Aufgabe 1: Iris-Klassifikation

Ziel: Klassifikation der Iris-Blumen in die drei Arten *setosa*, *versicolor* und *virginica*.

Vorgehen: Zunächst wurde der Iris-Datensatz geladen. AnschlieSSend erfolgte eine Aufteilung in Trainings- (80) und Testdaten (20) mit festem Seed (R: `set.seed(42)`, Python: `random_state=42`, RapidMiner: `random_seed=42`).

Zur Modellbildung wurde in R der folgende Befehl verwendet:

```r
# Laden und Überblick
data(iris)
set.seed(42)
# Split in Trainings- und Testdaten (80/20)
idx <- sample(1:nrow(iris), size = 0.8 * nrow(iris))
train <- iris[idx, ]
test  <- iris[-idx, ]
```

In Python kam folgender Code zum Einsatz:

```python
iris = datasets.load_iris(as_frame= True)
X = iris.data
y = iris.target
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)
```

In Orange wurde der Workflow mit den Modulen File → Data Sampler (80/20) → Random Forest → Test & Score realisiert. In RapidMiner wurde ein äquivalenter Prozess mit den Schritten Read CSV → Split Data → Random Forest → Apply Model → Performance umgesetzt.

Zur Evaluation wurden Metriken wie Accuracy, Precision, Recall, F1-Score und die Confusion Matrix herangezogen.
In R:

```r
library(randomForest)
model_rf <- randomForest(Species ~ ., data = train)
pred_rf  <- predict(model_rf, test)
table(pred_rf, test$Species)
```

In Python:

```python
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report

clf = RandomForestClassifier(random_state=42)
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
print(classification_report(y_test, y_pred))
```

Ergebnisse (Python/R):
R:

pred_rf	setosa	versicolor	virginica
setosa	9	0	0
versicolor	0	10	1
virginica	0	1	9

Python:

precision	recall	f1-score	support	
virginica	1.00	1.00	1.00	10
setosa	1.00	1.00	1.00	9
versicolor	1.00	1.00	1.00	11
accuracy	1.00	30		
macro avg	1.00	1.00	1.00	30
weighted avg	1.00	1.00	1.00	30

Interpretation: Das Modell zeigt eine sehr gute Trennschärfe, insbesondere zwischen *setosa* und den anderen beiden Arten.

Aufgabe 2: Algorithmusvergleich (Decision Tree, Naive Bayes, SVM)

Ziel: Vergleich der Klassifikationsleistung dreier Algorithmen auf demselben Datensatz und Splits.

Vorgehen: Als Datenbasis diente erneut der Iris-Datensatz. Die Daten wurden wie in Aufgabe 1 aufgeteilt. Es wurden drei Modelle trainiert:

Decision Tree: R: `rpart()`, Python: `DecisionTreeClassifier()`, Orange: Decision Tree-Widget, RapidMiner: Decision Tree → Apply Model.

Naive Bayes: R: `e1071::naiveBayes()`, Python: `GaussianNB()`, Orange: Naive Bayes-Widget, RapidMiner: Naive Bayes → Apply Model.

SVM: R: `e1071::svm()`, Python: `SVC()`, Orange: SVM-Widget, RapidMiner: SVM → Apply Model.

Die Modelle wurden mit Accuracy, Precision, Recall, F1 und ROC-AUC (für binäre und multiklassige Klassifikation) verglichen.
Hier als Beispiel der Decision Tree der durch das Python script erzeugt wurde:

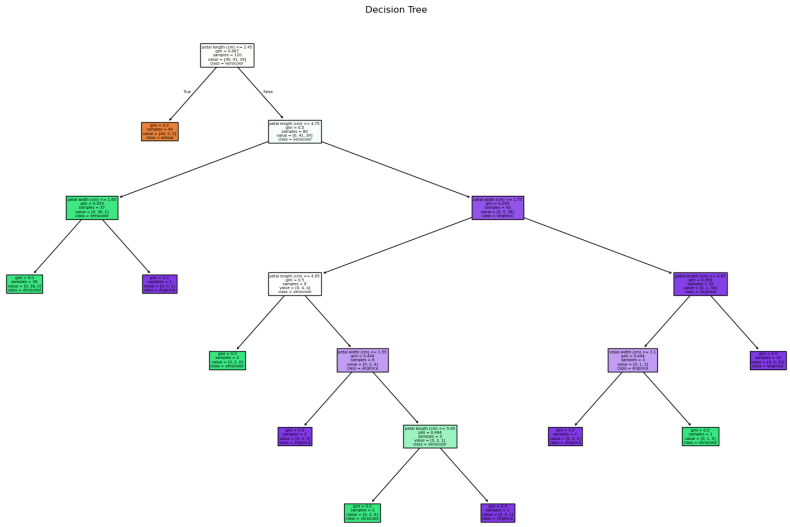


Abbildung 1: Auf meinem Discord ist das ganze auch noch mal in Hochauflösender link

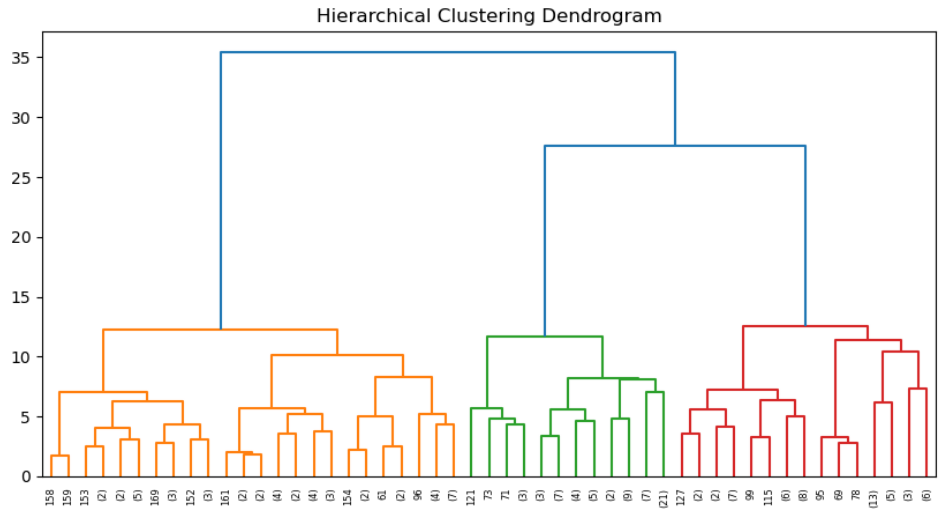
Interpretation: Alle Confusion Matrizen sind genau so wie die obigen aus Aufgabe 1 heiSSt alle classen wurden richtig zugeordnet.

Aufgabeă3: Unüberwachtes Clustering (RotweinDaten) mit Python

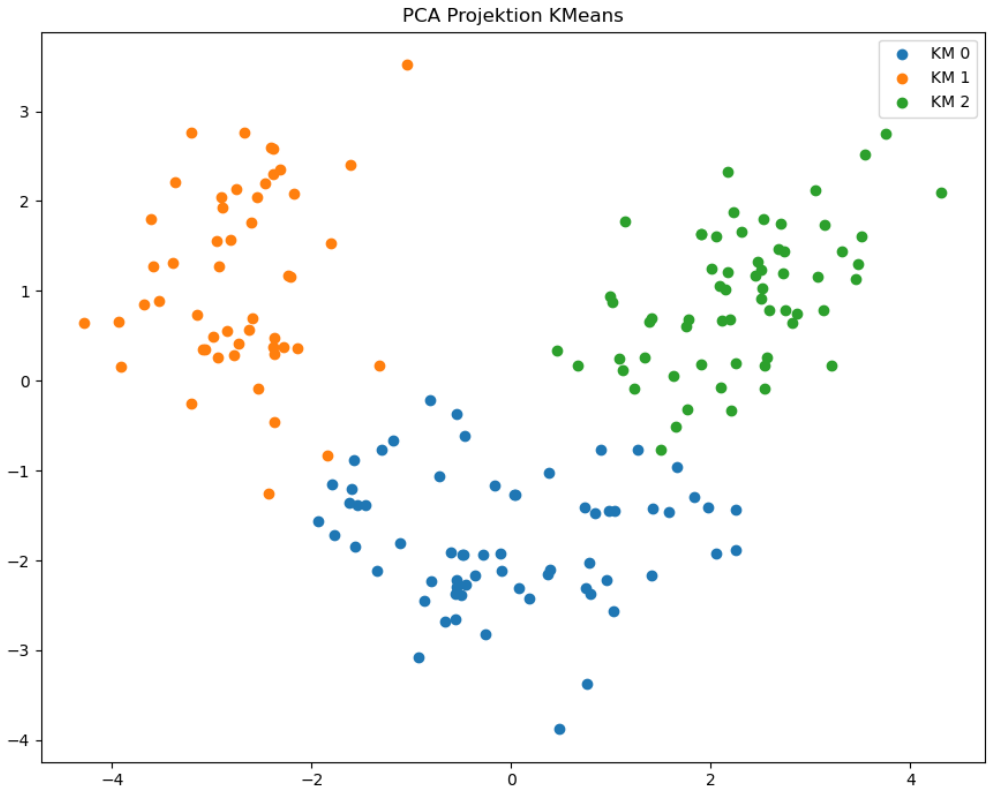
Das Ziel dieser Aufgabe ist es, im WineDatensatz mithilfe zweier ClusteringVerfahren, nämlich KMeans und hierarchischem Clustering, verborgene Gruppen in den Daten zu identifizieren. Dabei konzentrieren wir uns auf die chemischen KenngröSSen der Weine und untersuchen, ob sich anhand statistischer Kriterien sinnvolle Cluster bilden lassen.

Für das KMeansClustering wird das Modell mit `KMeans(n_clusters=3, random_state=42)` trainiert; parallel dazu wird ein hierarchisches Modell mit WardLinkage über `AgglomerativeClusterin linkage='ward')` erstellt und sein Dendrogramm mit `scipy.cluster.hierarchy.dendrogram` visualisiert. Die resultierenden Cluster werden anschlieSSend durch die Berechnung der mittleren FeatureWerte pro Gruppe charakterisiert und im zweidimensionalen Raum der ersten beiden Hauptkomponenten (PCA) farblich dargestellt.

Daraus ergibt sich dieses Hirachische Cluster:



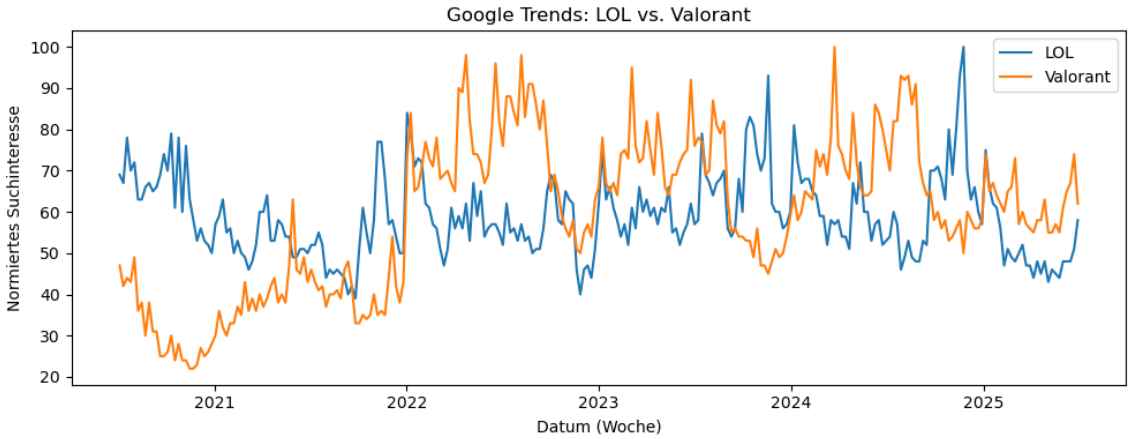
und diese PCA:



Als Ergebnis zeigt sich, dass drei Cluster die beste Kompromisslösung zwischen Kompaktheit und Trennschärfe liefern.

Aufgabe 4: Google Trends Clustering

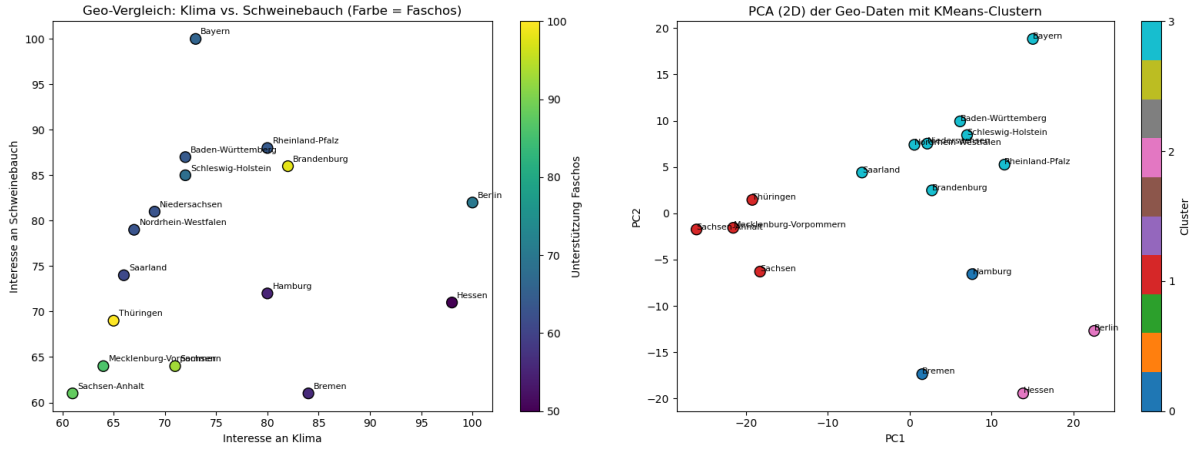
Ich habe mir Trend daten zu 4 verschiednen themen rausgesucht und noch eine extra tabelle mit hilfe von den Wahlergebnissen der letzten buindestagwahl erstellt. Einmal Vergleiche ich Klima mit dem Fleischintresse und vergleiche dann noch ob die Geodaten von den Suchanfragen im zusammenhang zu den Wahlwerten der AFD haben.
Dann habe ich noch probiert den intressens umschwung von League of Legends auf Valorant einzufangen aber das hat so semi geklappt.



Wie man sieht sind die beiden spiele seit dem Valorant aus der Beta ist ungefähr auf dem selben intressen level, dass könnte daran liegen das viele Lol spieler auch valorant spieler sind und umgekehrt.
Jetzt zum interressanten teil:
ich habe die Suchterme Klima und Schweinebauch als gegenidee des Weltkonsums ge-

Region	Value_Klima	Value_Schwein	Value_Faschos
Berlin	100	82	15.2
Hessen	98	71	17.7
Bremen	84	61	15.2
Brandenburg	82	86	34.4
Rheinland-Pfalz	80	88	19.2
Hamburg	80	72	11.0
Bayern	73	100	17.4
Schleswig-Holstein	72	85	16.1
Baden-Württemberg	72	87	19.4
Sachsen	71	64	38.5
Niedersachsen	69	81	17.6
Nordrhein-Westfalen	67	79	16.4
Saarland	66	74	21.6
Thüringen	65	69	38.7
Mecklenburg-Vorpommern	64	64	37.0
Sachsen-Anhalt	61	61	38.8

genübergestellt. Dazu habe ich die Wahlwerte der letzten Bundestagswahl für die AFD per Bundesland raus gesucht (Spalte Faschos in Porzent).



Wie man sehen kann sind viele der alten Ost und Westländer in je einer Gruppe. Zusätzlich entstehen noch zwei weiter gruppen die aus Hessen und Berlin, und die Aus Bremen und Hamburg. Die Berlin, Hessen Gruppe fällt auf durch starkes Klima interresse und swächere AFD Werte wärend die andere extra Gruppe wenig intresse an weder Klima noch Schweinefleisch hat aber auch eher schwache AFD Werte hat. In der ersten Graphik (links) sieht man auch das die Ost länder generell weniger diese beiden dinge gegoogled haben, vielleicht liegt das an erst späteren zugang zu weit verbreiteten Internet oder and einer generellen politischen unintresse.

Github

Wie immer sind alle meine benutzten Dateien auf meinem Github zu finden.