wurden die Daten zweimal stratifiziert einmal beim Einteilen der Test- und Trainingsdaten (direkt durch Einfügen des stratify-parameters), zum anderen für die Cross-Validation mit Kfold.

**Nebenüberlegungen**

Es wurde versucht die Werte vom Decision Tree zu verbessern, in dem Seed und Test-Size zu variieren. Die Testgröße wurde 0.01 bis 0.50 variiert. Da erwartet wurde, dass der Score erst ab 20% akzeptabel wird und es ab 50% schlechter würde. Im Falle des Seeds 1 zu 255.

Dabei zeigte sich, dass der Score bei 49% am besten war, aus diesem Grund wurde eine weitere Schleife gestartet mit einer Testgröße von 0.01 bis 0.99. Allerdings wurde dies verworfen.

LogisticRegressionCV wrude verworfen, da dieser Algorithmus Probleme mit dem Speicherplatz ergab.

**Zusammenfassung der Ergebnisse**

Genutzt wurde Seed 42 und Testsize 0.2.

**In der Tabelle sind die Werte ohne Gridsearch aufgeführt:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Name** | **Test\_score** | **Training\_score** | **Fitting** | **Scaler** | **Balanced accuracy** | **F1s-score average** | **MSE** |
| Logistic Regression | 58% | 60% | Underfitting | Standard | 31.5% | 27% | 1.22 |
| SVM.SVC | 40% | 43% | Underfitting | Standard | 25% | 24% | 0.5294 |
| Decision Tree | 57% | 53% | Underfitting | none | 27% | ? | 0.5906 |
| Bayes Gaussian | ­57% | 54% | Underfitting | none | 30% | 29% | 0.75 |

In der nicht optimierten ersten Version ergeben Decision Tree und Logistic Regression den besten Test und Trainingscore mit circa 60%. Die beste Variante der Bayes-Modelle ist GaussianNB mit circa 57%, SVM liegt darunter mit circa 40%.

Alle Modelle ergeben underfitting und keines kann die 8. Klasse, also die beste Qualität erkennen.

In der balanced accuracy liegt GaussianNB mit 30% ist das beste Model, oberhalb der SVM mit 25%, Decision Tree liegt dazuwischen mit 27%.

Bezüglich der Sklaierung wurde der StandardScaler benutzt. Dieser ergibt bei DecisionTree wie zu erwarten keinen Effekt. Beim GaussianNB führt er zu einem Fehler, weshalb im Falle der Bayes-Modelle keine Skalierung genutzt wurde.

Bezüglich MSE ergibt SVM den besten Wert und MSE den schlechtesten. Das SVM den besten könnte daran liegen, dass einige Ausreißer enthalten sind und damit SVM besser klar kommt als andere Algorithmen.

Die Logistic Regression wird durch starke Korrelationen gestört und davon sind vor allen in den unbereinigten Daten noch einige drin.

**Kurze Beschreibung einzelner Modelle (werden größtenteils extra beschrieben)**

In Fall von SVC konnte weder der Klasse 3 noch die Klasse 8 erkannt werden. Dies lässt den Schluss zu, dass von diesen Klassen nicht genug Daten vorhanden sind. Das Model war stark voreingenommen mit einem gewichteten Durchschnitt von 45% und der maximalen Support-number gehört zur Klasse 5. Bezogen auf dem Mittelwert aller Klassen (macro Avg F1 score) zeigt es eine niedrige Performanz um 24%

**Grid-search Optimierung**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Name** | **Test\_score** | **Training\_score** | **Fitting** | **Scaler** | **Balanced accuracy** | **F1s-score average** | **MSE** |
| Logistic Regression | 58% | 60% | Under | Standard | 30% | 27% | 1.22 |
| SVM.SVC | 52% | 64% | Underfitting? | Standard | 32% | 30% | 0.5625 |
| Decision Tree | 73% | 57% | overfitting | none | 29% |  | 0.54 |
| Bayes Gaussian | 57% | 56% | Underfitting | none | 26% | 26% | 0.68 |

Beim Bayes Modell wurde der Test-Score durch den Grid-Search etwas besser, dagegen sind die anderen Scores etwas verschlechtert. Und das, obwohl die Default-Werte mit in der Parametersuche enthalten sind, eventuell hält die Parametersuche den Trainings- und Testscore für wichtiger als die anderen Scores. Bei SVC verbessern sich alle Werte durch die Parameteroptimierung.

SVC ergibt die beste Balanced accuracy gefolgt von Decision Tree, beide Algorithmen kommen gut mit Ausreißern klar, was das Ergebnis verbessern könnte.

Die MSE-Verteilung ist nach Gridsearch ähnlich wie vor Gridsearch, somit gilt dort dieselbe Argumentation. Im Fall von Bayes verbessert sich der MSE etwas.

**Zusammenfassung der besten Parameter**

LogisticRegression, GridSearch: LogisticRegression(C=1, max\_iter=10000, penalty="l2", solver="lbfgs")

the best parameters, SVC, GridSearch: {'classifier\_\_C': 1, 'classifier\_\_degree': 3, 'classifier\_\_gamma': 'scale', 'classifier\_\_kernel': 'poly', 'scaler': StandardScaler()}

the best estimator, SVC, GridSearch: Pipeline(steps=[('scaler', StandardScaler()), ('classifier', SVC(C=1, class\_weight='balanced', kernel='poly', random\_state=42))])

GaussianNB (priors = None, var\_smoothing: np.float64(1.9306977288832496e-05), also **1.93 × 10⁻⁵** oder **0.0000193**.

GridSearch-Score ('DecisionTreeClassifier') < 0.70 ?: 0.5777618061133895

Beste parameter: {'criterion': 'gini', 'max\_depth': 6, 'max\_features': 9, 'min\_samples\_split': 4, 'splitter': 'random'}

**Konfusionmatrix der einzelnen Algorithmen, wenn möglich vor und nach Gridsearch**

*Vor GridSearch*

Die Logistische Regression ohne Normalisierung erkennt weder die schlechteste Klasse noch die beste. Alle anderen Klassen werden als Klasse 5 oder 6 vorhergesagt. Also den Klassen von welchen am meisten Daten vorhanden sind und man somit auch bei einer blinden Vorhersage, beziehungsweise beim zufälligen Raten. Die 7. Klasse wird noch teilweise erkannt und als Qualitätsstufe 6 einsortiert.

Der Decision Tree erkennt nur die mittleren Klassen und im Gegensatz zur logistischen Regression auch seltener die 7. Klasse eingeschätzt.

Ein Bild, das Text, Screenshot, Diagramm, Rechteck enthält.

Automatisch generierte Beschreibung

Gaussian Bayes macht Vorhersagen über alle Klassen und ist dabei meist eine Klasse entfernt. Er eignet sich somit zur Vorhersage, wenn man nicht nur 5 und 6 als Vorhersage bekommen möchte. SVC sagt wie die beiden anderen Algorithmen hauptsächlich 5 und 6 voraus. Ein Bild, das Text, Screenshot, Zahl, Diagramm enthält.

Automatisch generierte Beschreibung

Bei Bayes werden alle Klassen vorhergesagt meist ungefähr eine Klasse daneben. Er ist damit der beste, wenn man nicht nur Klasse oder 6 als Ergebnis haben möchte. Gaussian NB braucht Normalverteilungen, um funktionieren zu können. Die Werte sind zwar nicht genau normalverteilt, aber nahe genug, um eine Vorhersage zu ergeben.

SVC sagt wie die anderen Algorithmen vor allem Klasse 5 und 6 voraus.

*Konfusionmatrix nach Gridsearch*

Ein Bild, das Text, Screenshot, Diagramm, Rechteck enthält.

Automatisch generierte Beschreibung

Es ergeben sich keine Änderungen durch Grid-Search bei der Logistischen Regression. Beim Decision Tree werden nach Grid-Search die höheren Klassen etwas besser erkannt.

Ein Bild, das Text, Screenshot, Diagramm, Zahl enthält.

Automatisch generierte Beschreibung

Nach Gridsearch sagt auch Gaussian NB alle Klassen als5 oder 6 voraus. Die Scores wurden dadurch minimal besser, weil es überwiegend Klasse 5 und 6 in den Daten gibt. Grid-Search scheint auf die anderen Parameter optimiert zu haben.

SVC zeigt keine großen Änderungen.

**Zweite Runde ausprobieren der zusammengelegten Werte**

Es wurde ausprobiert, ob das Zusammenlegen der Daten erfolgreich war. Beim Zusammenlegen der Spalten, ging es vor allem darum, die starken Korrelationen zu beseitigen, da diese zu Probleme führen könnten.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Name** | **Test\_score** | **Training\_score** | **Fitting** | **Scaler** | **Balanced accuracy** | **F1s-score average** | **MSE** |
| Logistic Regression | 59,19% | 59,61% | under | Standard | 29% | 28% | 2.85 |
| SVM.SVC | 37% | 40% | Underfitting | Standard | 25% | 22% | 0.5625 |
| Decision Tree |  |  |  |  |  |  |  |
| GaussianNB | 0.56 | 0.57 | Underfitting | none | 0.3 | 0.29 | 0.67 |

Im Falle des GaussianNB werden durch das Zusammenlegen der Spalten die Training- und Testwerte leicht verbessert. Balance Accuracy und F1 bleiben gleich, MSE wird leicht schlechter. Das Zusammenlegen der Spalten, sollte vor

Mit Grid-Search

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Name** | **Test\_score** | **Training\_score** | **Fitting** | **Scaler** | **Balanced accuracy** | **F1s-score average** | **MSE** |
| Logistic Regression | 59,19% | 59,61% | under | Standard | 28.99% | 28% | 2.85 |
| SVM.SVC | 51% | 60% | Underfitting | Standard | 27% | 27% | 0.5110 |
| Decision Tree |  |  |  |  |  |  | 0.563 |
| Bayes Gaussian | 57% | 56% | Underfitting | None | 0.25 | 0.25 | 0.67 |

the best parameters, SVC, GridSearch: {'classifier\_\_C': 1, 'classifier\_\_degree': 3, 'classifier\_\_gamma': 'auto', 'classifier\_\_kernel': 'poly', 'scaler': RobustScaler()}

the best estimator, SVC, GridSearch: Pipeline(steps=[('scaler', RobustScaler()), ('classifier', SVC(C=1, class\_weight='balanced', gamma='auto', kernel='poly', random\_state=42))])

Bayes: Best parameters Gaussian: {'priors': None, 'var\_smoothing': np.float64(0.026826957952797274)}

**Konfusion Matrix**

***Ohne Gridsearch***

Ein Bild, das Text, Screenshot, Rechteck, Diagramm enthält.

Automatisch generierte Beschreibung

Bei der Logistischen Regression ergeben sich durch das Zusammenfassen der Spalten keine Änderung. Da die Zusammenfassung gemacht wurde, um die Korrelationen zu verringern und dadurch unter anderem die Logistische Regression zu verbessern, zeigt sich eindeutig, dass das Ziel nicht erreicht wurde. Dies könnte damit zussammenhängen, dass dennoch starke Korrelationen drin verblieben sind, wie zum Bespiel zwischen Zitronensäure und Acidity, welche drin gelassen wurden, weil Zitronensäure einer der wenigen Angaben bezüglich erwünschter Stoffe ist.

Ein Bild, das Text, Screenshot, Diagramm, Zahl enthält.

Automatisch generierte Beschreibung

Bayes wird durch das Zusammenfassen der Spalten schlechter. Dies lässt wieder darauf schließen, dass die Zusammenfassung nicht effektiv war. Und Informationen verschleiert wurden, welche vorher Bayes bei der Vorhersage geholfen hatten. SVC verändert sich nicht viel.

**Warum so schlecht:**

Der Datensatz enthält zu wenige Daten, um damit Machine Learning zu betreiben. Dazu kommt, dass diese unbalanciert sind. Es sind deutlich zu viele aus den mittleren Stufen vorhanden und zu wenige aus den schlechten und guten Stufen.

Zum Bearbeiten der Daten, um eine balancierteres Ergebnis zu bekommen hätte es verschiedene Strategien gegeben, welche jedoch aus Zeitgründen nicht durchgeführt werden konnten.

So hätte man Daten erfinden können, um aufzufüllen. Man hätte auch den Klassen unterschiedliches Gewicht gegeben können. Das wurde bei GaussianNB probiert, aber ohne den gewünschten Erfolg zu erzielen.

Auch hätte man aus den mittleren Spalten so viele Daten löschen können, bis in allen Klassen gleich viel vorhanden sind. Dies hätte einige Probleme beseitigt. Allerdings wären es dann noch weniger Daten gewesen und somit wiederum äußert schwierig gewesen.

Zusätzlich ist die Einteilung von Wein in Qualitätsstufen subjektiv, zum Bespiel trinken einige liebe trockenen Wein, während andere süßen Wein bevorzugen. Auch bei der Säure gibt es völlig unterschiedliche Vorlieben. So bevorzugen manche etwas Säure, da es frischer ist. Andere wollen überhaupt keine Säure und bekommen auch von Wein mit zu viel Säure Sodbrennen, so dass sie diese Weine gar nicht mögen.

Die Zusammenfassung der Spalten hat zu keiner Verbesserung geführt. Dies liegt vermutlich daran, dass die Zusammenfassung zwar aus chemischer Hinsicht Sinn ergeben, aber nicht auf das Machine-Learning optimiert sind. Auch wurde vergessen die volatile Acid-Spalte zu löschen, somit bekamen die Säurespalten zu viel Gewicht. Es konnten auch nicht alle starken Korrelationen gelöscht werden, da sich teilweise eine Löschung der Spalten nicht anbot. So wäre es nicht sinnvoll gewesen die Zitronensäure zu löschen, da diese zu den Geschmackstoffen gehört, welche gut für den Wein sind. Eine der wenigen, die meisten Angaben sind eher Fehlgeschmäcker. Die Dichte hängt nicht nur von der Spalte ab, mit der zusammengefasst wurde, sondern auch mit anderen Werten, somit wurde sie auch nicht gelöscht, um nicht Informationen zu verlieren, die hilfreich sein könnten. Dies führte aber insgesamt dazu, dass sich das gewünscht Ergebnis und zwar die starken Korrelationen loszuwerden nicht erreichen ließ. In einem weiterem Optimierrungsschritt müsste man die Zusammenfassung optimieren und mit und ohne die Spalten, welche eine starke Korrelation verursachen probieren.

Man hätte noch mehr Grafiken machen müssen, um zu gucken, ob es Eigenschaften gibt, welche nur für eine bestimmte Klasse zutreffen. So haben die höchsten Qualitätsstufen den meisten Alkohol. Dies hätte man für die Vorhersage einbauen können und diese grundsätzlich als Klasse 8 vorhersagen können.