VinoMetrics

James Kenny and Matthew Batchelor

Mission: Build a model that predicts the quality of wine

Data set: Wine Quality from UCI Machine Learning Repository; based on Vinho Verde wine from Northern Portugal

Description of data

WINE DATAFRAME	Input variable	Explained variable
Count	11	1
Index	['fixed acidity', 'volatile acidity', 'citric acid', 'residual sugar', 'chlorides', 'free sulfur dioxide', 'total sulfur dioxide', 'density','pH', 'sulphates', 'alcohol']	[‘quality’]
Dtype	float	int
Null values	0	
Instances red (of which duplicates)	1599 (240 = 15%)	
Instances white (of which duplicates)	4898 (937 = 19%)	

Optional: Variable overview

Input variables - effect **-** controlled by producer ✓

'fixed acidity' - more or less robust
'volatile acidity' - tanginess or sharpness ✓
'citric acid' - notes of lemon or lime ✓
'residual sugar' - sweetness ✓
'chlorides' - saltiness ✓
'free sulfur dioxide' - preservative ✓
'total sulfur dioxide' - stability and longevity ✓
'density' - richness (high = ‘full-bodied) ✓
'pH' - acidity ✓
'sulphates' - maintains freshness ✓
'alcohol' - strength (high = warm, full-bodied) ✓

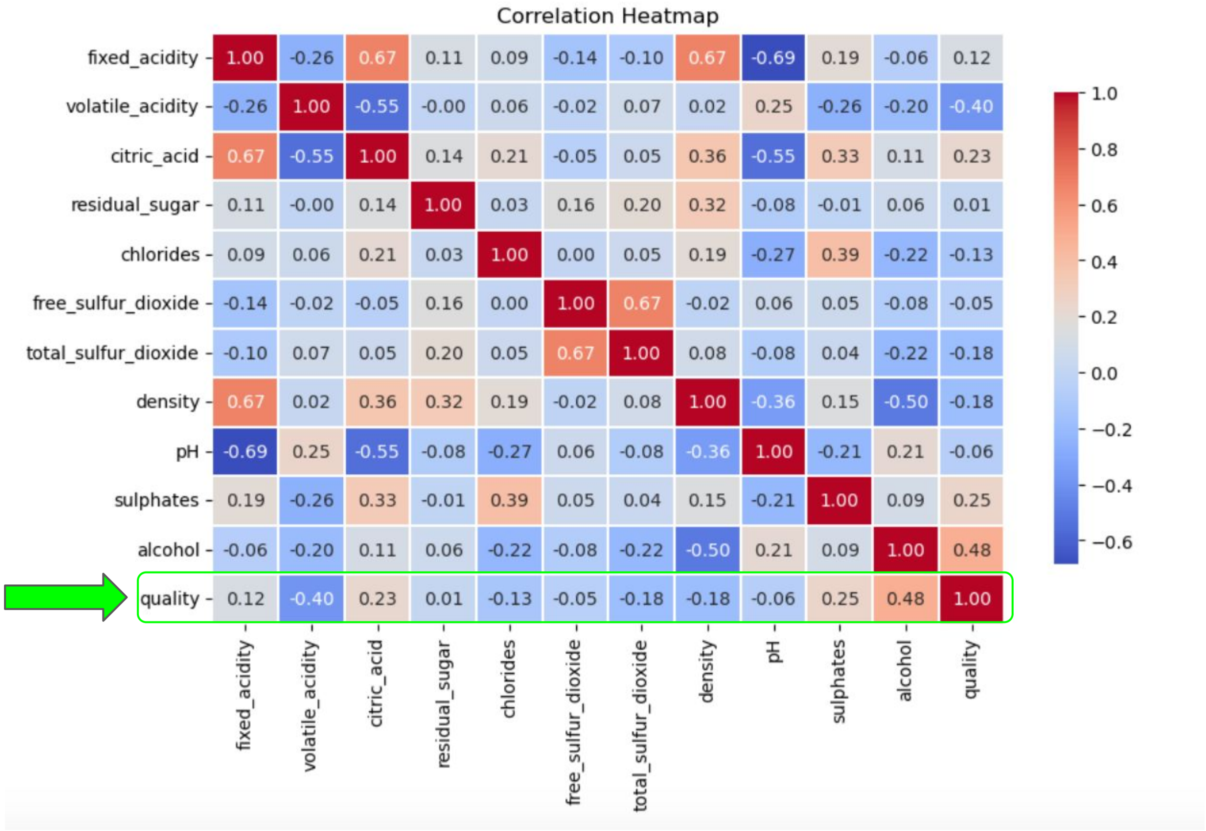
Output variable

‘quality’

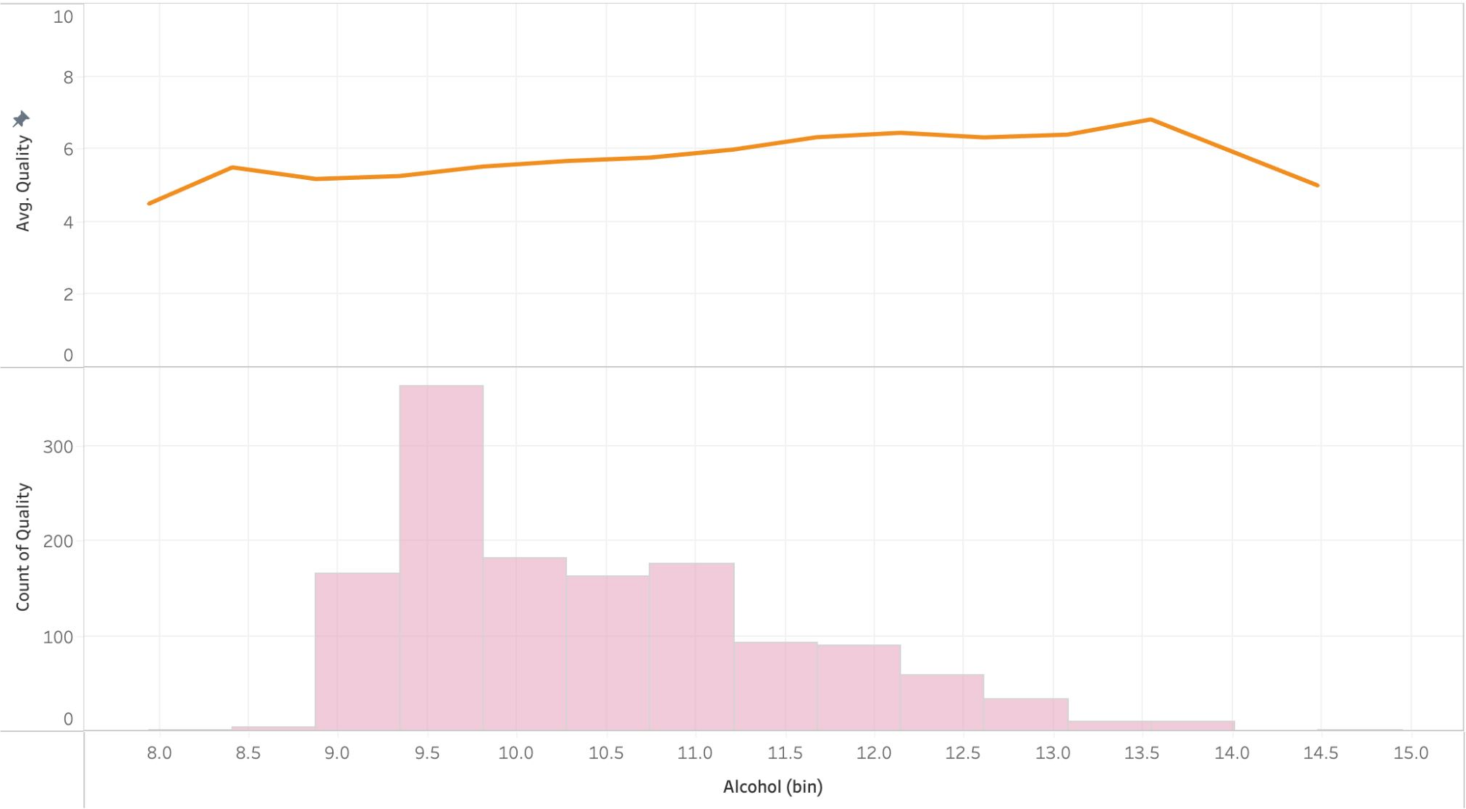
Score: 0-10

Mean average of mean average of 3 wine tastings by wine experts

Variable - correlation heatmap (red wine)



Optional: Strongest correlated variable combination (alcohol x quality) – example of ‘piecewise linear regression’



Linear regression - before optimization

Linear Regression Model

In [9]:
```
# Linear regression
lm = LinearRegression()
model = lm.fit(X_train, y_train)
print(f'model coefficients:\n {model.coef_}\n')
print(f'model intercept:\n {model.intercept_}\n')

# Applying model to X test
y_pred = model.predict(X_test)
#y_pred = pd.DataFrame(scaler_s_y.inverse_transform(y_pred)) # inversing y
y_pred = pd.DataFrame(y_pred)

y_pred = y_pred.rename(columns = {0:"y_pred"})

y_test = y_test.reset_index(drop=True)
y_test = y_test.rename(columns = {"quality":"y_test"})

residuals_df = pd.concat([y_test,y_pred], axis = 1)
residuals_df["residual"] = residuals_df["y_test"] - residuals_df["y_pred"]


# Root mean squared error
rmse = mse(y_test, residuals_df["y_pred"], squared=False)
print(f'Root mean squared error: {rmse} \n')

# R^2
r2 = r2_score(y_test, residuals_df["y_pred"])
print(f'R2: {r2} \n')

# Calculating adjusted R^2
n = X_train.shape[0]  # Number of observations in the training set
p = X_train.shape[1]  # Number of features used for training
adjusted_r2 = 1 - (1 - r2) * (n - 1) / (n - p - 1)
print(f'Adjusted R2: {adjusted_r2} \n')

residuals_df
```

```
model coefficients:
 [[-4.14972197e-02 -9.94118538e-01 -2.30359318e-02 -2.34947735e-03
  -2.04381921e+00  5.60218000e-03 -3.96020465e-03  2.31453631e+01
  -8.73151979e-01  9.07091584e-01  3.04300621e-01]]

model intercept:
 [-17.15994201]

Root mean squared error: 0.6377068568203348

R2: 0.39830469246210143

Adjusted R2: 0.3917320525585096
```

OLS Regression Results - before model optimization

In [16]:
```
X_train = sm.add_constant(X_train)
model = sm.OLS(y_train,X_train).fit()
model.summary()
```

Out[16]:

OLS Regression Results

Dep. Variable:	quality	**R-squared:**	0.347
Model:	OLS	**Adj. R-squared:**	0.340
Method:	Least Squares	**F-statistic:**	48.71
Date:	Thu, 08 Feb 2024	**Prob (F-statistic):**	1.27e-85
Time:	11:29:31	**Log-Likelihood:**	-1030.6
No. Observations:	1019	**AIC:**	2085.
Df Residuals:	1007	**BIC:**	2144.
Df Model:	11		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-17.1599	27.713	-0.619	0.536	-71.542	37.222
fixed_acidity	-0.0415	0.035	-1.201	0.230	-0.109	0.026
volatile_acidity	-0.9941	0.157	-6.320	0.000	-1.303	-0.685
citric_acid	-0.0230	0.189	-0.122	0.903	-0.394	0.348
residual_sugar	-0.0023	0.020	-0.115	0.909	-0.042	0.038
chlorides	-2.0438	0.513	-3.982	0.000	-3.051	-1.037
free_sulfur_dioxide	0.0056	0.003	1.950	0.051	-3.53e-05	0.011
total_sulfur_dioxide	-0.0040	0.001	-4.148	0.000	-0.006	-0.002
density	23.1454	28.311	0.818	0.414	-32.410	78.700
pH	-0.8732	0.261	-3.341	0.001	-1.386	-0.360
sulphates	0.9071	0.148	6.115	0.000	0.616	1.198
alcohol	0.3043	0.034	8.840	0.000	0.237	0.372

Omnibus:	14.137	**Durbin-Watson:**	1.974
Prob(Omnibus):	0.001	**Jarque-Bera (JB):**	17.934
Skew:	-0.177	**Prob(JB):**	0.000128
Kurtosis:	3.545	**Cond. No.**	1.15e+05

Hypothesis testing

H0 : $\beta 1 = \beta 2 = \beta 3 = \beta 4 = \beta 5 = \beta 6 = \beta 7 = \beta 8 = \beta 9 = \beta 10 = \beta 11 = 0$

REJECTED

Null hypothesis: The coefficients of all input variables are equal to 0.
None of our wine properties predict wine quality.

H1 : $\beta 1 = \beta 2 = \beta 3 = \beta 4 = \beta 5 = \beta 6 = \beta 7 = \beta 8 = \beta 9 = \beta 10 = \beta 11 \mathrel{!=} 0$

Significant linear relationship between at least one of the 11 input variables and the explained variable.
At least one of our wine properties predicts wine quality.

R-squared:	0.347
Adj. R-squared:	0.340
F-statistic:	48.71
Prob (F-statistic):	1.27e-85

Linear regression - optimization

Optimization techniques applied:

- IQR method to address outliers (Applied)
  - **Action:** Converted outliers to 0.25/0.75 percentile
- Recursive Feature Elimination (RFE) to decrease redundancy (Tested, not applied)
  - **Action:** dropped 'free_sulfur_dioxide' after testing
- Scaler/Transformer iteration (Applied) – exhaustive trial and error approach to improve R2 and adjusted R2
  - **Action:** Standard Scaler on *X*, no scaler on *y*

Linear regression - impact

R2:

Before optimization: 0.347
After optimization: **0.409**

R2 adjusted:

Before optimization: 0.340
After optimization: **0.404**

Linear regression - impact

Print coefficients mapping?

Stated mission:

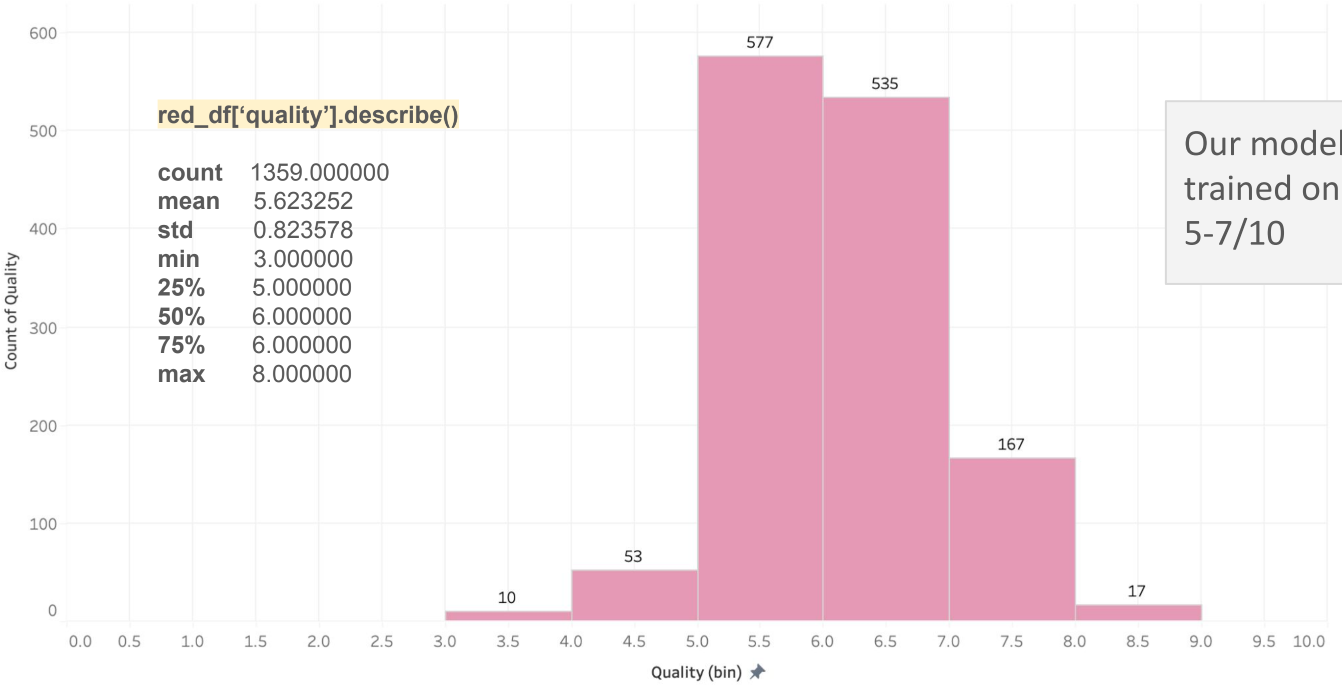
- Build a model that predicts the quality of wine

Outcome:

- 40% of the variability in wine quality can be accounted for by the input variables
- 60% of the variability in wine quality is unexplained by the input variables

Model evaluation

<Distribution of scores (red wine, all data minus duplicates)>



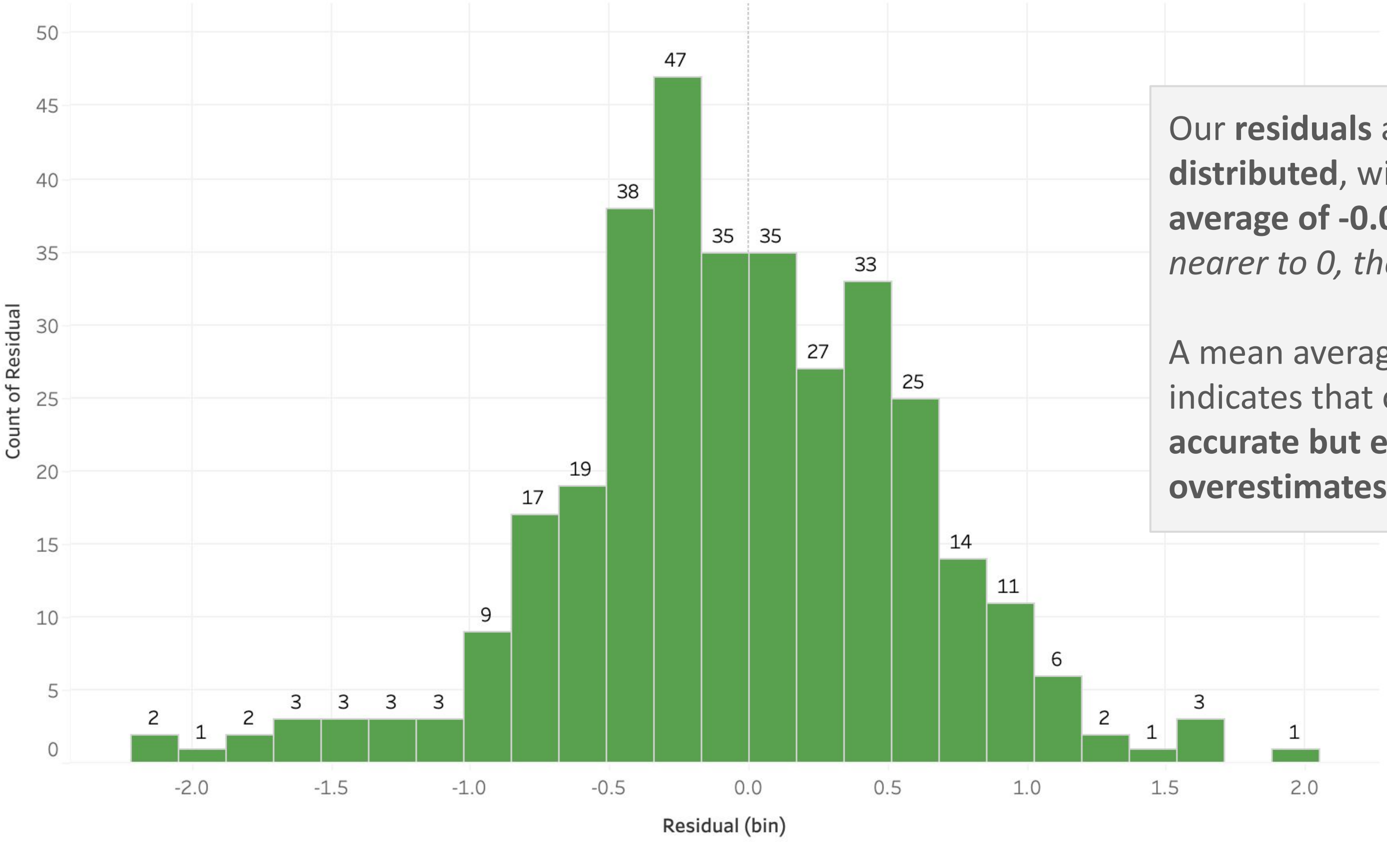
red_df['quality'].describe()

count	1359.000000
mean	5.623252
std	0.823578
min	3.000000
25%	5.000000
50%	6.000000
75%	6.000000
max	8.000000

Our model was predominantly trained on wines that scored 5-7/10

Model evaluation

<Wine Data: Red wine: Distribution of Residuals (test data, 340 instances)>

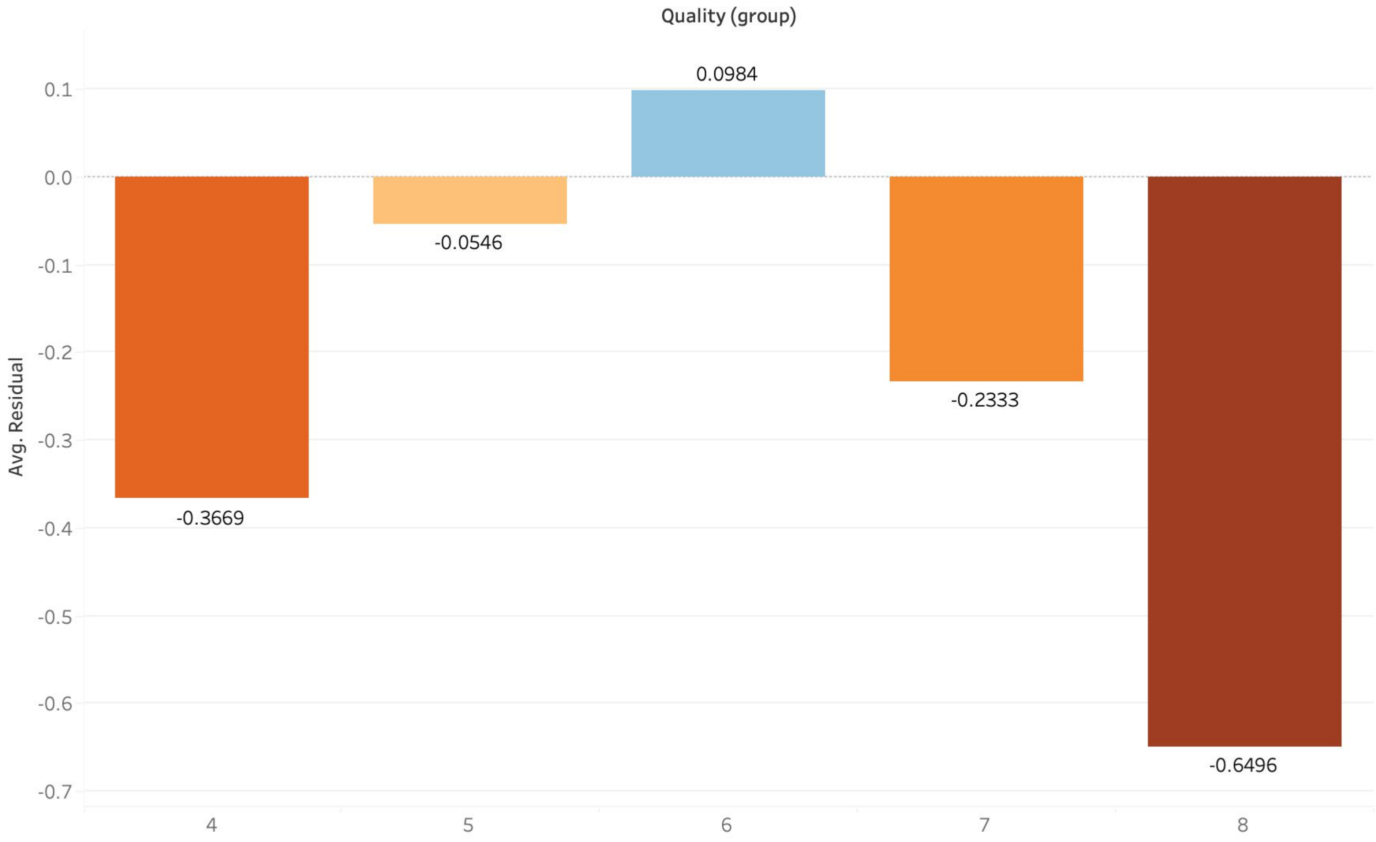


Our **residuals** are **normally distributed**, with a **mean average of -0.03** (*guide: the nearer to 0, the better*).

A mean average residual of -0.03 indicates that our model is **accurate but ever-so-slightly overestimates** wine quality.

Model evaluation

<Mean of residuals by quality group (red wine, test data, 340 instances>



Our model was predominantly trained on wines that scored 5-7/10

As a result, our model is a **much stronger predictor** (lower residuals) **for wines in the 5-7 quality groups**.

Constraints of model

- **data availability:** lack of accessible data to further train model; wine properties seem to be the 'secret sauce' and wine producers are not necessarily incentivised to publish this data

- **consumer value:** as this data is not printed on wine bottles, our model is of little value to consumers

- **colour sensitivity:** a model trained on red wine failed to predict white wine quality as effectively
    - **other possible sensitivities:** no data on effect of grape or location

- **R2:** 60% of variability in quality is unexplained by this model

Conclusion

- While flawed, our model has *some* degree of explanatory power in predicting wine quality. On a scale of 0-10 points, we can predict score to within 0.6 points of accuracy on average

- In a commercial setting, our model would fulfil a business need of enabling wine producers to increase their likelihood of producing high-quality wine

INTRODUCING THE MVP OF THE **VINOMETRICS WINE QUALITY PREDICTOR ©**