Hochschule für Wirtschafts-, Rechts- und Sozialwissenschaften, Internationale Beziehungen und Informatik

Coding Project – Building a web application based on an OLS-regression-model

Sandro Gössi

Marco Müller

Noa Portmann

Norbert Sos

University of St. Gallen

Programming – Introduction Level

Mario Silic

25.05.2022

Spring Semester 2022

1 General project description

The group project implemented a code creating an interactive web application underlying a data analysis and a regression model training. The model returns predictions regarding the quality of red and white wine.

1.1 How to use the application?

The usage of the web application is very simple and consists of following three steps:

1. **Open the following link:**
   https://share.streamlit.io/pyt2hon/web_application_wine_data/main/WebApplication.py
2. **Enter your wine data**
3. **Receive your own predicted quality from 0 to 10 (0 = very bad, 10 = very good)**

1.2 Target

This group project defined an objective consisting of providing wine producers the opportunity to optimize their wine ingredients specific to what type of wine they are producing. To accomplish that goal this project uses a linear regression model called ordinary least squares (OLS), which will be mentioned later on. Additionally, a web application has been developed to let users determine the quality of a wine by varying certain parameters.

1.3 Resources

The inputs used by the project consist of two datasets which are available on kaggle.com with following two links.

White wine:

https://www.kaggle.com/datasets/piyushagni5/white-wine-quality

Red wine:

https://www.kaggle.com/datasets/uciml/red-wine-quality-cortez-et-al-2009

The datasets contain physiochemical and sensory variables about white and red variants of a Portuguese wine. The dataset of red wine contains 1599 entries, while the white wine dataset contains 4898 entries.

For the codes for the web application and the underlying analysis you can use the following link to the GitHub repository.

GitHub Repository:

https://github.com/Pyt2hon/Web_application_Wine_data/blob/main/Wine_Regression.ipynb

Besides this document, a README.md and a requirement.txt you will find the Jupyter Notebook called Wine_Regression.ipynb containing all the important information underlying our regression model. You will also find a file called WebApplication.py, which was used for the web application. The project is fully written in Python with the help of the Jupyter Notebook and the simple web application programmed in PyCharm.

2 Code description

This chapter has the purpose of briefly explaining the code behind the regression model to understand its origin.

Code part 1:

At first, we have chosen two wine datasets from Kaggle that are used throughout our analysis. Important libraries such as pandas, matplotlib, statsmodels, numpy, seaborn and sklearn, that are used in the project, are imported.

```
import pandas as pd
import matplotlib.pyplot as plt
import statsmodels.api as sm
import numpy as np
import seaborn as sns
from sklearn.model_selection import train_test_split
```

The above-mentioned two datasets are reading into a Pandas data frame from Google Drive.

```
# Reading in both datasets into a Pandas DataFrame

# Reading Red Wine Dataset from Google Drive
red_wine = pd.read_csv("https://drive.google.com/uc?export=download&id=1eRcH9IRsAMzmRCNIKm4lePIVslnhxUSQ")

# Reading White Wine Dataset from Google Drive
white_wine = pd.read_csv("https://drive.google.com/uc?export=download&id=1htdZLgJU6CbQfnSuGvJOOHZUukE9_aJh")
```

The two datasets are inspected and it is shown that the red wine dataset contains 12 features and 1599 rows, while the white wine dataset consists of 12 features and 4898 rows.

Code part 2:

After reading in the dataset and inspecting the dataset we evaluate the dataset on the variables, the missing values and their datatype with the code .info().

```
<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
<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
```

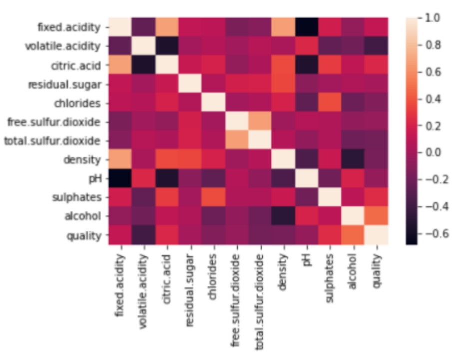
We can see that both datasets contain the same features, no missing values and mostly floats.

Code part 3:

The correlation between features is depicted on a heatmap from dark (lowest correlation coefficient) to light (highest correlation coefficient). High correlations would indicate problems with our prediction models.

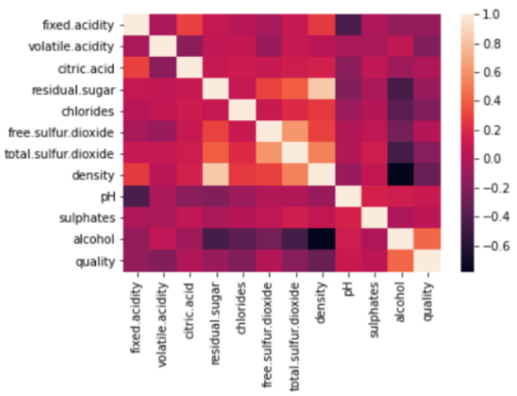
```
##Inspect the correlation of the features with a correlation heatmap

#Correlation heatmap of the red wine dataset
sns.heatmap(red_wine.corr())
```

```
<AxesSubplot:>
```




```
#Correlation heatmap of the white wine dataset
sns.heatmap(white_wine.corr())
```

```
<AxesSubplot:>
```



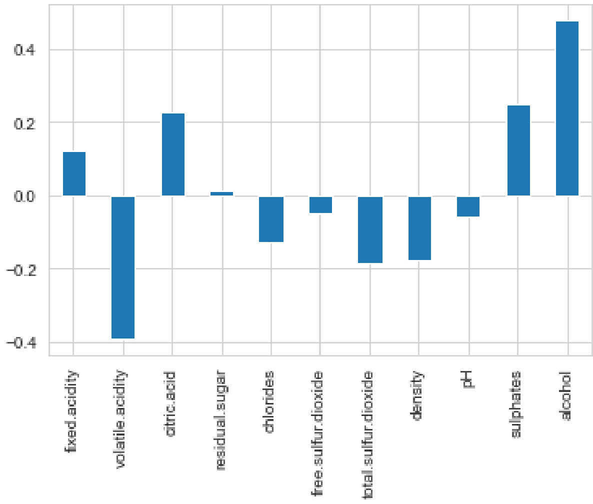
Each feature is analyzed further with different scatterplots. The scatterplots show how wine quality relates to each feature.

Code part 4:

Through plotting with the following code, the most important factors considering the quality can be visualized:

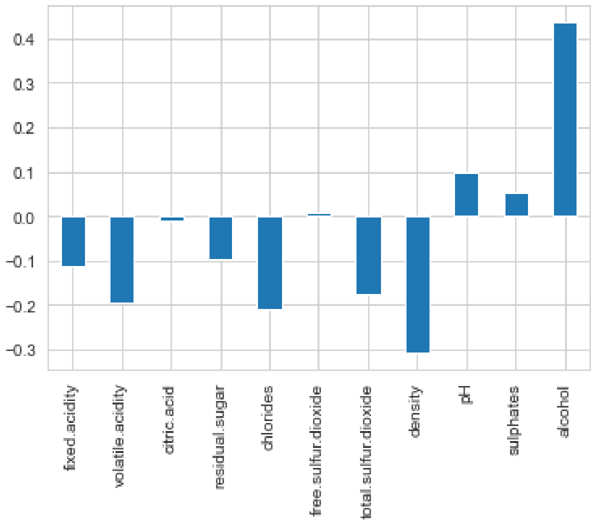
```
#Correlation coefficients of the red wine data set
red_wine_correlations = red_wine.corr()['quality'].drop('quality')
print(red_wine_correlations)

#Barplot of the correlation coefficients
red_wine_correlations.plot(kind='bar')
```

The barplot shows that the most important parameters for the red wine are, not surprising, alcohol with a positive correlation coefficient of almost 0.5 and the volatile acidity with a moderate negative correlation coefficient of almost -0.4.




```
#Correlation coefficients of the white wine data set
white_wine_correlations = white_wine.corr()['quality'].drop('quality')
print(white_wine_correlations)

#Barplot of the correlation coefficients
white_wine_correlations.plot(kind='bar')
```

The barplot for the white wine correlation shows that alcohol is again the most dominant factor in the model with a positive correlation coefficient of 0.44. The density follows with a negative correlation coefficient of -0.3.



Code part 5:

After having analyzed the correlations of the columns with the target variable it is important to prevent multicollinearity of the independent variables. To do so the code uses a heatmap. The correlation between each feature is depicted with a colored heatmap. Dark red showing strong positive correlation and dark blue implying strong negative correlation.

```
##Inspect the correlations even further with a collored heatmap table incl. correlation coeffiecients

# Correlation table for red wines
red_wine_corr = red_wine.corr()
red_wine_corr.style.background_gradient(cmap='coolwarm')
```

	fixed.acidity	volatile.acidity	citric.acid	residual.sugar	chlorides	free.sulfur.dioxide	total.sulfur.dioxide	density	pH	sulphates	alcohol	quality
fixed.acidity	1.000000	-0.256131	0.671703	0.114777	0.093705	-0.153794	-0.113181	0.668047	-0.682978	0.183006	-0.061668	0.124052
volatile.acidity	-0.256131	1.000000	-0.552496	0.001918	0.061298	-0.010504	0.076470	0.022026	0.234937	-0.260987	-0.202288	-0.390558
citric.acid	0.671703	-0.552496	1.000000	0.143577	0.203823	-0.060978	0.035533	0.364947	-0.541904	0.312770	0.109903	0.226373
residual.sugar	0.114777	0.001918	0.143577	1.000000	0.055610	0.187049	0.203028	0.355283	-0.085652	0.005527	0.042075	0.013732
chlorides	0.093705	0.061298	0.203823	0.055610	1.000000	0.005562	0.047400	0.200632	-0.265026	0.371260	-0.221141	-0.128907
free.sulfur.dioxide	-0.153794	-0.010504	-0.060978	0.187049	0.005562	1.000000	0.667666	-0.021946	0.070377	0.051658	-0.069408	-0.050656
total.sulfur.dioxide	-0.113181	0.076470	0.035533	0.203028	0.047400	0.667666	1.000000	0.071269	-0.066495	0.042947	-0.205654	-0.185100
density	0.668047	0.022026	0.364947	0.355283	0.200632	-0.021946	0.071269	1.000000	-0.341699	0.148506	-0.496180	-0.174919
pH	-0.682978	0.234937	-0.541904	-0.085652	-0.265026	0.070377	-0.066495	-0.341699	1.000000	-0.196648	0.205633	-0.057731
sulphates	0.183006	-0.260987	0.312770	0.005527	0.371260	0.051658	0.042947	0.148506	-0.196648	1.000000	0.093595	0.251397
alcohol	-0.061668	-0.202288	0.109903	0.042075	-0.221141	-0.069408	-0.205654	-0.496180	0.205633	0.093595	1.000000	0.476166
quality	0.124052	-0.390558	0.226373	0.013732	-0.128907	-0.050656	-0.185100	-0.174919	-0.057731	0.251397	0.476166	1.000000

```
# Correlation table for red wines
white_wine_corr = white_wine.corr()
white_wine_corr.style.background_gradient(cmap='coolwarm')
```

	fixed.acidity	volatile.acidity	citric.acid	residual.sugar	chlorides	free.sulfur.dioxide	total.sulfur.dioxide	density	pH	sulphates	alcohol	quality
fixed.acidity	1.000000	-0.022697	0.289181	0.089021	0.023086	-0.049396	0.091070	0.265331	-0.425858	-0.017143	-0.120881	-0.113663
volatile.acidity	-0.022697	1.000000	-0.149472	0.064286	0.070512	-0.097012	0.089261	0.027114	-0.031915	-0.035728	0.067718	-0.194723
citric.acid	0.289181	-0.149472	1.000000	0.094212	0.114364	0.094077	0.121131	0.149503	-0.163748	0.062331	-0.075729	-0.009209
residual.sugar	0.089021	0.064286	0.094212	1.000000	0.088685	0.299098	0.401439	0.838966	-0.194133	-0.026664	-0.450631	-0.097577
chlorides	0.023086	0.070512	0.114364	0.088685	1.000000	0.101392	0.198910	0.257211	-0.090439	0.016763	-0.360189	-0.209934
free.sulfur.dioxide	-0.049396	-0.097012	0.094077	0.299098	0.101392	1.000000	0.615501	0.294210	-0.000618	0.059217	-0.250104	0.008158
total.sulfur.dioxide	0.091070	0.089261	0.121131	0.401439	0.198910	0.615501	1.000000	0.529881	0.002321	0.134562	-0.448892	-0.174737
density	0.265331	0.027114	0.149503	0.838966	0.257211	0.294210	0.529881	1.000000	-0.093591	0.074493	-0.780138	-0.307123
pH	-0.425858	-0.031915	-0.163748	-0.194133	-0.090439	-0.000618	0.002321	-0.093591	1.000000	0.155951	0.121432	0.099427
sulphates	-0.017143	-0.035728	0.062331	-0.026664	0.016763	0.059217	0.134562	0.074493	0.155951	1.000000	-0.017433	0.053678
alcohol	-0.120881	0.067718	-0.075729	-0.450631	-0.360189	-0.250104	-0.448892	-0.780138	0.121432	-0.017433	1.000000	0.435575
quality	-0.113663	-0.194723	-0.009209	-0.097577	-0.209934	0.008158	-0.174737	-0.307123	0.099427	0.053678	0.435575	1.000000

The datasets are prepared for the multiple linear regression by dropping the multicollinear variables to prevent interference later. With 0.84 “Density” and “residual_sugar” show too high corelation coefficients. As it’s easier for the user to enter the amount of residual sugar than to measure the density, we drop the density. Furthermore, we drop the variable “pH” because it is dependent of the variables volatile and fixed acidity.

Code part 6:

After using the train_test_split()-function with a random_state of 5 for reproducibility and a test-size of 30%, to prevent a hardly trained model or overfitting, the following code can perform an OLS-regression on the training data. By default, statsmodel fits a line passing through the origin, i.e. it does not fit an intercept. To find the intercept a constant is added. In the case of the white wine dataset the constant is 2.58. The OLS regression model results show that only about 27% of the variance of the quality can be explained by the model. This is shown by the R-squared value.

```
##Training the Multiple Linear Regression Model on the White Wine Train dataset.

#By default, statsmodels fits a line passing through the origin, i.e. it doesn't fit an intercept.
# Adding a constant, so that it also fits an intercept.
X_train = sm.add_constant(X_train, prepend=False)

# Training and fitting the multiple linear regression model
model  = sm.OLS(y_train, X_train)
model1 = model.fit()

print(model1.summary()) # Only about 27% of the variance of the quality can be explained by the model
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                quality   R-squared:                       0.274
Model:                            OLS   Adj. R-squared:                  0.272
Method:                 Least Squares   F-statistic:                     143.4
Date:                Sun, 22 May 2022   Prob (F-statistic):          3.47e-230
Time:                        18:37:12   Log-Likelihood:                -3961.4
No. Observations:                3428   AIC:                             7943.
Df Residuals:                    3418   BIC:                             8004.
Df Model:                           9
Covariance Type:            nonrobust
========================================================================================
                           coef    std err          t      P>|t|      [0.025      0.975]
----------------------------------------------------------------------------------------
fixed.acidity           -0.0602      0.017     -3.591      0.000      -0.093      -0.027
volatile.acidity        -1.9103      0.136    -14.026      0.000      -2.177      -1.643
citric.acid             -0.0126      0.117     -0.108      0.914      -0.242       0.217
residual.sugar           0.0238      0.003      7.786      0.000       0.018       0.030
chlorides               -1.7935      0.648     -2.769      0.006      -3.063      -0.524
free.sulfur.dioxide      0.0047      0.001      4.562      0.000       0.003       0.007
total.sulfur.dioxide    -0.0005      0.000     -1.182      0.237      -0.001       0.000
sulphates                0.4161      0.118      3.535      0.000       0.185       0.647
alcohol                  0.3703      0.014     27.084      0.000       0.344       0.397
const                    2.5787      0.220     11.736      0.000       2.148       3.010
==============================================================================
Omnibus:                       84.882   Durbin-Watson:                   1.991
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              186.506
Skew:                           0.093   Prob(JB):                     3.17e-41
Kurtosis:                       4.128   Cond. No.                     7.44e+03
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 7.44e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
```

Code part 7:

The same code structure is used to train the model for the red wine dataset. In this case the constant is 2.82. The R-squared value, which describes the amount of the variance of the quality which can be described by the independent variables, is 0.35. This implies that in this regard the red wine model is slightly performing better than the model for white wine.

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                quality   R-squared:                       0.349
Model:                            OLS   Adj. R-squared:                  0.344
Method:                 Least Squares   F-statistic:                     66.08
Date:                Sun, 22 May 2022   Prob (F-statistic):           3.56e-97
Time:                        18:37:20   Log-Likelihood:                -1102.1
No. Observations:                1119   AIC:                             2224.
Df Residuals:                    1109   BIC:                             2274.
Df Model:                           9
Covariance Type:            nonrobust
========================================================================================
                           coef    std err          t      P>|t|      [0.025      0.975]
----------------------------------------------------------------------------------------
fixed.acidity            0.0364      0.016      2.298      0.022       0.005       0.068
volatile.acidity        -1.1789      0.144     -8.186      0.000      -1.461      -0.896
citric.acid             -0.0910      0.174     -0.524      0.601      -0.432       0.250
residual.sugar           0.0053      0.015      0.351      0.725      -0.024       0.035
chlorides               -1.7123      0.469     -3.654      0.000      -2.632      -0.793
free.sulfur.dioxide      0.0028      0.003      1.061      0.289      -0.002       0.008
total.sulfur.dioxide    -0.0026      0.001     -2.940      0.003      -0.004      -0.001
sulphates                0.8231      0.129      6.372      0.000       0.570       1.077
alcohol                  0.2695      0.021     13.015      0.000       0.229       0.310
const                    2.8245      0.285      9.917      0.000       2.266       3.383
==============================================================================
Omnibus:                       19.564   Durbin-Watson:                   2.000
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               26.946
Skew:                          -0.192   Prob(JB):                     1.41e-06
Kurtosis:                       3.656   Cond. No.                     1.48e+03
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.48e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
```

Code part 8:

The prepared model can now be used to perform predictions on both datasets. A good measure to evaluate the strength of the model is the root mean squared error (RMSE). It tells us about the average deviation the quality prediction values have from the true quality values.

```
# Adding a constant to fit the size
X_test = sm.add_constant(X_test, prepend=False)

# Making Predictions of the wine quality of the test data
predictions_white_wine = model1.predict(X_test)

# Calculating the root mean squared error to evaluate it's performance
RMSE = np.sqrt(np.mean((predictions_white_wine - y_test)**2))

RMSE
```

```
0.7257855549422482
```

Regarding the white wine test data, with 0.726 the mean difference between the model and the real values is not even 1 quality point, which is pretty good. Although there still is some upside-potential.

Now we can use the same methods on the red wine test data.

```
# Adding a constant to fit the size
X_test = sm.add_constant(X_test, prepend=False)

# Making Predictions of the wine quality of the test dataset
predictions_red_wine = model2.predict(X_test)

# Calculating the root mean squared error to evaluate it's performance
RMSE = np.sqrt(np.mean((predictions_red_wine - y_test)**2))

RMSE
```

```
0.6500780804966902
```

The Multiple Linear Regression Model on the red wine test dataset is evaluated by calculating the mean squared error. The result is 0.65 which is a slightly better result. This comes little bit surprising as it is the smaller dataset.

Code part 9:

Finally, we analyse, which wine of the 6495 wines contain the highest alcohol content. It turns out that 14.9% is the highest content. Its index is 5552, a red wine.

```
#Now we finally get to the important questions that students are most interested in:

#Which wine of the 6495 wines in the two datasets has the highest alcohol content?

all_wines = pd.merge(white_wine, red_wine, on = ['fixed.acidity', 'volatile.acidity', 'citric.acid', 'residual.sugar',
       'chlorides', 'free.sulfur.dioxide', 'total.sulfur.dioxide', 'density',
       'pH', 'sulphates', 'alcohol', 'quality'], how = 'outer')

print(all_wines["alcohol"].max())

#Is it a red wine or a white wine?
print(white_wine["alcohol"].max())
print(red_wine["alcohol"].max()) #It's a red wine!!!

#The wine with the highest alcohol content is... "drum roll":
winner = all_wines.loc[all_wines['alcohol'] == 14.9]
winner
```

```
14.9
14.2
14.9
```

	fixed.acidity	volatile.acidity	citric.acid	residual.sugar	chlorides	free.sulfur.dioxide	total.sulfur.dioxide	density	pH	sulphates	alcohol	quality
5552	15.9	0.36	0.65	7.5	0.096	22.0	71.0	0.9976	2.98	0.84	14.9	5

3 Web application

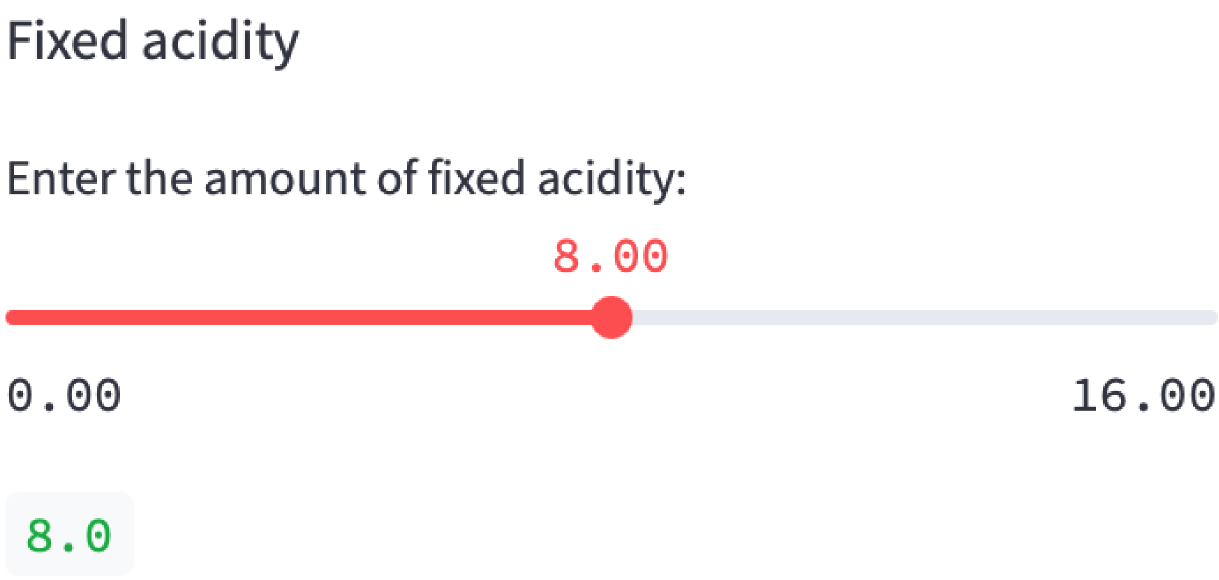
The web application was programmed in PyCharm and was primarily based on Streamlit. This on the grounds that Streamlit is open-source, simple and is suitable for data science. Additionally, there is no extra requirements for the user to open the application via link.

Code part 10:

Below an example from the code underlying the web application and its output are shown.

```
# Displays a slider widget for the user to enter float values from 0 to 16 for the fixed acidity
Fixed_acidity = row1_col2.slider(
    "Enter the amount of fixed acidity:",
    min_value=0.0,
    max_value=16.0,
    value=8.0)  # Sets a default value of 8
row1_col2.write(Fixed_acidity)  # Shows the user the entered input
```

Output in the web application:



With the help of streamlit (st) functions like .radio() for radio button widgets, .slider for slider widgets and .write() or .subheader() for displaying values the code was able to receive input values on the web application and display them. Eventually the addition of the values and their weights according to the OLS regression (see code part 10) model made it possible to make a prediction.

Code part 11:

On the basis of the OLS regression model, it is possible to implement a web application with a prediction feature. For the prediction the evaluated parameters can be used individually for each wine type as can be seen below.

```
model1.params #White Wine

fixed.acidity           -0.060157
volatile.acidity        -1.910340
citric.acid             -0.012632
residual.sugar           0.023824
chlorides               -1.793469
free.sulfur.dioxide      0.004697
total.sulfur.dioxide    -0.000531
sulphates                0.416117
alcohol                  0.370318
const                    2.578745
dtype: float64
```

```
model2.params #Red Wine

fixed.acidity            0.036430
volatile.acidity        -1.178876
citric.acid             -0.090977
residual.sugar           0.005276
chlorides               -1.712330
free.sulfur.dioxide      0.002817
total.sulfur.dioxide    -0.002601
sulphates                0.823083
alcohol                  0.269529
const                    2.824479
dtype: float64
```

To return a prediction the entered inputs were simply multiplied by their parameter values depending on their wine type, the calculated constant was added and then all values were summed up.

Code part 12:

The last part of the project was to display the predicted wine quality. The code uses a simple conditional statement. If the entered value for wine type is “Red wine” the specific value is returned with st.subheader(). The same proceeding is applied for the white wine.

```
# Returns a rounded quality-value for red wine if the entered wine-type is red
if Wine_type == "Red wine":
    st.subheader(f"Your red wine's quality is: {round(Red_quality,2)}")

# Returns a rounded quality-value for white wine if the entered wine-type is white
elif Wine_type == "White wine":
    st.subheader(f"Your white wine's quality is: {round(White_quality,2)}")
```

4 Limitations and summary

While dealing with this project many things were learned, but some flaws regarding the model revealed themselves. One of the most essential limitations of this project lies in the accuracy of the model. On the one hand, only about 27% respectively 35% of the variance of the quality can be explained by the other variables. This can be seen as a result of other crucial factors like the combination of the parameters or even the preferences of the customers. On the other hand, the regression model only uses linear dependencies in its evaluation, not taking polynomial dependencies into consideration. Furthermore, it is implausible to think that a producer can vary all these parameters freely. In reality he is bound by the grape variety and external influences like weather and temperature.

Despite those flaws the model helps to simulate the effect of specific parameters in order to predict the quality for white and red wine. Therefore, it can be said that the web application may have a certain customer value, which could obviously be optimized in the future, but that would have gone beyond the scope of this project.