

Wine Quality

Guy Kahana & Anat Peled



Project IV: Classification Models in ML & Advanced techniques



The Dataset

- ◀ The dataset was taken from [Kaggle](#), but originally downloaded from the UCI Machine Learning Repository.
- ◀ The dataset refers to red and white variants of the Portuguese "Vinho Verde" wine. The reference [Cortez et al., 2009]. Due to privacy and logistic issues, only physicochemical (inputs) and sensory (the output) variables are available (e.g. there is no data about grape types, wine brand, wine selling price, etc.).
- ◀ The goal of our project is to predict the wine quality using its features
- ◀ This prediction may assist wine makers in the producing process

Acknowledgements: 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, 47(4):547-553, 2009.



The Dataset

- ◀ Few nulls are present
- ◀ 12 numerical feature, 1 object (wine type)

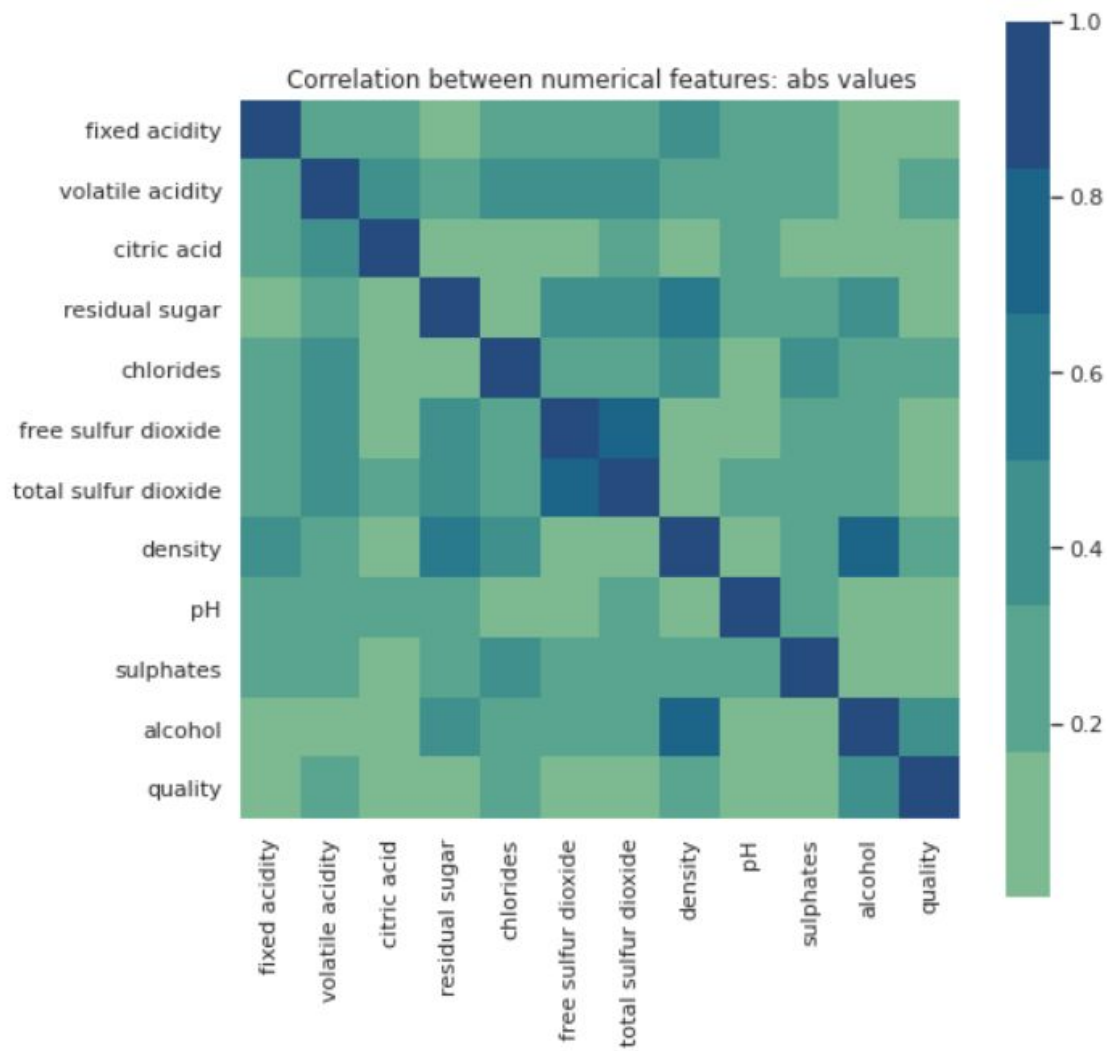
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6497 entries, 0 to 6496
Data columns (total 13 columns):
#   Column                      Non-Null Count  Dtype
---  -
0   type                        6497 non-null   object
1   fixed acidity               6487 non-null   float64
2   volatile acidity           6489 non-null   float64
3   citric acid                 6494 non-null   float64
4   residual sugar              6495 non-null   float64
5   chlorides                   6495 non-null   float64
6   free sulfur dioxide         6497 non-null   float64
7   total sulfur dioxide        6497 non-null   float64
8   density                     6497 non-null   float64
9   pH                          6488 non-null   float64
10  sulphates                   6493 non-null   float64
11  alcohol                     6497 non-null   float64
12  quality                     6497 non-null   int64
dtypes: float64(11), int64(1), object(1)
memory usage: 660.0+ KB
```

	type	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
0	white	7.0	0.27	0.36	20.7	0.045	45.0	170.0	1.0010	3.00	0.45	8.8	6
1	white	6.3	0.30	0.34	1.6	0.049	14.0	132.0	0.9940	3.30	0.49	9.5	6
2	white	8.1	0.28	0.40	6.9	0.050	30.0	97.0	0.9951	3.26	0.44	10.1	6
3	white	7.2	0.23	0.32	8.5	0.058	47.0	186.0	0.9956	3.19	0.40	9.9	6
4	white	7.2	0.23	0.32	8.5	0.058	47.0	186.0	0.9956	3.19	0.40	9.9	6



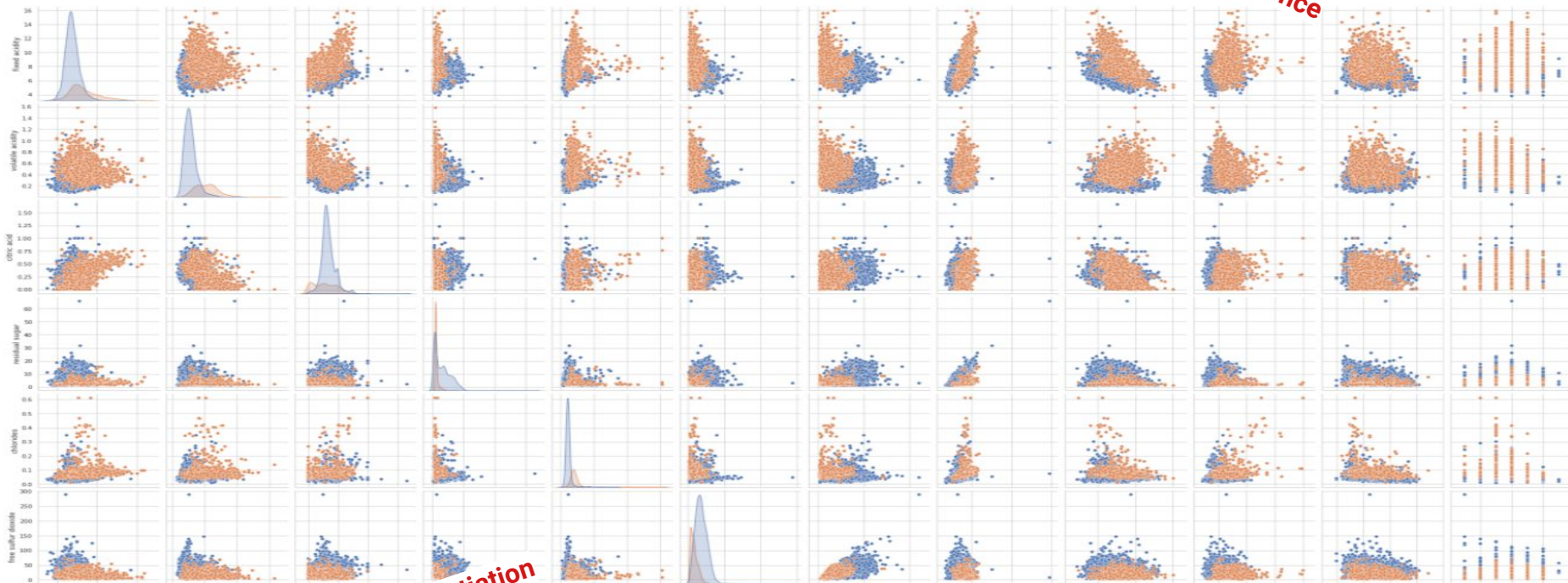
The Dataset

- ◀ Linear correlation is limited
- ◀ Alcohol (0.44), density (0.31), volatile acidity (0.27) and chlorides (0.20) are the highest correlated features





The Dataset



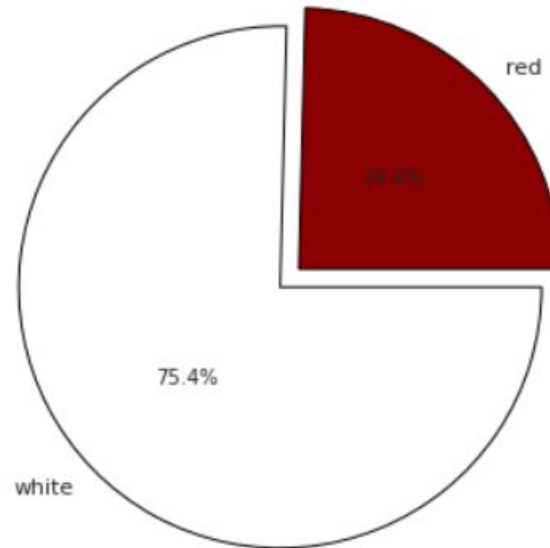
Outlier presence

Wine type differentiation



Type

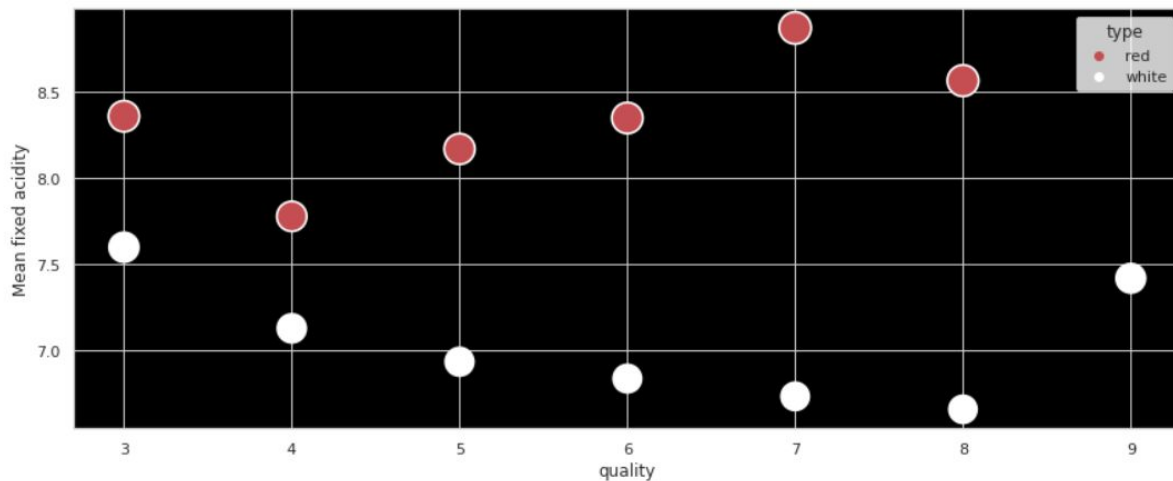
- ◀ Type: Two types of wines: Red wine & white wine
- ◀ Ratio of 3:1 with white dominance



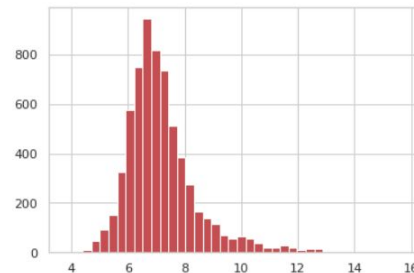


Fixed Acidity

- Fixed acids include tartaric, malic, citric, and succinic acids which are found in grapes . Reducing acids significantly might lead to wines tasting flat.
- There is consistent gap in mean values for red & white wines



count	6,487.00
mean	7.22
std	1.30
min	3.80
25%	6.40
50%	7.00
75%	7.70
max	15.90

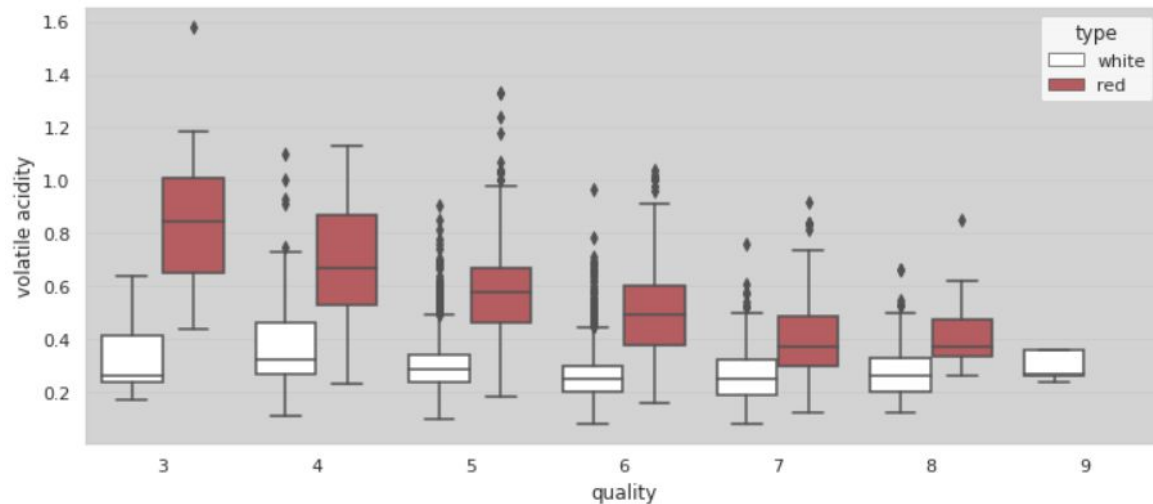




Volatile Acidity

count	6,489.00
mean	0.34
std	0.16
min	0.08
25%	0.23
50%	0.29
75%	0.40
max	1.58

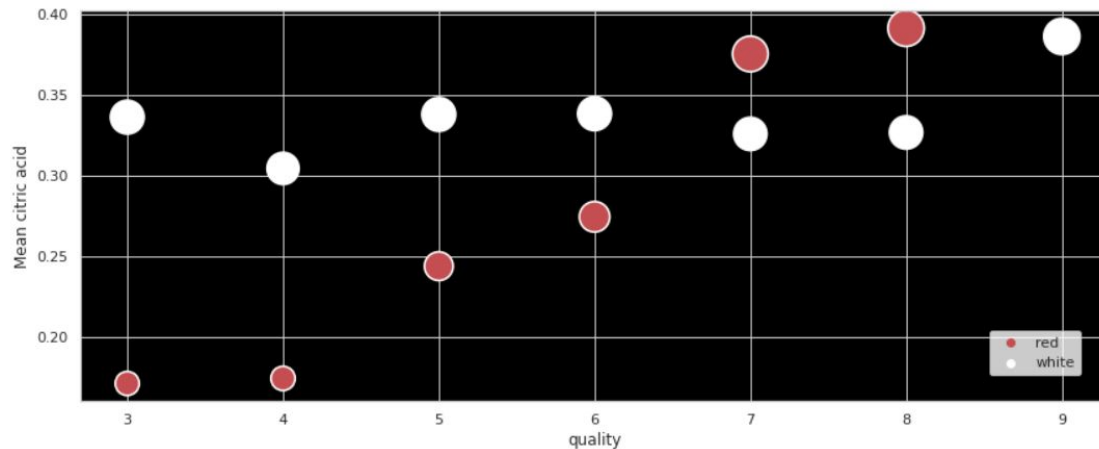
- ◀ Excess of volatile acids are undesirable and lead to unpleasant flavour
- ◀ Red wine volatile acidity is higher than white wines and tends to descent with quality
- ◀ Mean white wines fixed acidity is almost linear



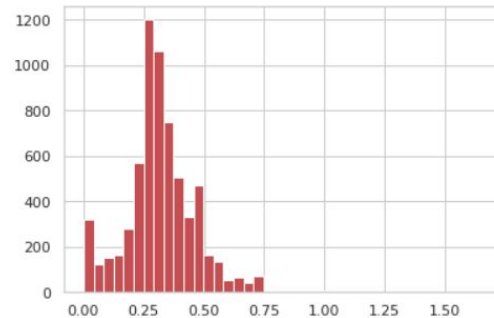


Citric Acid

- ◀ This is one of the fixed acids which gives a wine its freshness. Usually most of it is consumed during the fermentation process and sometimes it is added separately to give the wine more freshness.
- ◀ At quality levels <5 - mean citric acid levels are relatively different



count	6,494.00
mean	0.32
std	0.15
min	0.00
25%	0.25
50%	0.31
75%	0.39
max	1.66

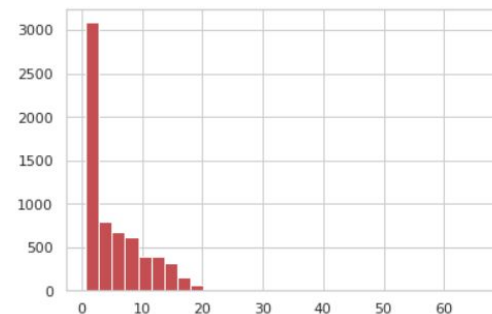
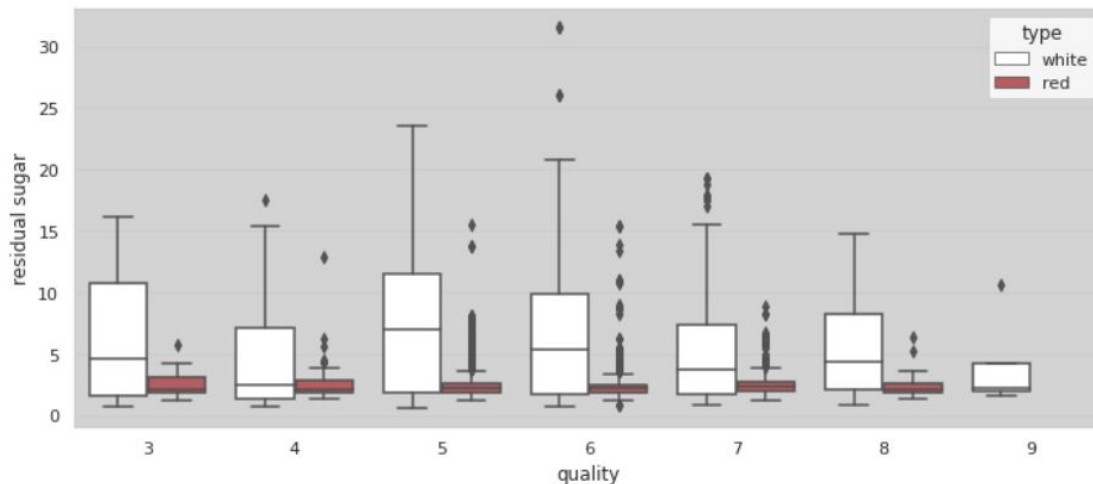




Residual Sugar

count	6,495.00
mean	5.44
std	4.76
min	0.60
25%	1.80
50%	3.00
75%	8.10
max	65.80

- ◀ This typically refers to the natural sugar from grapes which remains after the fermentation process stops, or is stopped.



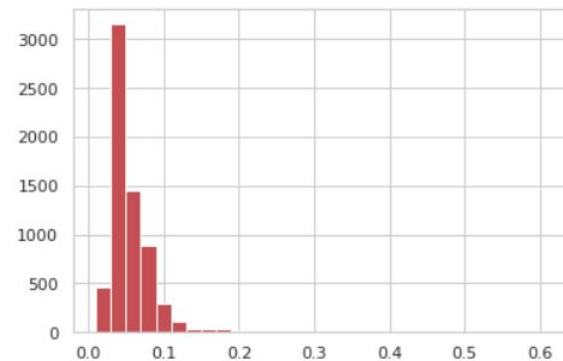
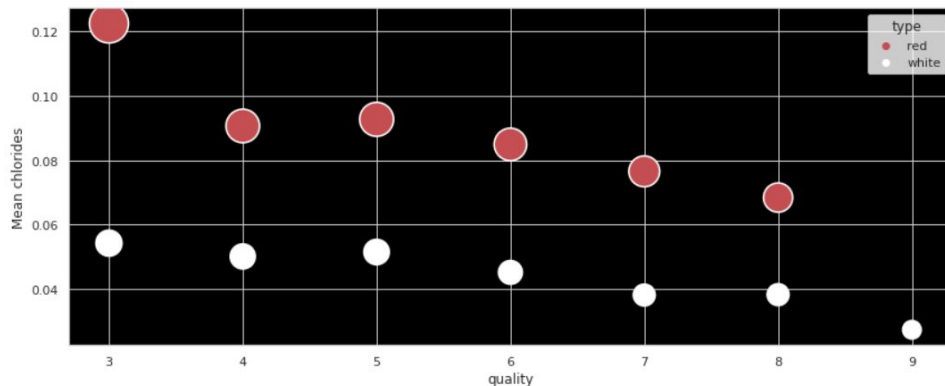
* outlier ~60 removed for plotting purposes



Chlorides

count	6,495.00
mean	0.06
std	0.04
min	0.01
25%	0.04
50%	0.05
75%	0.07
max	0.61

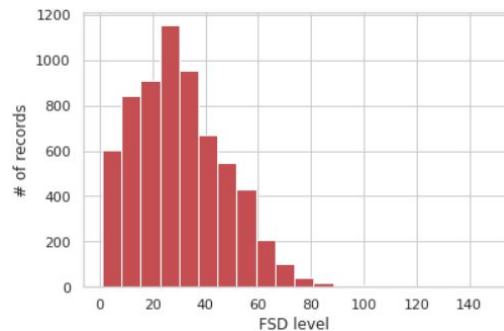
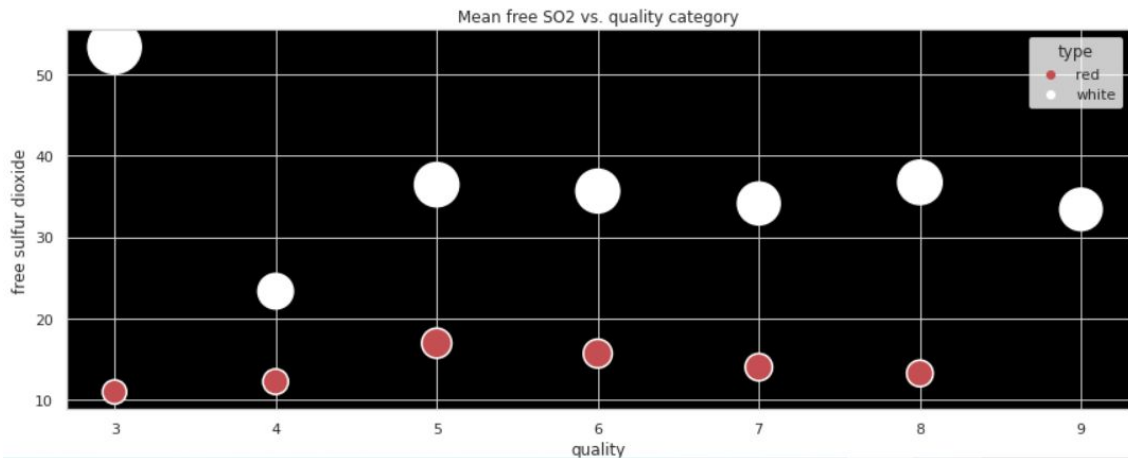
- ◀ This is the chloride concentration in the wine
- ◀ Mean chloride level decline with quality increase
- ◀ Red Wines have higher chloride levels than white wines





Free Sulfur Dioxide (SO_2)

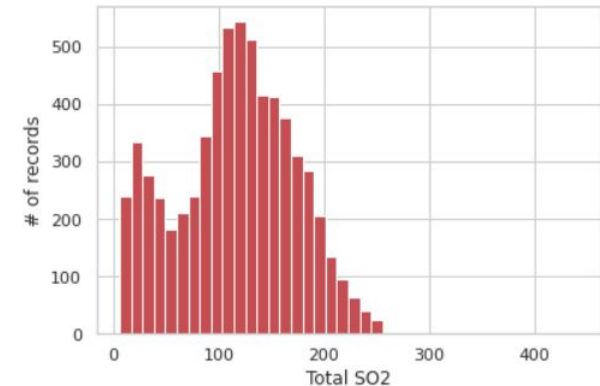
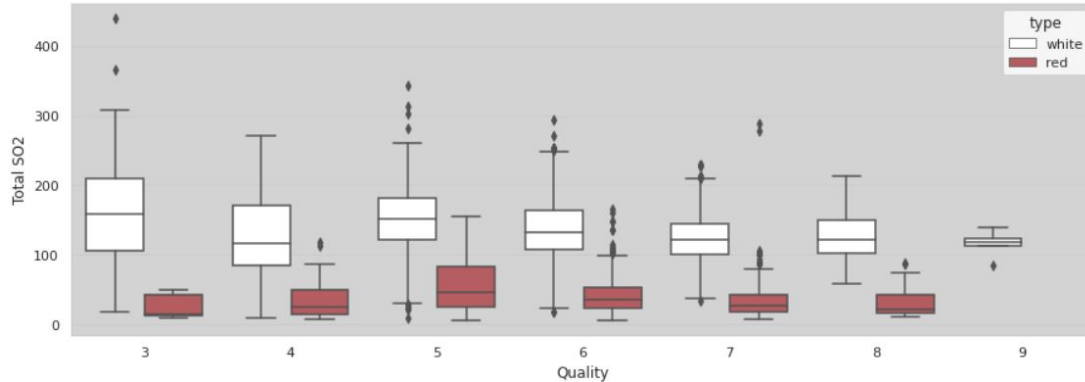
- ◀ Also known as sulfites, too much of it is undesirable and gives a pungent odour.
- ◀ White wines has higher free SO_2 levels
- ◀ Mean level of SO_2 in white wines with quality = 4 is showing anomalous





Total Sulfur Dioxide

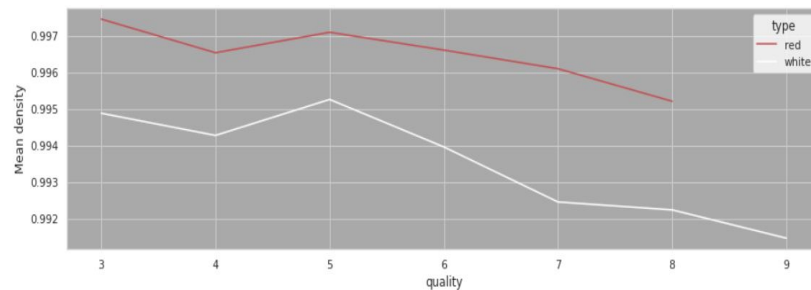
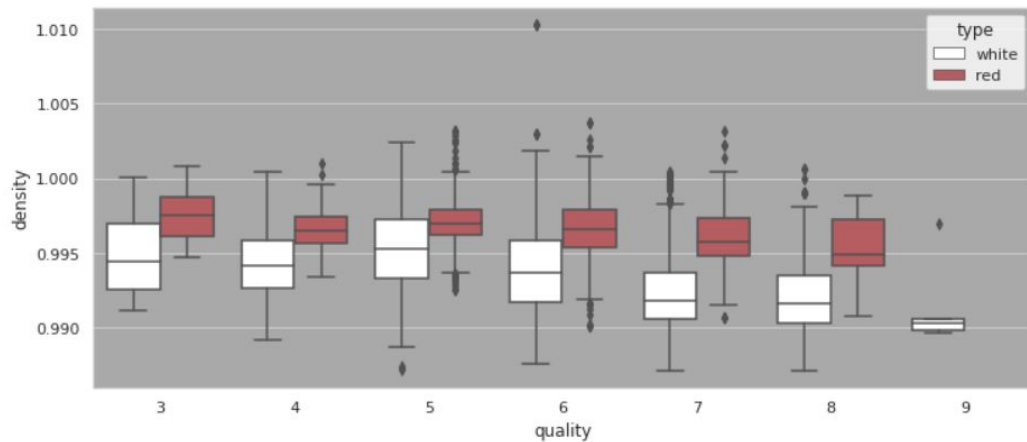
- ◀ This is the sum total of the bound and the free sulfur dioxide. This is mainly added to kill harmful bacteria and preserve quality and freshness.
- ◀ Total mean SO_2 levels are higher in white wines than in red wines, but remain relatively constant among quality groups.





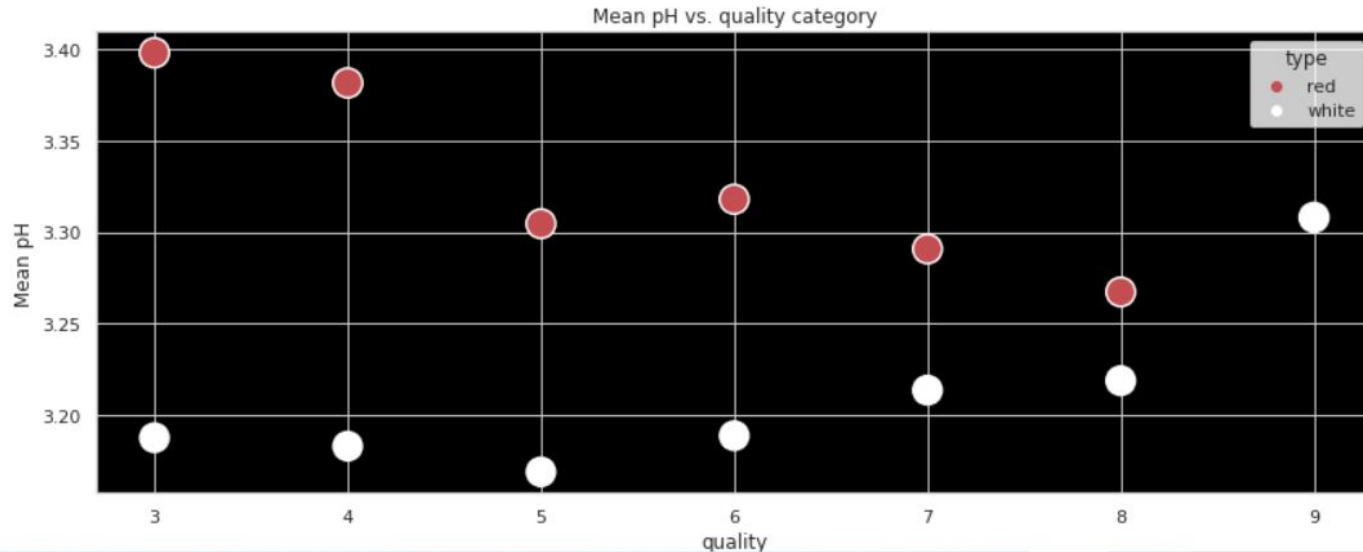
Density

- ◀ This can be represented as a comparison of the weight of a specific volume of wine to an equivalent volume of water.
- ◀ Mean density values decline as quality increase in both types of wines.
- ◀ Majority of values between 0.99 and 1.005



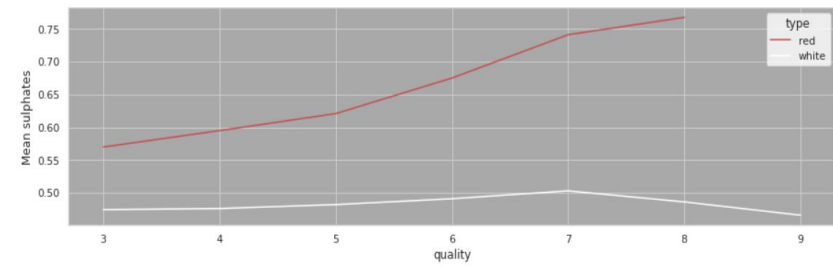


- ◀ Also known as the potential of hydrogen, this is a numeric scale to specify the acidity or basicity the wine. Most wines have a pH between 2.9 and 3.9 and are therefore acidic.
- ◀ Red wines are less acidic than white wines in average when it comes to low qualities levels

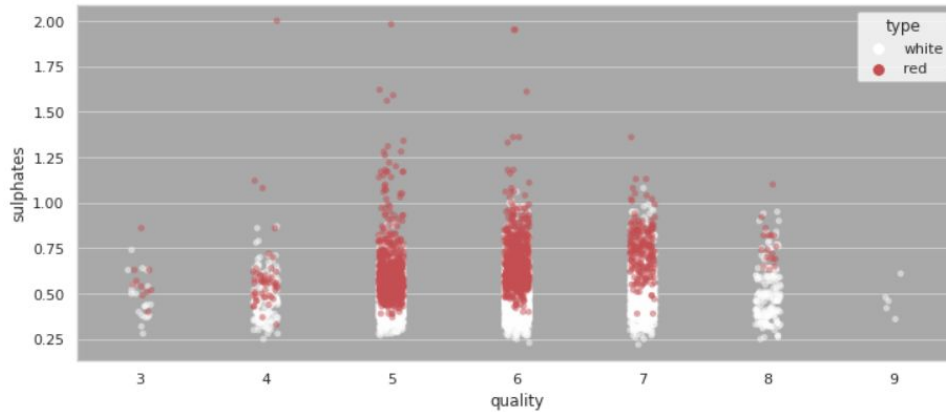




Sulphates



- ◀ These are mineral salts containing sulfur. They are connected to the fermentation process and affects the wine aroma and flavour.
- ◀ Red wines have higher sulphates values than white wines.
- ◀ Mean sulphates level is constant in white wines, and graduating in red wines.

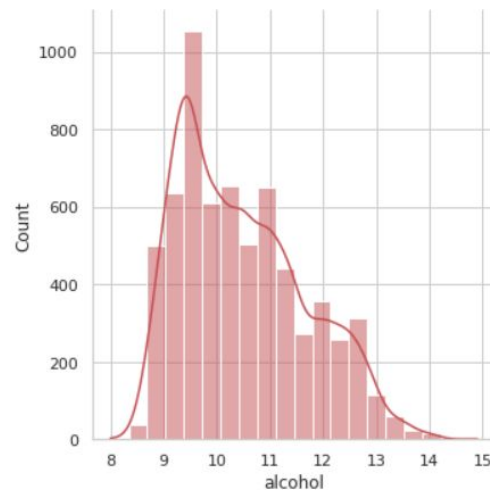
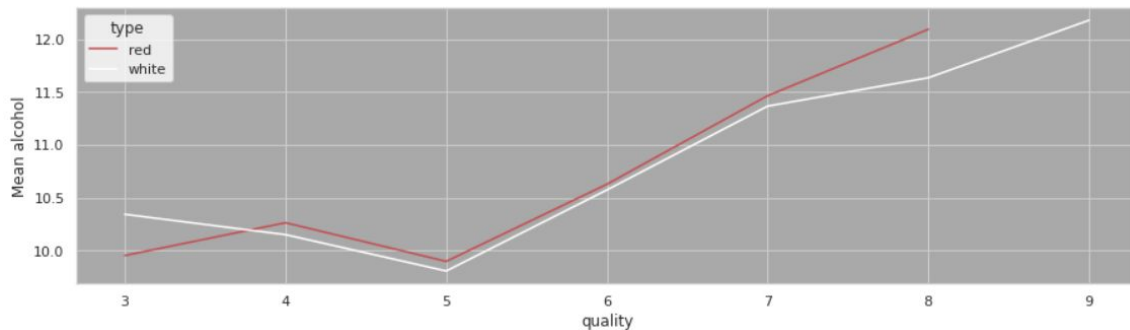




Alcohol

count	6497.000000
mean	10.491801
std	1.192712
min	8.000000
25%	9.500000
50%	10.300000
75%	11.300000
max	14.900000

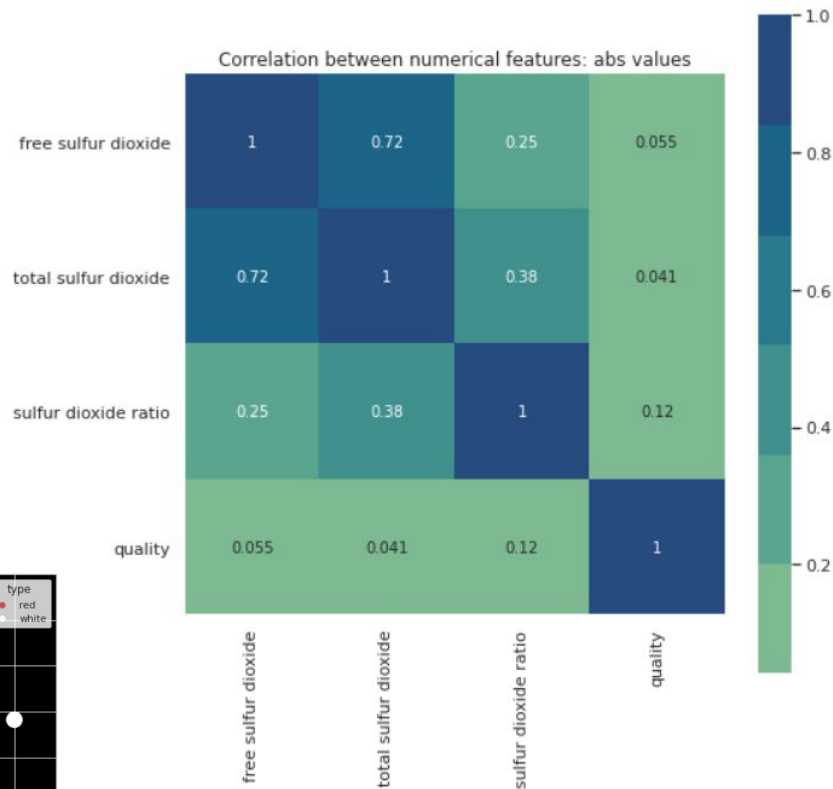
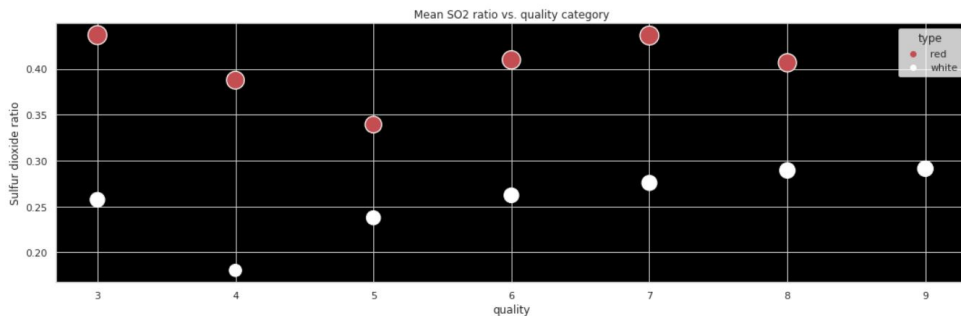
- ◀ It's usually measured in % vol or alcohol by volume (ABV).
- ◀ Alcohol % doesn't vary between wine types, but tend to increase with quality





Sulfur Dioxide Ratio

- Free SO₂ / Total SO₂
- Higher ratio in red wines
- Stronger correlation to quality

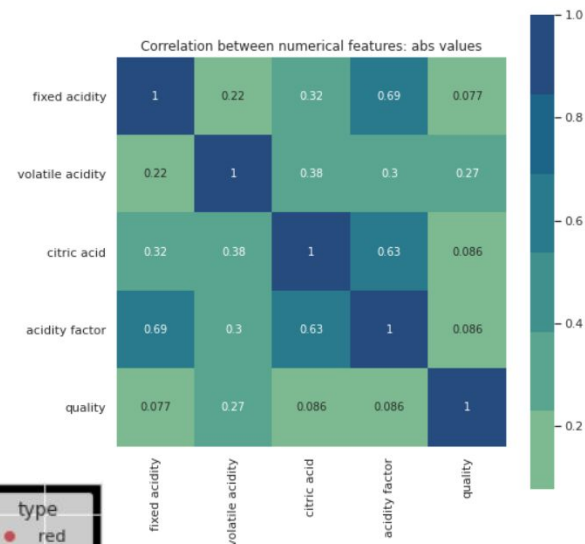
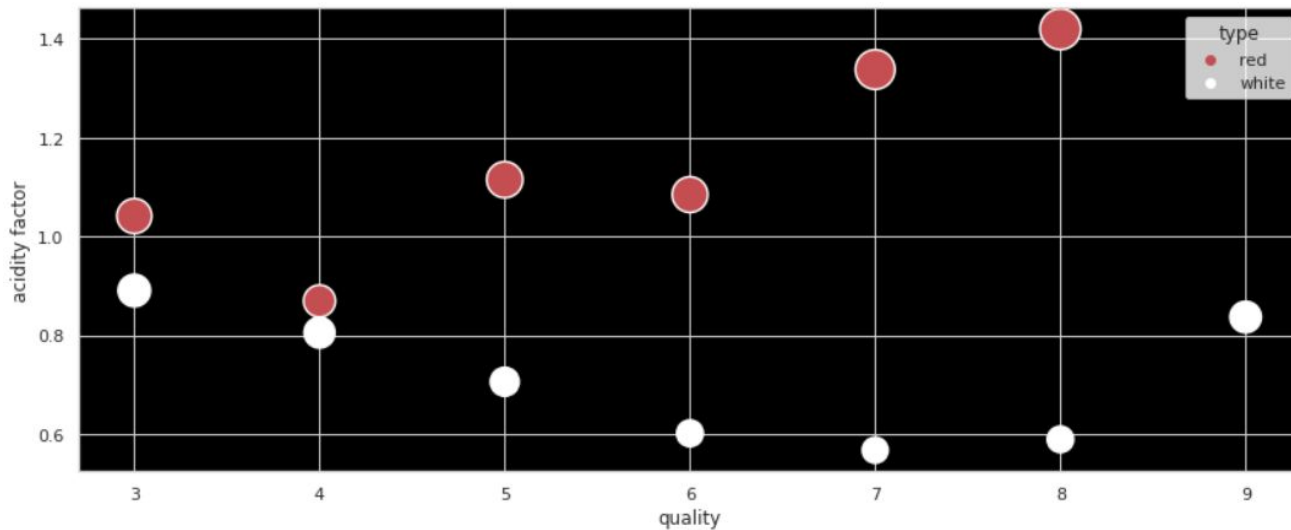


Total SO ₂				
Molecular SO ₂	Sulphite SO ₃ -	Bisulphite HSO ₃ -	Unstable compounds (sugars, uronic acids, ketonic acids, etc.)	Stable compounds (acetaldehyde)
Free SO ₂			Bound SO ₂	



Acidity Factor

- ◀ Multiply of the 3 acid features to create a new feature
- ◀ Fixed at 3 & 4, drift apart at higher quality levels

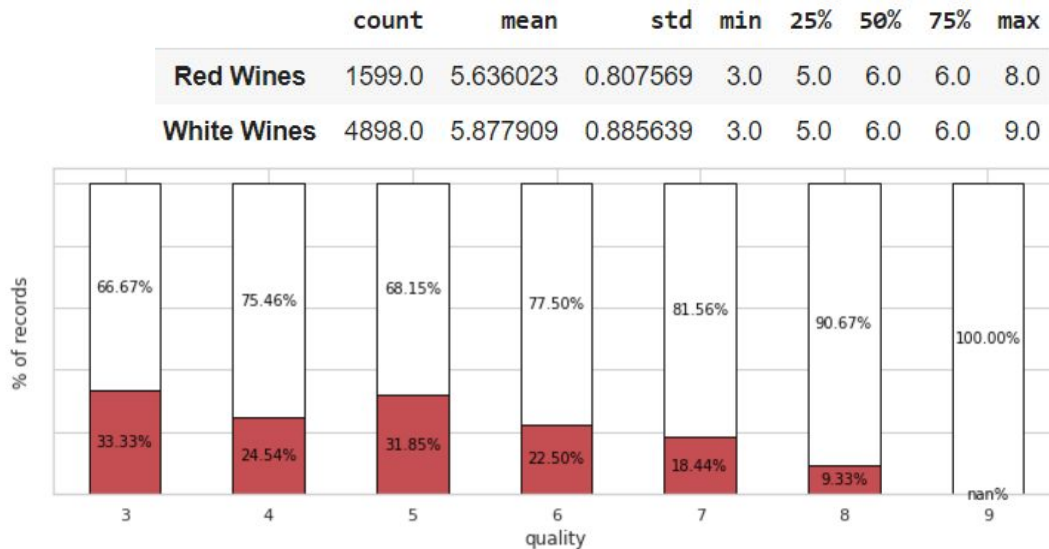




Quality (Target)

- Wine experts graded the wine quality between 0 (very bad) and 10 (very excellent). The eventual quality score is the median of at least three evaluations made by the same wine experts.
- imbalanced data
- 9 is applicable at white wines only

(with 5 observations!)





Data Insights

- ◀ Most features show different behavior regarding the wine type, but the type by itself can't predict the quality.
- ◀ Should we predict the wine type, one could expect high prediction rates.
- ◀ Quality prediction rate is expected to be low, due to imbalanced dataset
- ◀ The ability to predict categories 3, 4, 8 & 9 is expected to be very low due to small sample size
- ◀ Imputing nulls must take into consideration the wine type
- ◀ Outliers removals is essential in almost 50% of features
- ◀ Scaling is required due to different values scale (<1, 10s, 100s)
- ◀ The engineered features were aimed to unite similar indicative feature to a more solid feature. But while in sulfur we decided to drop the original features, at acidity we will use it all.
- ◀ **We suspect that applying different prediction models on white and red wines may get higher prediction score.**



Pre Processing



Pre-Processing

Apply in	Transformer	Alcohol	Acidity factor	citric acid	chlorides	Density	fixed acidity	Free Sulfur Dioxide
Train	Outlier_limit			V	V	V		V
Train + Test	SimpleImputer			V	V			
	MinMaxScaler	V	V	V	V	V	V	
	Other		eng. Feature					

Apply in	Transformer	Sulfur Dioxide Ratio	pH	residual sugar	Sulphates	Total Sulfur Dioxide	type	volatile acidity
Train	Outlier_limit			V	V	V		V
Train + Test	SimpleImputer		V	V				V
	MinMaxScaler		V	V	V			V
	Other	eng. Feature					[0,1]	



Pre-Processing: Outliers removal

- ◀ Dictionary with modified limits to features
- ◀ limit_value function
- ◀ FunctionTransformer applied

```
1 def limit_value(X, **val_dict):
2     """This function recieves a dataframe and returns
3     it without the outliers
4     """
5     for col, val in val_dict.items():
6         X = X.loc[X[col] < val, :]
7     return X
```

```
1 # Dictionary of outliers
2 outlier_dict = {'free sulfur dioxide': 150,
3                'total sulfur dioxide': 400,
4                'density': 1.01,
5                'sulphates': 1.75,
6                'volatile acidity': 1.5,
7                'citric acid': 1.2,
8                'residual sugar': 50,
9                'chlorides': 0.5}
10
11 # Clean outliers
12 outlier_limit = FunctionTransformer(limit_value, kw_args = outlier_dict)
13 df = outlier_limit.transform(df)
```




Pre-Processing: Feature selection & split



```
1 col_to_drop = ['free sulfur dioxide', 'total sulfur dioxide']  
2 df = df.drop(labels=col_to_drop, axis=1)
```

```
1 # split the data  
2 X_train, X_test, y_train, y_test =  
3     split(df.drop('quality', axis=1), df['quality'],  
4           test_size = 0.25,  
5           random_state = 12345,  
6           stratify=df['quality'])
```

X_Train: (4849, 12) | y_Train: 4849

X_Test: (1617, 12) | y_Test: 1617



Apply Imputer & Scaler

```
1 cols = ['fixed acidity', 'volatile acidity', 'citric acid',
2         'residual sugar', 'chlorides', 'density', 'pH',
3         'sulphates', 'alcohol', 'acidity factor']
4
5 imputer = SimpleImputer(strategy = 'mean')
6 scaler = MinMaxScaler()
7
8 X_train.loc[X_train['type'] == 0] = imputer.fit_transform(X_train.loc[X_train['type'] == 0])
9 X_train.loc[X_train['type'] == 1,:] = imputer.fit_transform(X_train.loc[X_train['type'] == 1])
10
11 X_train.loc[X_train['type'] == 0,cols] = scaler.fit_transform(X_train.loc[X_train['type'] == 0,cols])
12 X_train.loc[X_train['type'] == 1,cols] = scaler.fit_transform(X_train.loc[X_train['type'] == 1,cols])
13
14 X_test.loc[X_test['type'] == 0] = imputer.fit_transform(X_test.loc[X_test['type'] == 0])
15 X_test.loc[X_test['type'] == 1] = imputer.fit_transform(X_test.loc[X_test['type'] == 1])
16
17 X_test.loc[X_test['type'] == 0,cols] = scaler.fit_transform(X_test.loc[X_test['type'] == 0,cols])
18 X_test.loc[X_test['type'] == 1,cols] = scaler.fit_transform(X_test.loc[X_test['type'] == 1,cols])
```

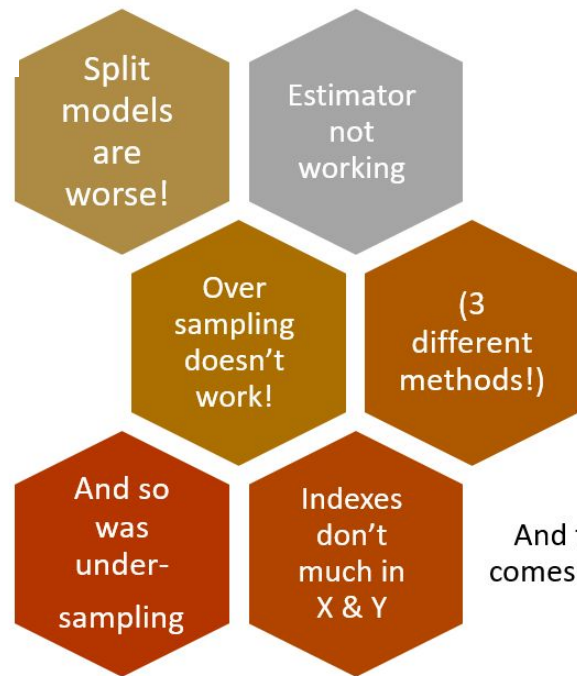


Plan vs. Reality...

Plan



Reality



And then Dror comes to rescue...



Split Model

- ◀ Get 2 model - one for red, one for white
- ◀ In fit: fit each population with its own model
- ◀ In transform: predict the population, concat the data and return a reinexed y series

```
1 class run_estimator (BaseEstimator, TransformerMixin):
2     """ This transformer recives a DF(X) and a target(y),
3         and split it to two populations: red wines and white wines
4     """
5
6     def __init__(self, model_r, model_w, classes=[0,1]):
7         self.red_model_ = model_r
8         self.white_model_ = model_w
9         self.classes = classes
10
11
12     def fit (self, X, y=None):
13         X_red = X[X.type==0].copy()
14         y_red = y[X.type==0].copy()
15         self.red_model_.fit(X_red, y_red)
16         X_white = X[X.type==1].copy()
17         y_white = y[X.type==1].copy()
18         self.white_model_.fit(X_white, y_white)
19         return self
20
21     def predict(self, X):
22         X_red = X[X.type==0].copy()
23         X_white = X[X.type==1].copy()
24         y_red_pred = pd.Series(self.red_model_.predict(X_red),index=X_red.index)
25         y_white_pred = pd.Series(self.white_model_.predict(X_white),index=X_white.index)
26         y_pred = pd.concat([y_red_pred, y_white_pred], axis=0)
27         return y_pred.reindex_like(X)
```



Cohen's Kappa score

$$k \equiv \frac{p_o - p_e}{1 - p_e}$$

`kappa_scorer = make_scorer(cohen_kappa_score)`

Cohen's kappa coefficient (κ) is a statistic which measures inter-rater agreement for qualitative (categorical) items. It is generally thought to be a more robust measure than simple percent agreement calculation, as κ takes into account the possibility of the agreement occurring by chance.

		B	
		Yes	No
A	Yes	a	b
	No	c	d

		B	
		Yes	No
A	Yes	10	15
	No	20	05

The observed proportionate agreement is:

$$p_o = \frac{a+d}{a+b+c+d} = \frac{10+5}{50} = 0.3$$

$$p_{Yes} = \frac{a+b}{a+b+c+d} \cdot \frac{a+c}{a+b+c+d}$$

$$p_{No} = \frac{c+d}{a+b+c+d} \cdot \frac{b+d}{a+b+c+d}$$

$$p_e = p_{Yes} + p_{No}$$

Kappa value interpretation Landis & Koch (1977):

<0 No agreement

0 — .20 Slight

.21 — .40 Fair

.41 — .60 Moderate

.61 — .80 Substantial

.81–1.0 Perfect

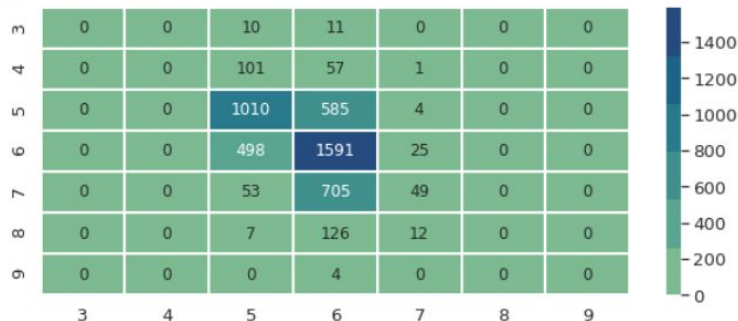


Baseline code without gridsearch

```
1 # Run Baseline Logistic Regression with all data
2 model = LogisticRegression(multi_class = 'ovr')
3 model.fit(X_train, y_train)
4 y_train_pred = model.predict(X_train)
5 y_test_pred = model.predict(X_test)
6
7 # CrossValidation
8 cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=123)
9 scores = cross_val_score(model, X_train, y_train, cv=cv, scoring=kappa_scorer)
10
11 # Print the results
12 print('CV:', scores)
13 print('CV mean:', scores.mean())
14 print(f'Train Cohen kappa score is: {cohen_kappa_score(y_train, y_train_pred):.3}')
15 print(f'Test Cohen kappa score is: {cohen_kappa_score(y_test, y_test_pred):.3}')
16 cm_plot(confusion_matrix(y_train, y_train_pred), model)
```

```
1 def cm_plot(cm, model):
2     fig, ax = plt.subplots(figsize = (10,4))
3     df_cm = pd.DataFrame(cm, index = [i for i in model.classes_],
4                           columns = [i for i in model.classes_])
5     sns.set(font_scale=1.2)
6     cmap = sns.color_palette("crest")
7     ax = sns.heatmap(df_cm, cmap = cmap, annot=True,
8                     fmt = 'd',
9                     linewidth = 1,
10                    annot_kws={"size": 12})
11     return plt.show()
```

CV: [0.21214465 0.25169972 0.26543686 0.29325305 0.23251694]
CV mean: 0.25102447713094
Train Cohen kappa score is: 0.251
Test Cohen kappa score is: 0.239



Output



Model Results: LogisticRegrssion

```
# Logistic Regression Classifier
model_w = LogisticRegression(multi_class = 'ovr', random_state=42, penalty='l2', C=1,
                             class_weight={3:2, 4:5, 5:6, 6:6, 7:8, 8:4, 9:1})
model_r = LogisticRegression(multi_class = 'ovr', random_state=42, penalty='l2', C=1.6,
                             class_weight={3: 1, 4: 3, 5: 5, 6: 7, 7: 9, 8: 1})
```

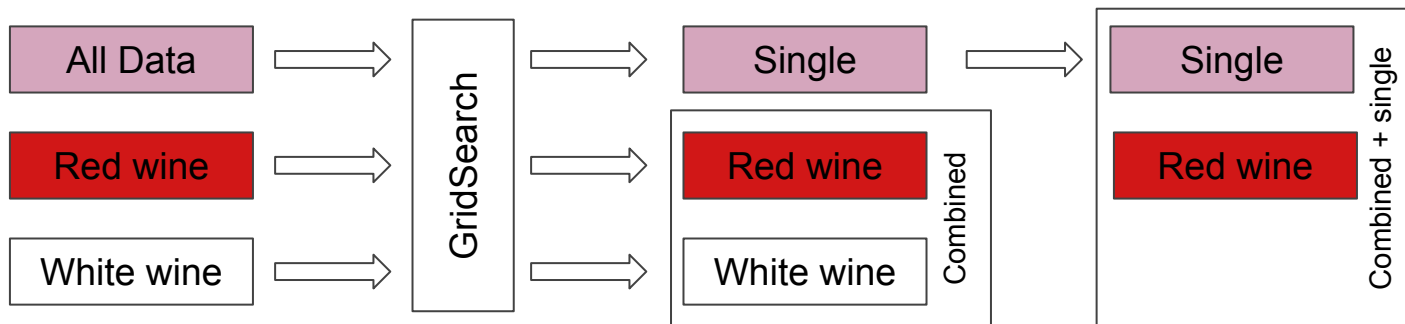
	Single: Baseline	Red	White	Combined
Train	0.251	0.35	0.28	0.301
Test	0.239	0.33	0.25	0.269
Mean CV	0.241	0.31	0.24	0.271



Model Results: VotingClassifier

```
classifiers = [('LR', LogisticRegression(multi_class='ovr', random_state=42)),
               ('DTC', DecisionTreeClassifier(max_depth=5, random_state=42)),
               ('SVC', SVC(probability=True, random_state=42))]
```

	Single	Red	White	Combined	Combined+ Single as white
Train	0.66	0.45	0.32	0.36	0.745
Test	0.3	0.33	0.26	0.283	0.306
Mean CV	0.33	0.32	0.26	0.273	0.33





Model Results: XGBoost

	Single	Red	White	Combined	Combined+Single as white
Train	0.99	0.4	0.74	0.663	0.854
Test	0.41	0.32	0.41	0.393	0.441
Mean CV	0.42	0.32	0.39	0.365	0.394

```
model_r = XGBClassifier(random_state=0, max_depth=2,  
                        n_estimators=30, colsample_bytree=0.8,  
                        subsample = 0.3)  
model_w = XGBClassifier(random_state=0, max_depth=8, n_estimators=100,  
                        colsample_bytree=0.9, alpha=1,  
                        objective='multi:softmax')
```



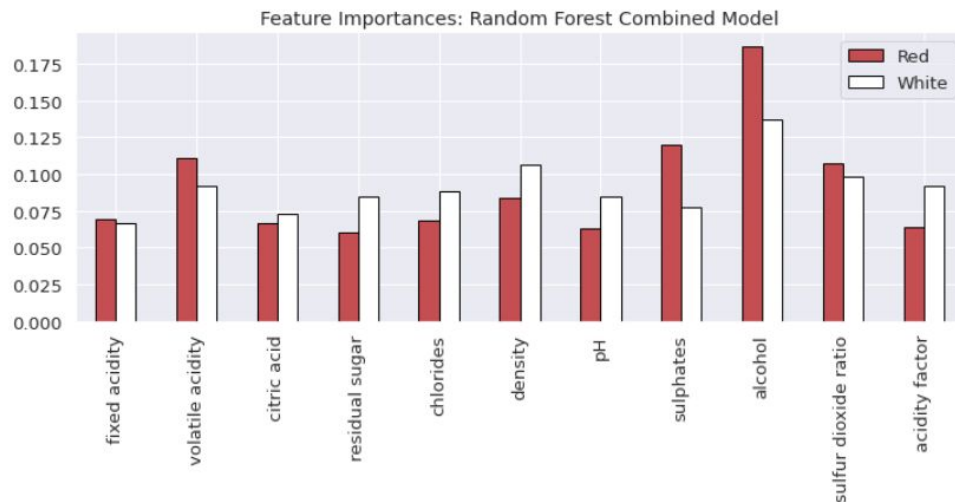
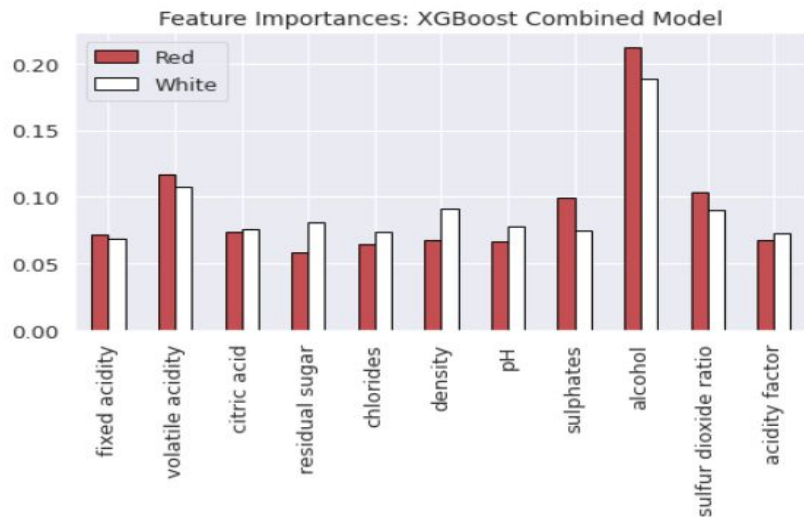
Model Results: RandomForest

	Single	Red	White	Combined	Combined+Single as white
Train	0.97	0.8	0.92	0.891	0.951
Test	0.42	0.33	0.44	0.418	0.453
Mean CV	0.44	0.41	0.43	0.441	0.45

```
model_r = RandomForestClassifier (n_estimators=120,  
                                random_state=0, class_weight={3: 1, 4: 3, 5: 6, 6: 7, 7: 9, 8: 1},  
                                max_depth = 8)  
model_w = RandomForestClassifier(random_state=0, max_depth = 15, n_estimators=150,  
                                class_weight={3:2, 4:2, 5:7, 6:6, 7:9, 8:4, 9:1})
```



Feature Importance

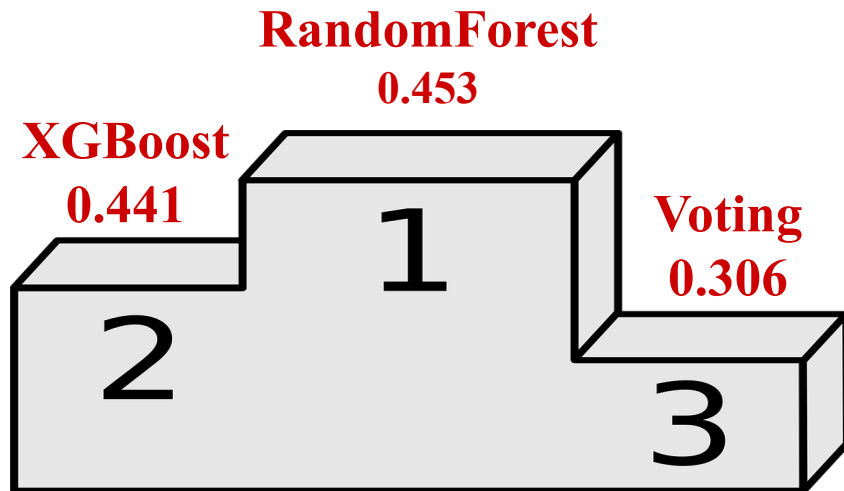


- ▶ Alcohol is dominant in both models
- ▶ red and wine have different importance
- ▶ Out engineered features proved to be valuable!



Conclusions

- ◀ Combined data failed to improve model scores
- ◀ Nevertheless, splitting the data into 2 populations was worth the effort
- ◀ Applying grid search on each population and the joined data gave added value





Further exploring

1. Oversampling \ undersampling should be executed on the split model
2. Combining 2 different models (instead of one model with different hyper-parameters)
3. More time to GridSearch and improve the scores ;-)
4. Binning the quality target into good, medium and bad
5. Predicting wine type

Thank You

