Wine Quality Assessment

Projektarbeit Deep Learning

Alexandre Bouyer · Februar 2024

Motivation

- Vorhersage sensorischer Weinbewertung anhand physikalisch-chemischer Parameter (Alkohol, pH-Wert usw.)
- Sensorische Prüfung Teil des Zertifizierungsprozesses
- Besseres Verständnis der Zusammenhänge
- Verbesserungen bei der Weinproduktion
- Unterschiedliche Kundenpräferenzen besser adressieren
- Preisgestaltung

Weindaten

- Je ein Dataset mit Rot- und Weißweindaten (Proben aus Vinho Verde aus der Minho Region in Portugal)
- Elf numerische Features (Labormessungen). Extremwerte sind als valide Datenpunkte zu betrachten
- Label ist ordinal skaliert, von 0 (schlechteste Benotung) bis 10
- Label als Median der Bewertungen einer Probe durch mindestens drei Verkoster (in Blindverkostungen)
- Also keine Duplikate?
- Keine fehlenden Werte

Ziele im Projekt

- Ergebnisse aus Paper reproduzieren, möglichst verbessern
- Mit Ergebnissen und Methoden aus ML (scikit-learn) vergleichen
- Unterschiedliche Ansätze ausprobieren und vergleichen, u. a.:
  - Regression vs. Klassifikation
  - verschiedene Verlustfunktionen und Metriken
  - Hyperparameter-Tuning mit KerasTuner:
    - Random Search
    - Hyperband
    - Bayesian Optimization

Vorgehensweise · Aufbau Programme

- Datasets separat verarbeiten
- Preprocessing. Duplikate entfernen (als Variante)
- Normalisierung
- Für Regression und Klassifikation:
    - Beste Hp suchen (alle drei Tuner-Versionen)
    - Modelle mit den gefundenen Hp bilden und trainieren
    - Evaluieren und vergleichen

Preprocessing · Datenübersicht

Rotweine

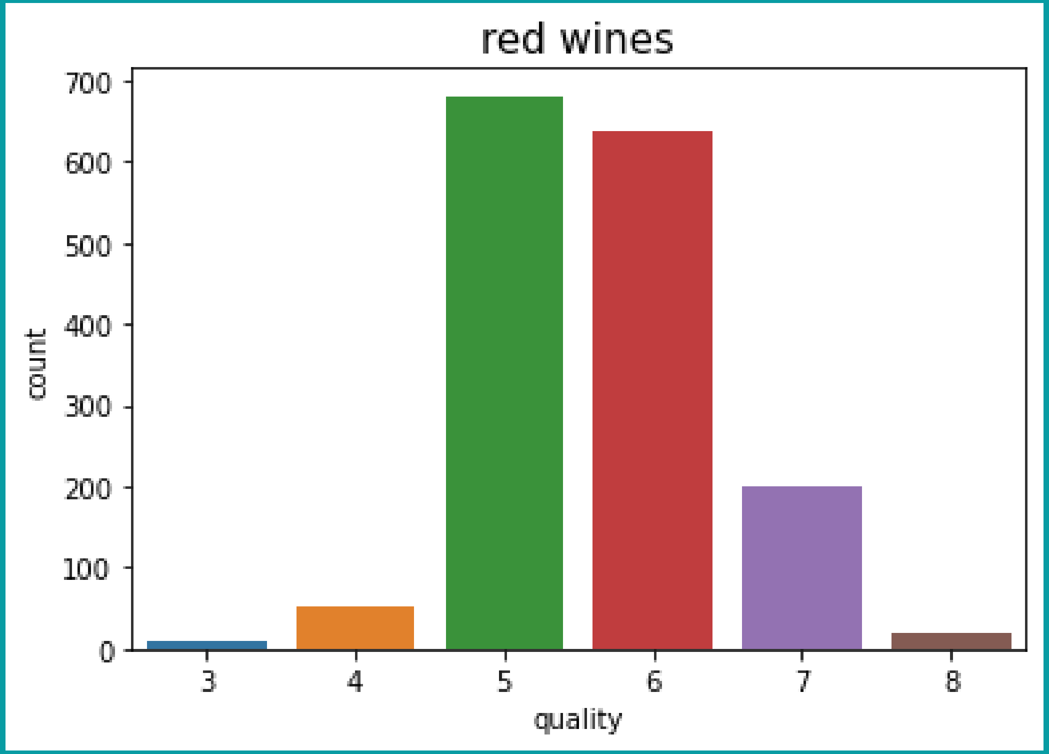
```
./data/winequality/winequality-red.csv
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1599 entries, 0 to 1598
Data columns (total 12 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   fixed acidity         1599 non-null   float64
 1   volatile acidity      1599 non-null   float64
 2   citric acid           1599 non-null   float64
 3   residual sugar        1599 non-null   float64
 4   chlorides             1599 non-null   float64
 5   free sulfur dioxide   1599 non-null   float64
 6   total sulfur dioxide  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
dtypes: float64(11), int64(1)
memory usage: 150.0 KB
Number of duplicates: 460, Percentage of duplicates: 29%
Shape after drop: (1359, 12)
```

Weißweine

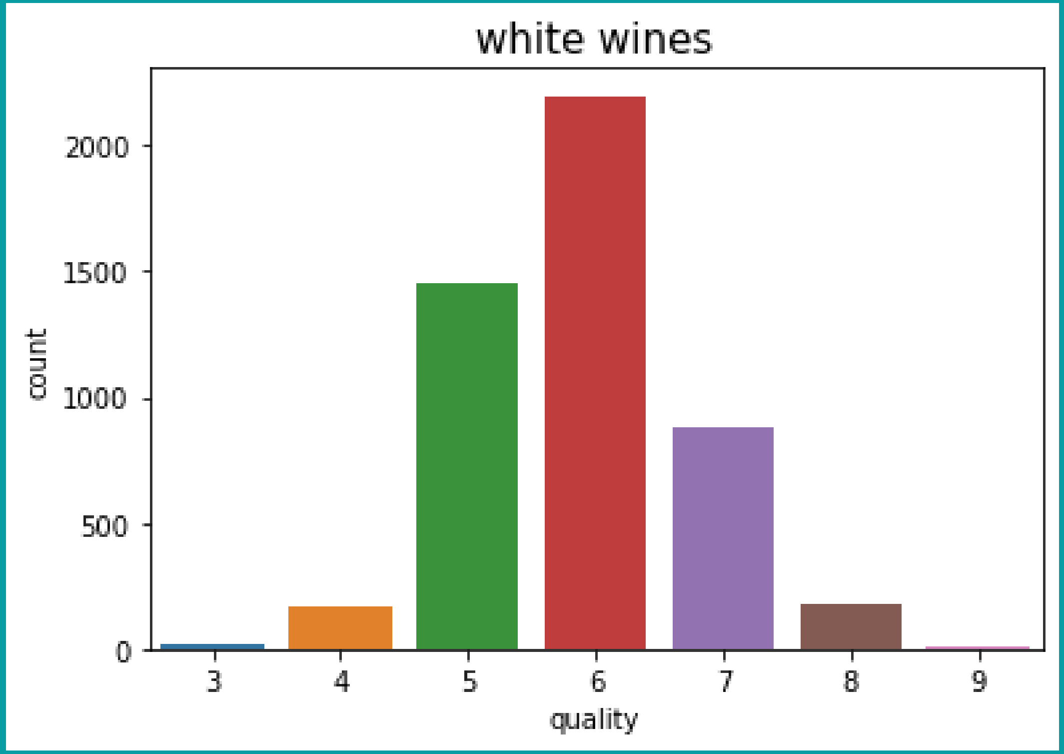
```
./data/winequality/winequality-white.csv
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4898 entries, 0 to 4897
Data columns (total 12 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   fixed acidity         4898 non-null   float64
 1   volatile acidity      4898 non-null   float64
 2   citric acid           4898 non-null   float64
 3   residual sugar        4898 non-null   float64
 4   chlorides             4898 non-null   float64
 5   free sulfur dioxide   4898 non-null   float64
 6   total sulfur dioxide  4898 non-null   float64
 7   density               4898 non-null   float64
 8   pH                    4898 non-null   float64
 9   sulphates             4898 non-null   float64
 10  alcohol               4898 non-null   float64
 11  quality               4898 non-null   int64
dtypes: float64(11), int64(1)
memory usage: 459.3 KB
Number of duplicates: 1709, Percentage of duplicates: 35%
Shape after drop: (3961, 12)
```

Preprocessing · Verteilung der Labelwerten

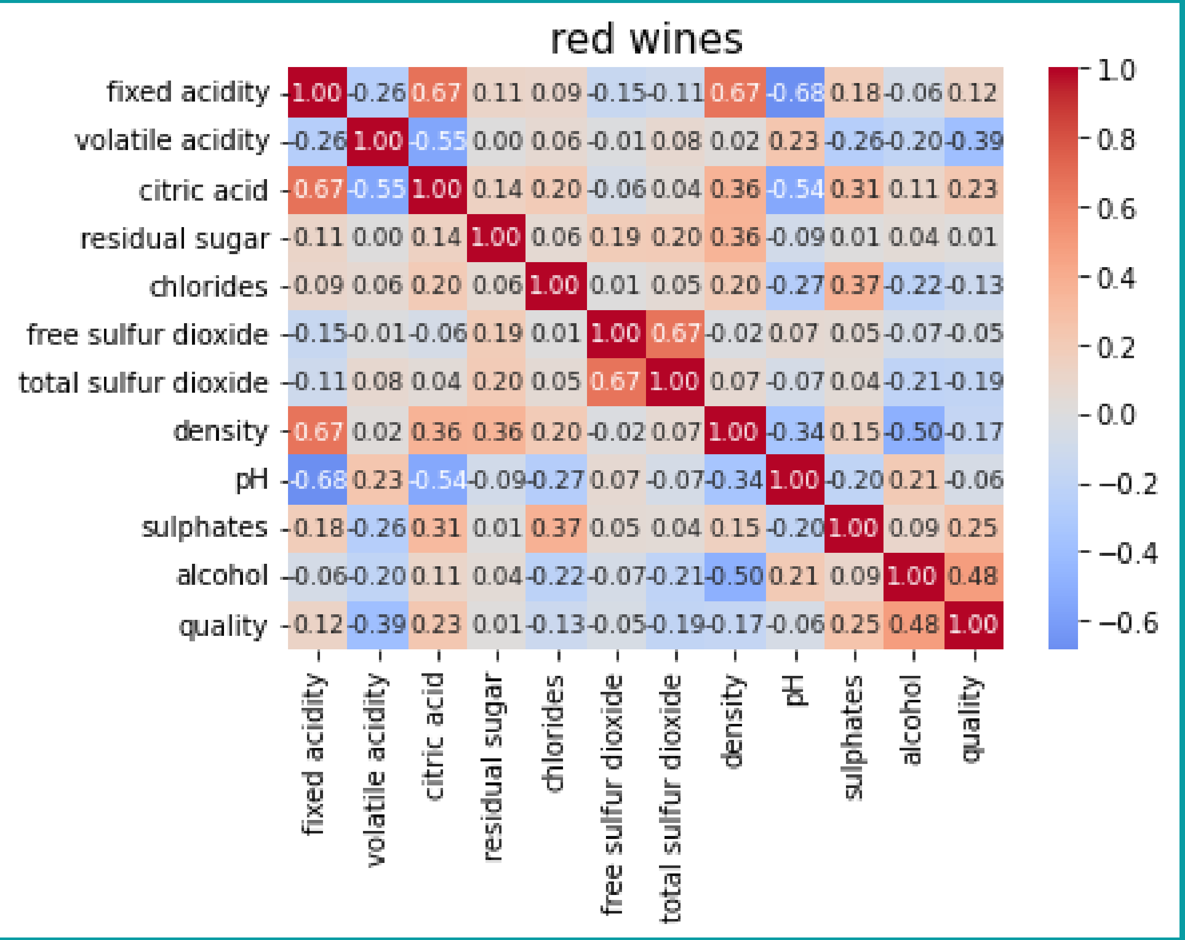
Rotweine



Weißweine

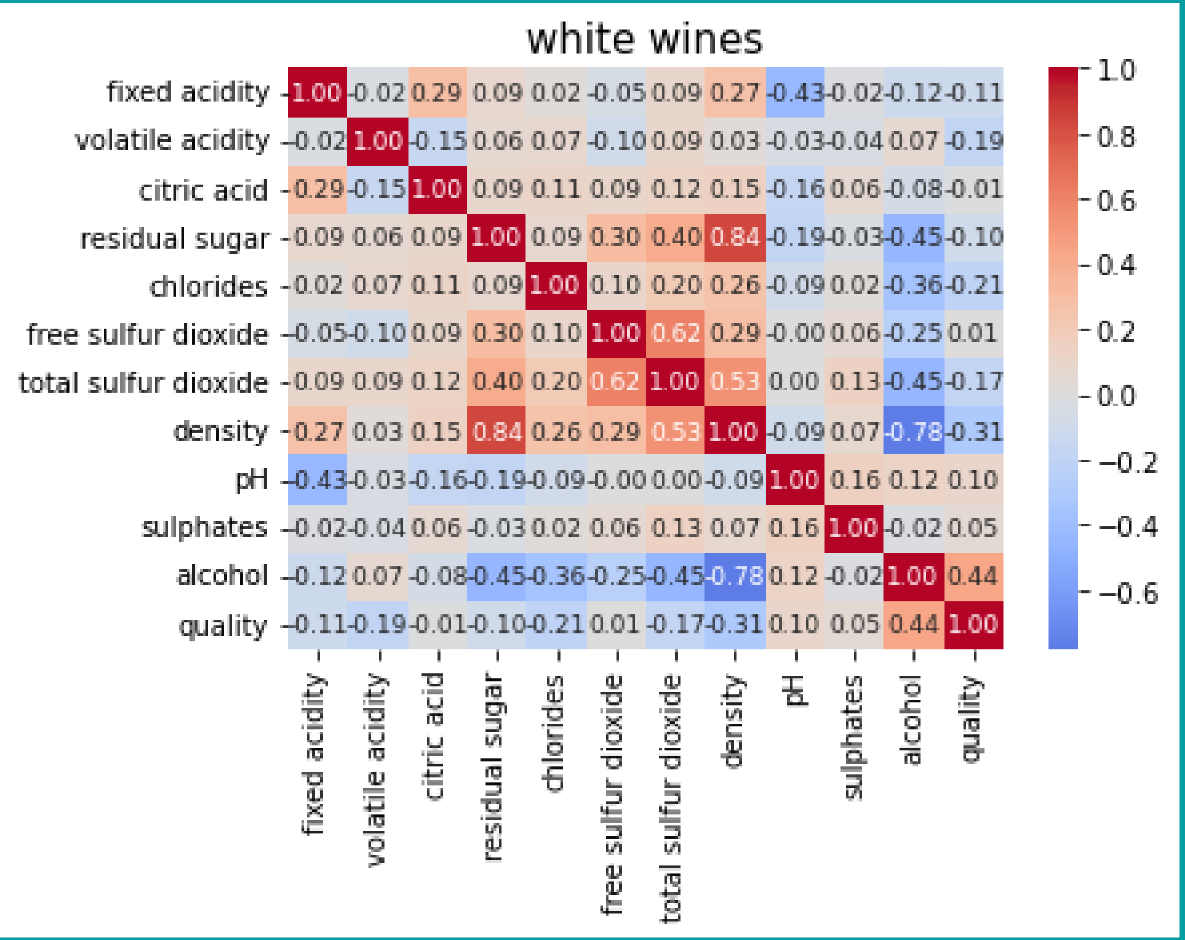


Preprocessing · Korrelationsmatrix Rotweine



Höchste Korrelationen Feature-Target (in Abslolutwerten): alcohol (+) > vol. acidity (-) > sulphates (+)

Preprocessing · Korrelationsmatrix Weißweine



Höchste Korrelationen Feature-Target (in Abslolutwerten): alcohol (+) > density (-) > chlorides (-)

Regression · Hyperparameter · Rot und Weiß

Rotweine

```
In [9]: best_trials_hp_band[0].summary()
Trial 0050 summary
Hyperparameters:
n_hidden: 6
n_units: 132
lr: 0.04396073111098763
optimizer: sgd
loss: mae
tuner/epochs: 12
tuner/initial_epoch: 4
tuner/bracket: 1
tuner/round: 1
tuner/trial_id: 0048
Score: 0.45514973998069763

In [10]: best_params_hp_band[0].values
Out[10]:
{'n_hidden': 6,
 'n_units': 132,
 'lr': 0.04396073111098763,
 'optimizer': 'sgd',
 'loss': 'mae',
 'tuner/epochs': 12,
 'tuner/initial_epoch': 4,
 'tuner/bracket': 1,
 'tuner/round': 1,
 'tuner/trial_id': '0048'}
```

Weißweine

```
In [38]: best_trials_hp_band[0].summary()
Trial 0016 summary
Hyperparameters:
n_hidden: 8
n_units: 86
lr: 0.0530105533703494
optimizer: sgd
loss: mae
tuner/epochs: 12
tuner/initial_epoch: 4
tuner/bracket: 2
tuner/round: 2
tuner/trial_id: 0012
Score: 0.5134857892990112

In [39]: best_params_hp_band[0].values
Out[39]:
{'n_hidden': 8,
 'n_units': 86,
 'lr': 0.0530105533703494,
 'optimizer': 'sgd',
 'loss': 'mae',
 'tuner/epochs': 12,
 'tuner/initial_epoch': 4,
 'tuner/bracket': 2,
 'tuner/round': 2,
 'tuner/trial_id': '0012'}
```

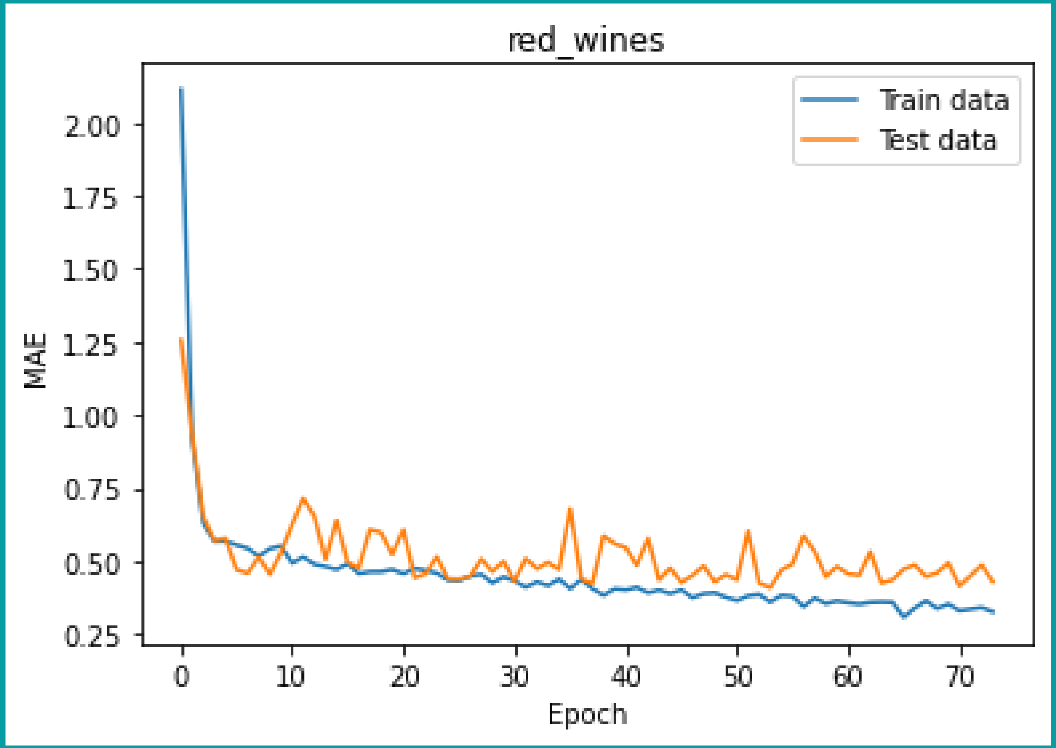
Regression · Model

```python
model = Sequential()
model.add(Input(shape=shape))

# red wines
if ds_name == 'red_wines':

    optimizer = SGD(learning_rate=0.04396)
    loss = 'mae'

    for _ in range(6):
        model.add(Dense(132, activation='relu'))

# white wines
else:
    optimizer = SGD(learning_rate=0.0530)
    loss = 'mae'

    for _ in range(8):
        model.add(Dense(86, activation='relu'))

model.add(Dense(1))

model.summary()

early_stop = EarlyStopping(monitor='val_loss', patience=20, verbose=1)

model.compile(loss=loss, optimizer=optimizer, metrics=['mae'])

model.fit(X_train, y_train, epochs=1000,
          validation_data=(X_test, y_test),
          callbacks=[early_stop])
```

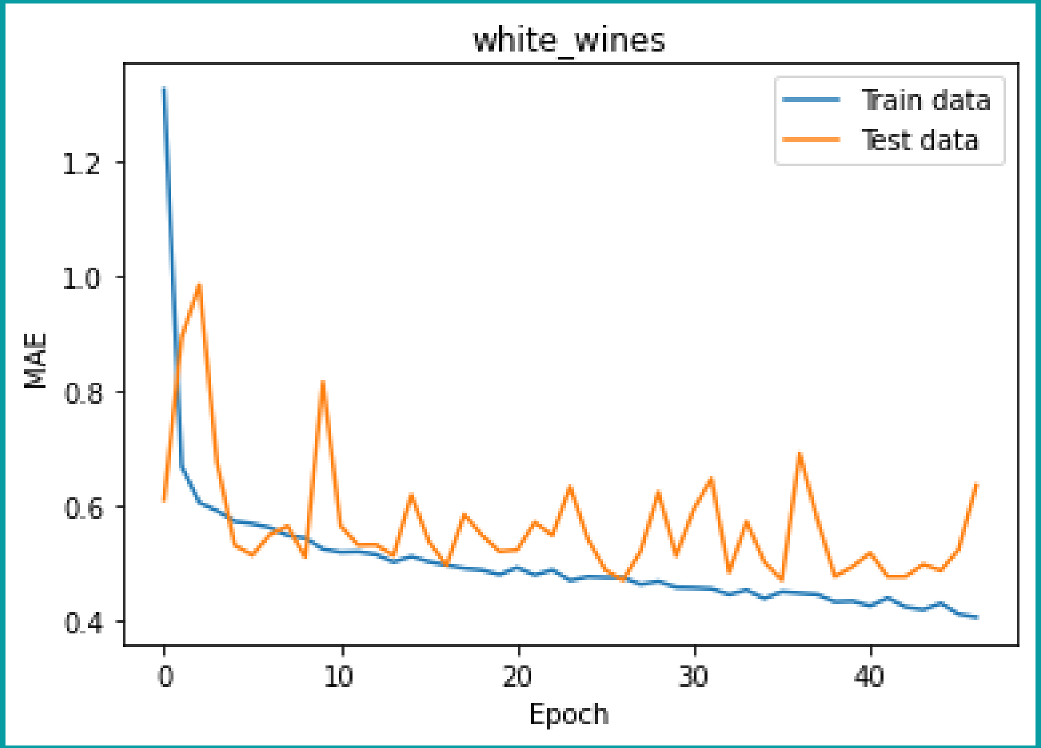
Regression · Ergebnisse Training · Rot und Weiß

Verlustfunktion = Metrik: MAE

Rotweine



Weißweine



Regression · Ergebnisse Training · Rot und Weiß

MAE und diverse Accuracy-Metriken

Rotweine

```
In [28]: mae = model.evaluate(X_test, y_test)
10/10 [==============================] - 0s 2ms/step - loss: 0.4299 - mae: 0.4299

In [29]: accuracy = keras.metrics.Accuracy()
    ...: accuracy.update_state(y_test, y_pred)
    ...: accuracy.result().numpy()
Out[29]: 0.671875

In [30]: accuracy_T(y_test, y_pred, 0.5)
Out[30]: 0.671875

In [31]: accuracy_T(y_test, y_pred, 1)
Out[31]: 0.96875
```

Weißweine

```
In [45]: mae = model.evaluate(X_test, y_test)
31/31 [==============================] - 0s 2ms/step - loss: 0.6361 - mae: 0.6361

In [46]: accuracy = keras.metrics.Accuracy()
    ...: accuracy.update_state(y_test, y_pred)
    ...: accuracy.result().numpy()
Out[46]: 0.5408163

In [47]: accuracy_T(y_test, y_pred, 0.5)
Out[47]: 0.5408163265306123

In [48]: accuracy_T(y_test, y_pred, 1)
Out[48]: 0.9581632653061225
```

Regression · Ergebnisse Paper · Rot und Weiß

The wine modeling results (test set errors and selected models; best values in bold)

	Red wine			White wine		
	MR	NN	SVM	MR	NN	SVM
MAD	0.50±0.00	0.51±0.00	**0.46**±0.00*	0.59±0.00	0.58±0.00	**0.45**±0.00*
$\text{Accuracy}_{T=0.25}$ (%)	31.2±0.2	31.1±0.7	**43.2**±0.6*	25.6±0.1	26.5±0.3	**50.3**±1.1*
$\text{Accuracy}_{T=0.50}$ (%)	59.1±0.1	59.1±0.3	**62.4**±0.4*	51.7±0.1	52.6±0.3	**64.6**±0.4*
$\text{Accuracy}_{T=1.00}$ (%)	88.6±0.1	88.8±0.2	**89.0**±0.2$^\diamond$	84.3±0.1	84.7±0.1	**86.8**±0.2*

Klassifikation · Hyperparameter · Rot und Weiß

Rotweine

```
In [10]: best_trial_hp_bayes[0].summary()
Trial 03 summary
Hyperparameters:
n_hidden: 4
n_units: 91
lr: 0.05279137469610778
optimizer: adam
Score: 0.668749988079071

In [11]: best_params_hp_bayes[0].values
Out[11]: {'n_hidden': 4, 'n_units': 91, 'lr': 0.05279137469610778,
'optimizer': 'adam'}
```

Weißweine

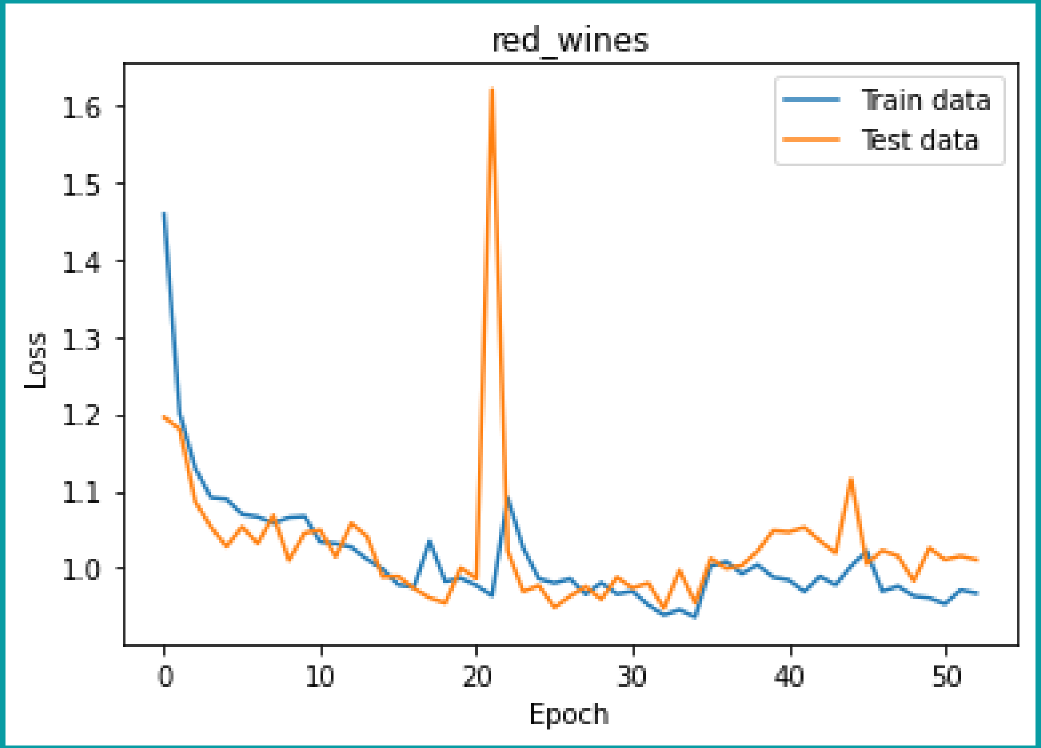
```
In [24]: best_trial_hp_bayes[0].summary()
Trial 06 summary
Hyperparameters:
n_hidden: 8
n_units: 185
lr: 0.0022114463484160314
optimizer: adam
Score: 0.6010203957557678

In [25]: best_params_hp_bayes[0].values
Out[25]:
{'n_hidden': 8,
 'n_units': 185,
 'lr': 0.0022114463484160314,
 'optimizer': 'adam'}
```

Klassifikation · Model

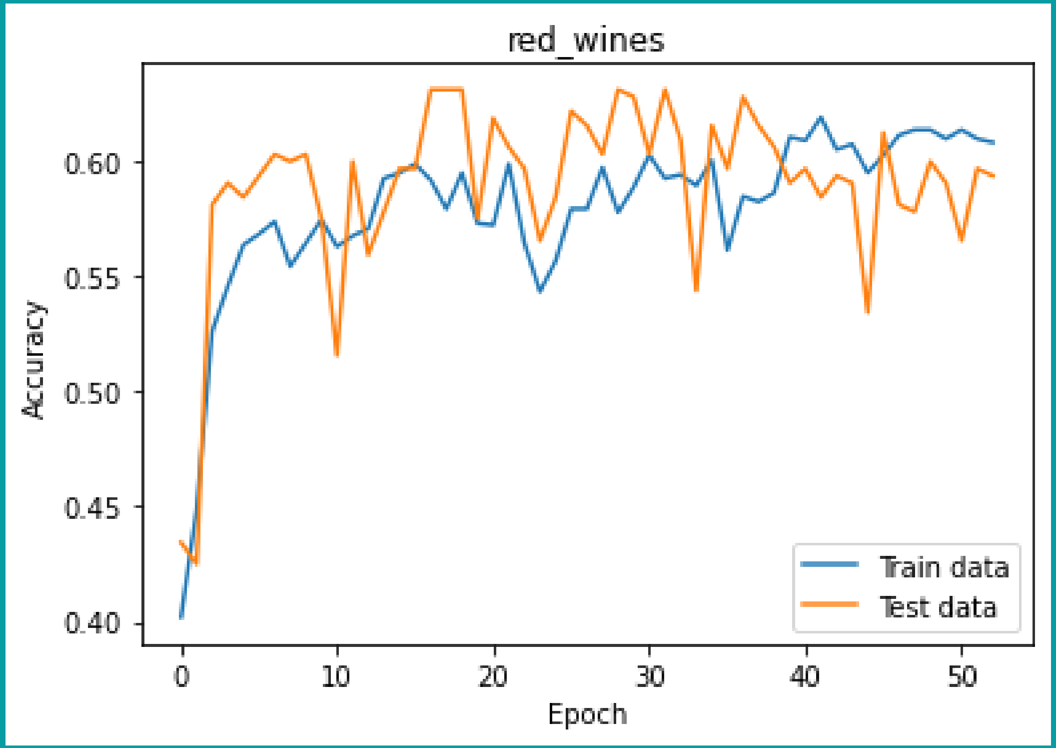
```
model = Sequential()
model.add(Input(shape=shape))

# red wines
if ds_name == 'red_wines':

    optimizer = Adam(learning_rate=0.05279)

    for _ in range(4):
        model.add(Dense(91, activation='relu'))

# white wines
else:
    optimizer = Adam(learning_rate=0.00221)

    for _ in range(8):
        model.add(Dense(185, activation='relu'))

model.add(Dense(output_shape, activation='softmax'))

model.summary()

early_stop = EarlyStopping(monitor='val_loss', patience=20, verbose=1)

model.compile(loss='categorical_crossentropy', optimizer=optimizer, metrics=['accuracy'])

model.fit(X_train, y_train, epochs=1000,
          validation_data=(X_test, y_test),
          callbacks=[early_stop])
```

Klassifikation · Ergebnisse Training · Rotweine

Verlustfunktion: CategoricalCrossentropy

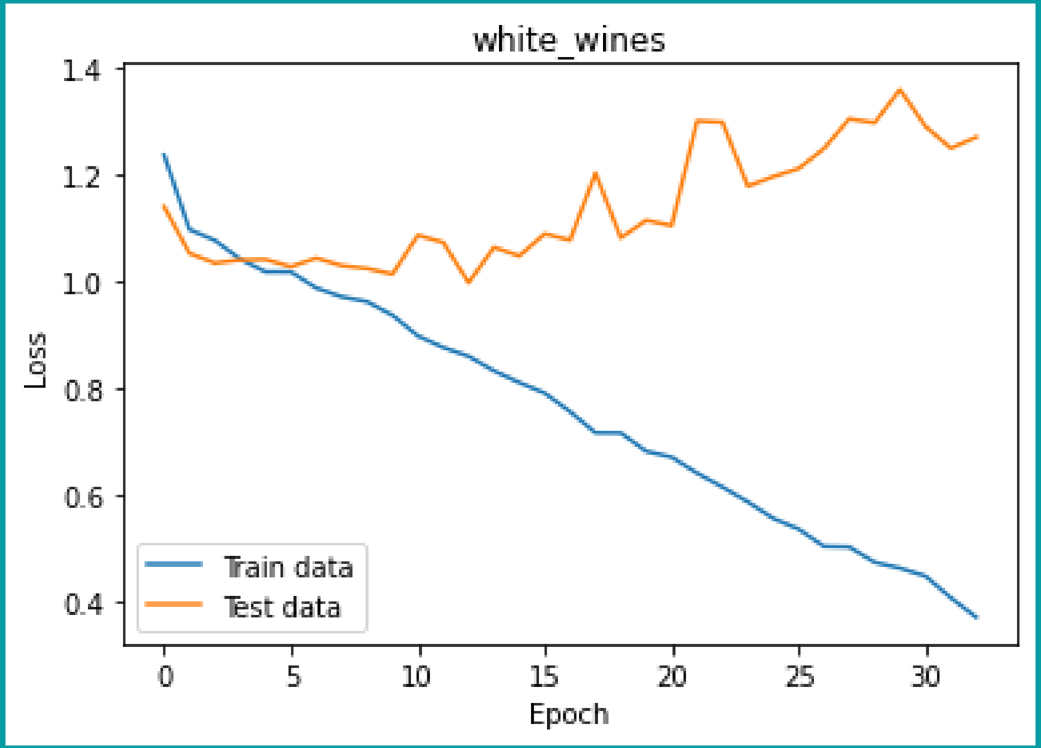


Metrik: Accuracy

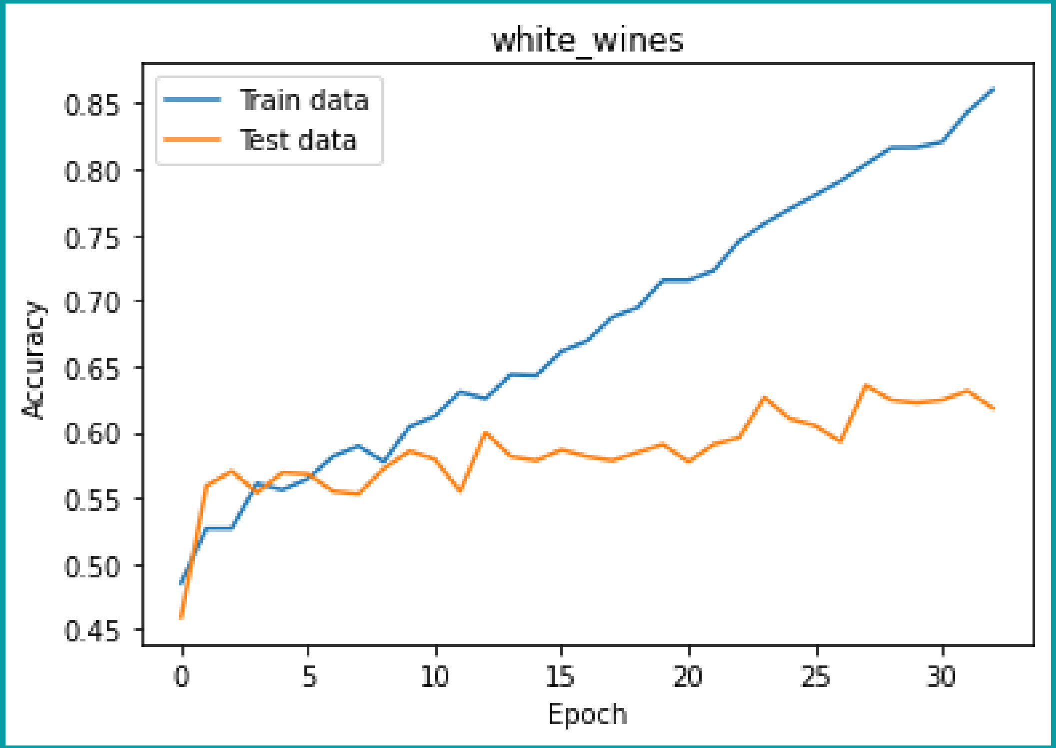


Klassifikation · Ergebnisse Training · Weißweine

Verlustfunktion: CategoricalCrossentropy



Metrik: Accuracy



Klassifikation · Ergebnisse Training · Rotweine

Diverse Accuracy-Metriken

```
In [120]: accuracy = model.evaluate(X_test, y_test)
10/10 [==============================] - 0s 2ms/step - loss: 1.0427 - accuracy: 0.5938

In [121]: y_pred = model.predict(X_test)
     ...:
     ...: cat_accuracy = keras.metrics.CategoricalAccuracy()
     ...: cat_accuracy.update_state(y_test, y_pred)
     ...: cat_accuracy.result().numpy()
10/10 [==============================] - 0s 2ms/step
Out[121]: 0.59375

In [122]: y_test_ord = np.argmax(y_test, axis=-1)
     ...: y_pred_ord = np.argmax(y_pred, axis=-1)
     ...:
     ...: accuracy = keras.metrics.Accuracy()
     ...: accuracy.update_state(y_test_ord, y_pred_ord)
     ...: accuracy.result().numpy()
Out[122]: 0.59375

In [123]: accuracy_T(y_test_ord, y_pred_ord, 0.5)
Out[123]: 0.59375

In [124]: accuracy_T(y_test_ord, y_pred_ord, 1)
Out[124]: 0.959375

In [125]: top2_accuracy = keras.metrics.TopKCategoricalAccuracy(k=2)
     ...: top2_accuracy.update_state(y_test, y_pred)
     ...: top2_accuracy.result().numpy()
Out[125]: 0.828125
```

Klassifikation · Ergebnisse Training · Weißweine

Diverse Accuracy-Metriken

```
In [100]: accuracy = model.evaluate(X_test, y_test)
31/31 [==============================] - 0s 2ms/step - loss: 1.2713 - accuracy: 0.6184

In [101]: cat_accuracy = keras.metrics.CategoricalAccuracy()
     ...: cat_accuracy.update_state(y_test, y_pred)
     ...: cat_accuracy.result().numpy()
Out[101]: 0.6183674

In [102]: accuracy = model.evaluate(X_test, y_test)
31/31 [==============================] - 0s 2ms/step - loss: 1.2713 - accuracy: 0.6184

In [103]: y_pred = model.predict(X_test)
     ...:
     ...: cat_accuracy = keras.metrics.CategoricalAccuracy()
     ...: cat_accuracy.update_state(y_test, y_pred)
     ...: cat_accuracy.result().numpy()
31/31 [==============================] - 0s 2ms/step
Out[103]: 0.6183674

In [104]: y_test_ord = np.argmax(y_test, axis=-1)
     ...: y_pred_ord = np.argmax(y_pred, axis=-1)
     ...:
     ...: accuracy = keras.metrics.Accuracy()
     ...: accuracy.update_state(y_test_ord, y_pred_ord)
     ...: accuracy.result().numpy()
Out[104]: 0.6183674
```

```
In [105]:
     ...: def accuracy_T(u, v, tolerance=0.5):
     ...:     hit = np.vectorize(lambda x: 1 if x <= tolerance else 0)
     ...:     return (hit(abs(u-v)).sum() / len(u))

In [106]: accuracy_T(y_test_ord, y_pred_ord, 0.5)
Out[106]: 0.6183673469387755

In [107]: accuracy_T(y_test_ord, y_pred_ord, 1)
Out[107]: 0.9418367346938775

In [108]: top2_accuracy = keras.metrics.TopKCategoricalAccuracy(k=2)
     ...: top2_accuracy.update_state(y_test, y_pred)
     ...: top2_accuracy.result().numpy()
Out[108]: 0.89489794
```

Vergleich Ergebnisse

	Red wine			White wine		
Metrik	Paper - Regr.	Projekt - Regr.	Projekt - Klass.	Paper - Regr.	Projekt - Regr.	Projekt - Klass.
MAE	0.51	0.43		0.58	0.64	
$\text{Accuracy}_{T=0.5}$	59.1	67.2	59.4	52.6	54.1	61.8
$\text{Accuracy}_{T=1.0}$	88.8	96.9	95.9	84.7	95.8	94.2

Referenzen

- Cortez, P., Teixeira, J., Cerdeira, A., Almeida, F., Matos, T., Reis, J. (2009). **Using Data Mining for Wine Quality Assessment.** In: Discovery Science. DS 2009. Lecture Notes in Computer Science, vol 5808, Springer

- P. Cortez, A. Cerdeira, F. Almeida, T. Matos and J. Reis. **Modeling wine preferences by data mining from physicochemical properties.** In: Decision Support Systems, Elsevier