Scuola universitaria professionale della Svizzera italiana
Bachelor di Ingegneria Informatica

SUPSI

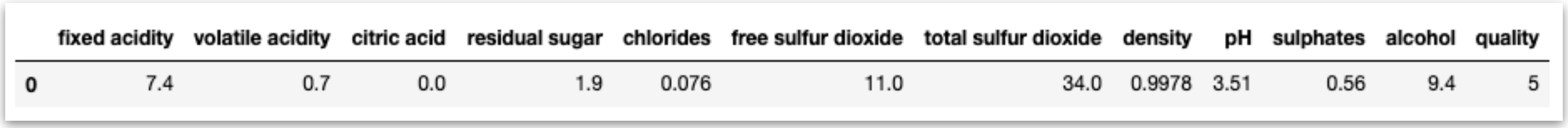
Red wine quality prediction

Machine Learning course project

Nicolò Rubattu

nicolo.rubattu@student.supsi.ch

The Dataset



	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
0	7.4	0.7	0.0	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5

- Red and White Wine
  The inputs include objective tests (e.g. PH values) and the output is based on sensory data (median of at least 3 evaluations made by wine experts). Each expert graded the wine quality between 0 (very bad) and 10 (very excellent)

- Objective: <u>predict quality</u> → *Regression* + *Classification*

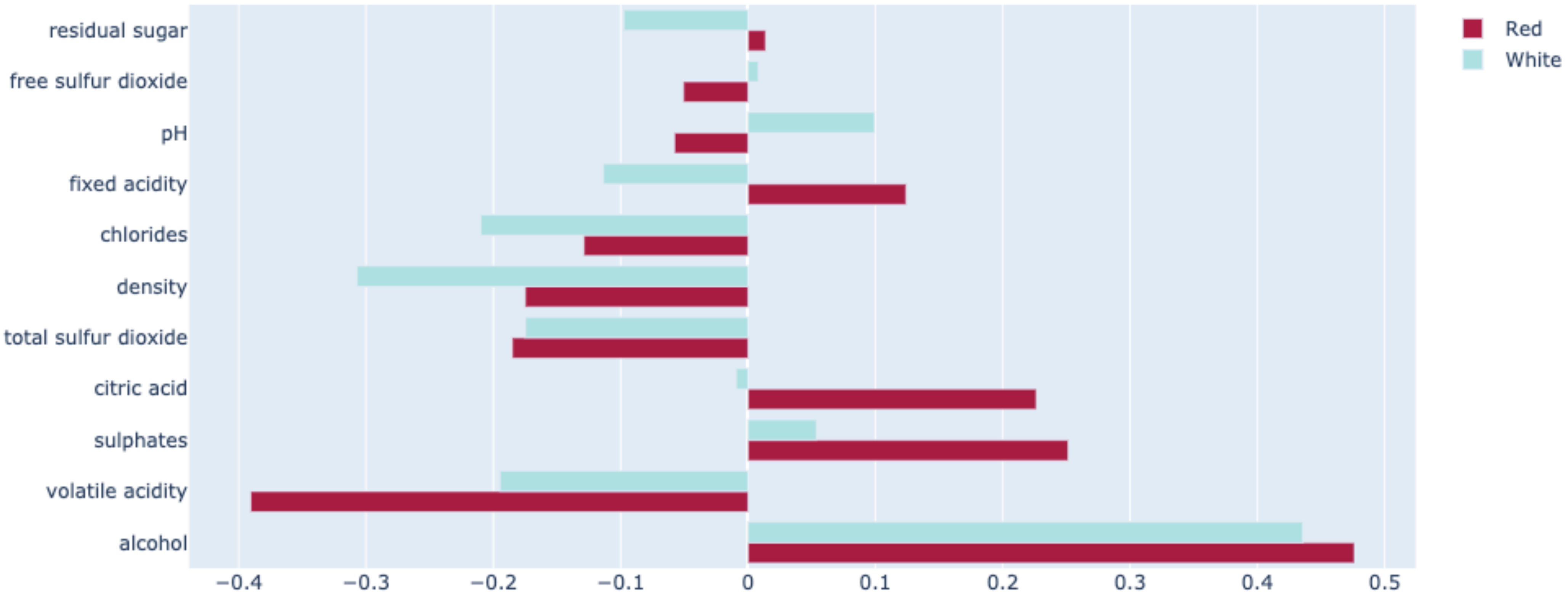
- *Credits:*

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. ISSN: 0167-9236.

Available at: [@Elsevier] http://dx.doi.org/10.1016/j.dss.2009.05.016
[Pre-press (pdf)] http://www3.dsi.uminho.pt/pcortez/winequality09.pdf
[bib] http://www3.dsi.uminho.pt/pcortez/dss09.bib

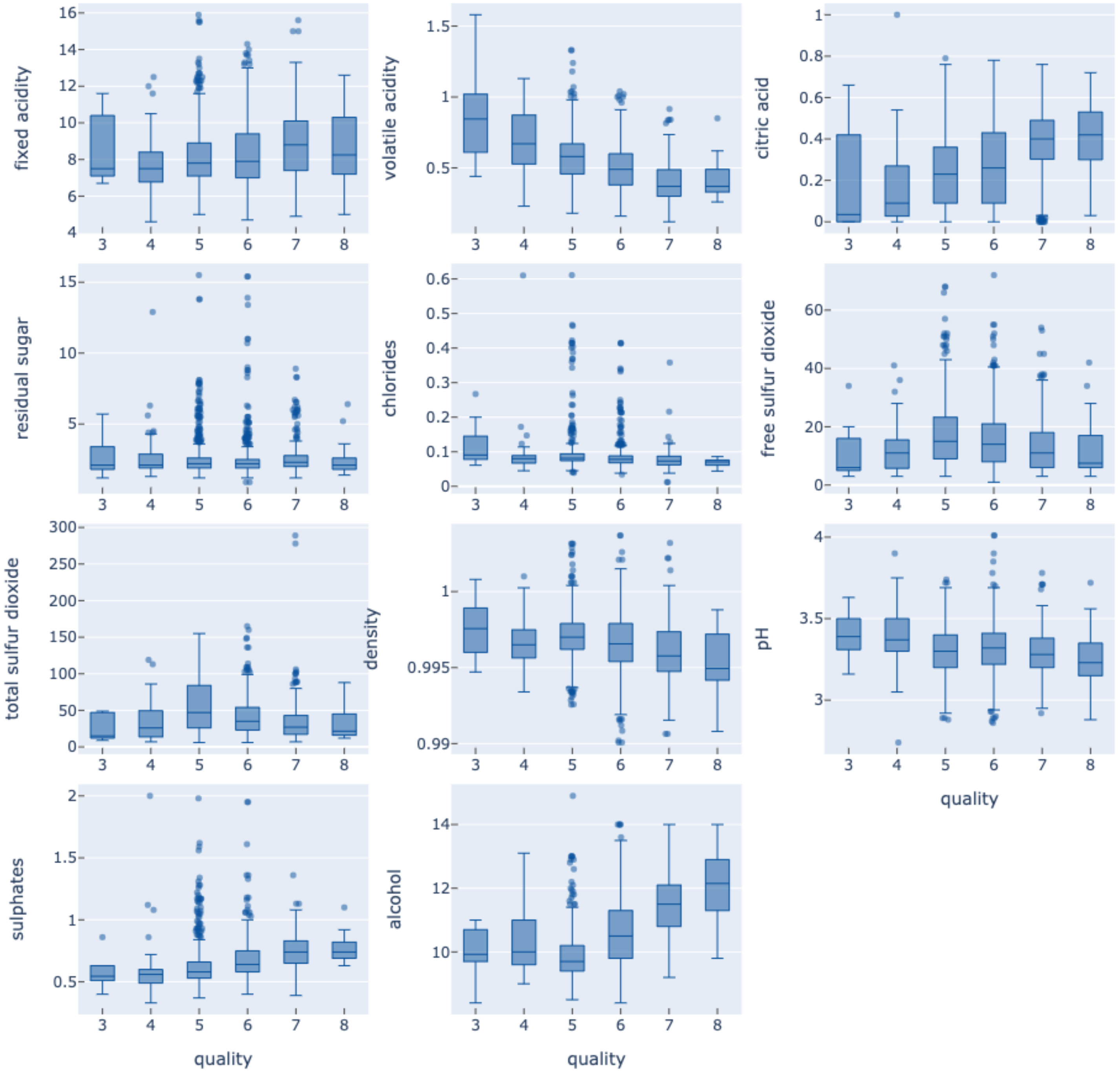
Data Analisys

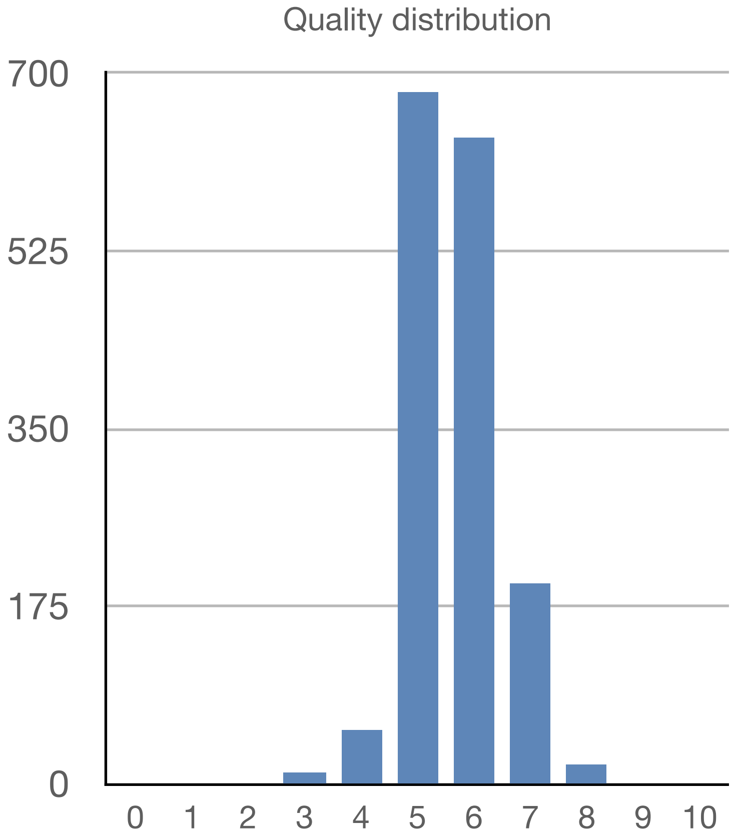
- Number of Instances: red wine - 1599; white wine 4898
- Number of Attributes: 11 (fixed acidity, volatile acidity, citric acid, residual sugar, chlorides, free sulfur dioxide, total sulfur dioxide, density, pH, sulphates, alcohol) + quality
- Statistic analysis, outliers, NaN values, correlation analysis



Red Wine choice

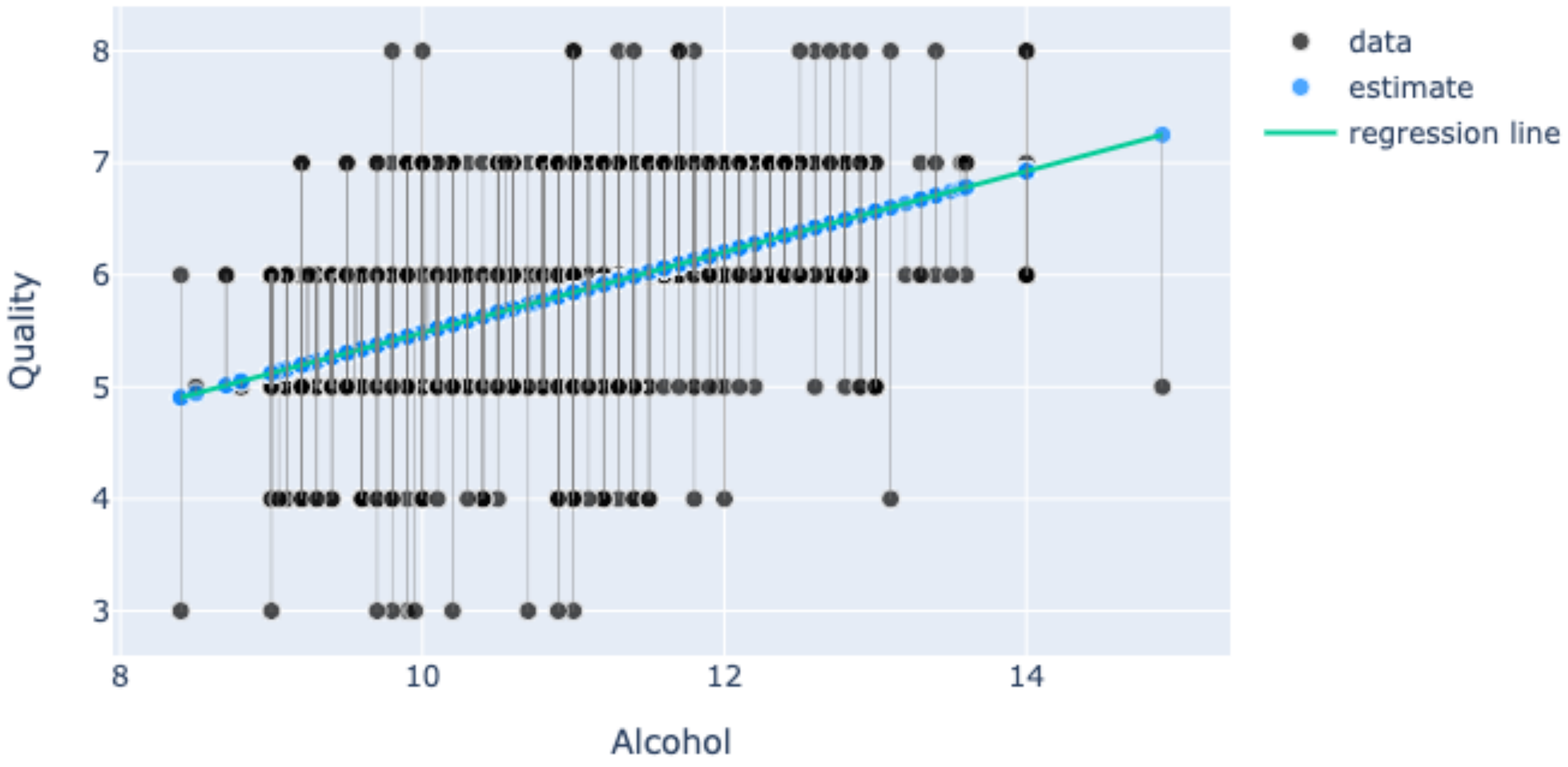
BoxPlot overview of features and target correlation with outliers check.

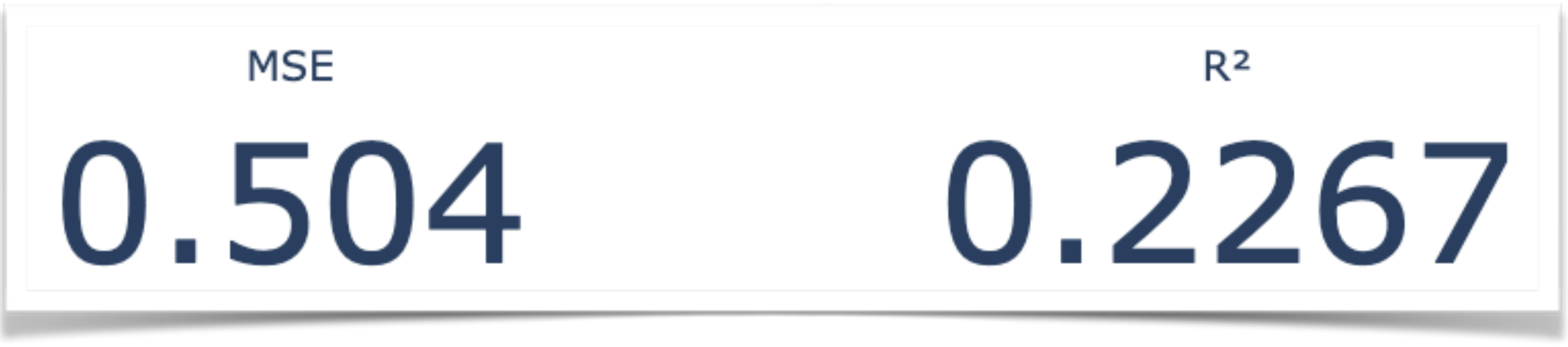



Quality distribution

Regression

Univariate linear regression (alcohol)





MSE

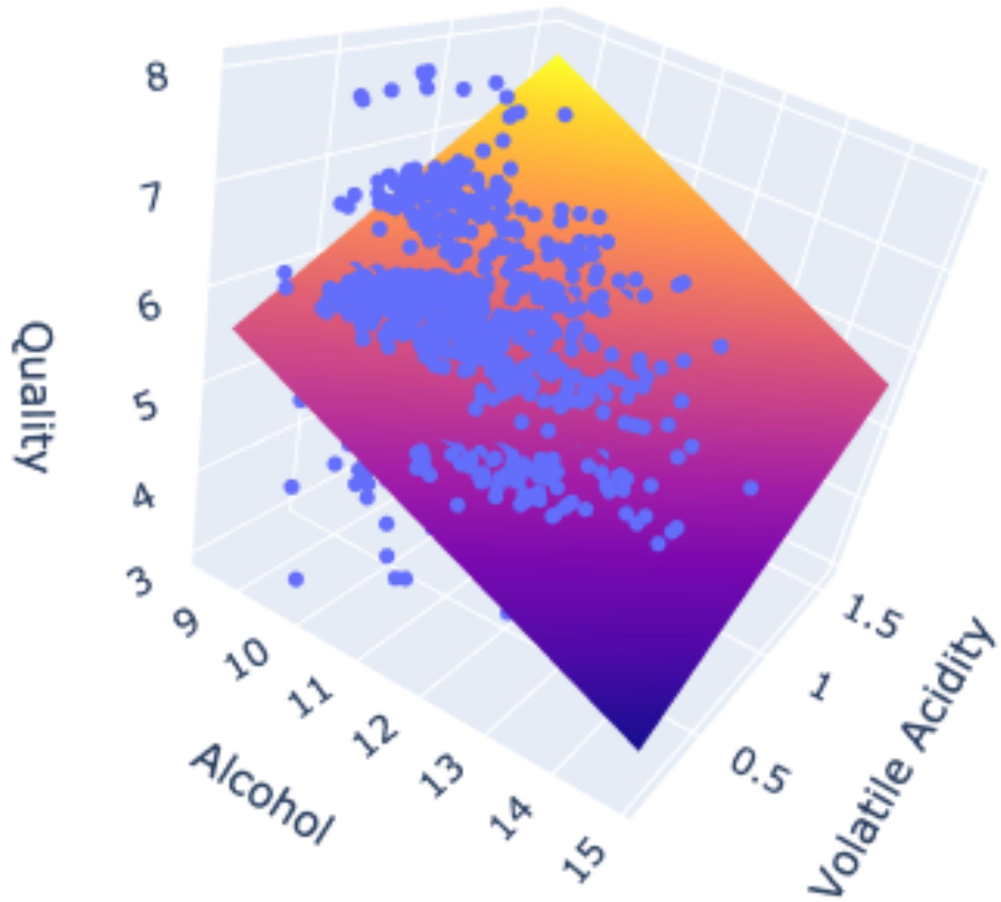
0.504

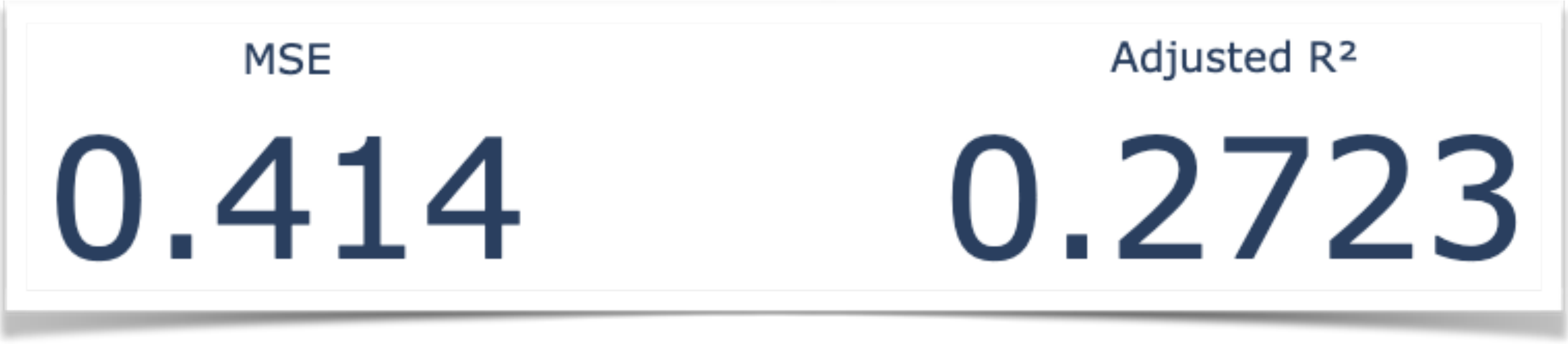
R²

0.2267

Regression

Simple Multivariate linear regression (alcohol + volatile acidity)




MSE

0.414

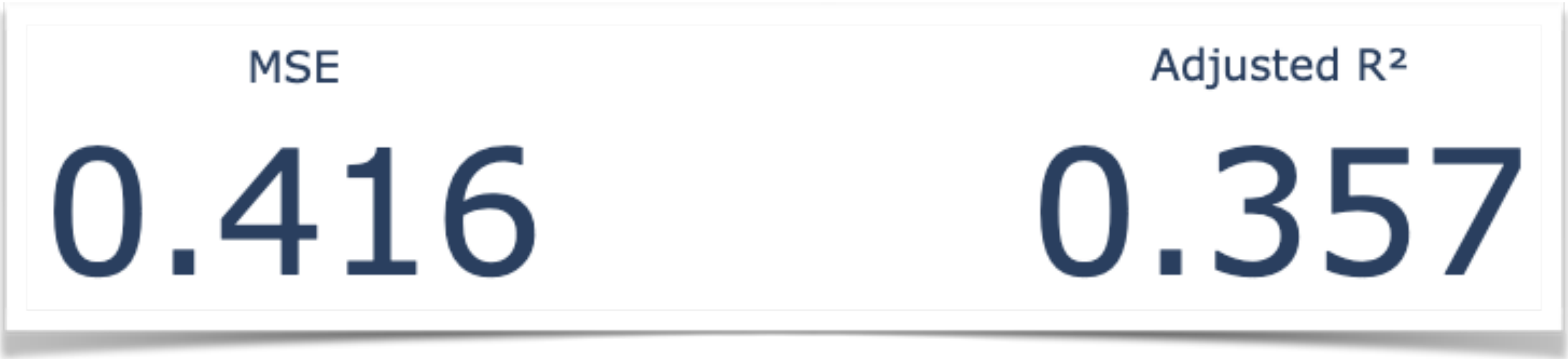
Adjusted R²

0.2723

Regression

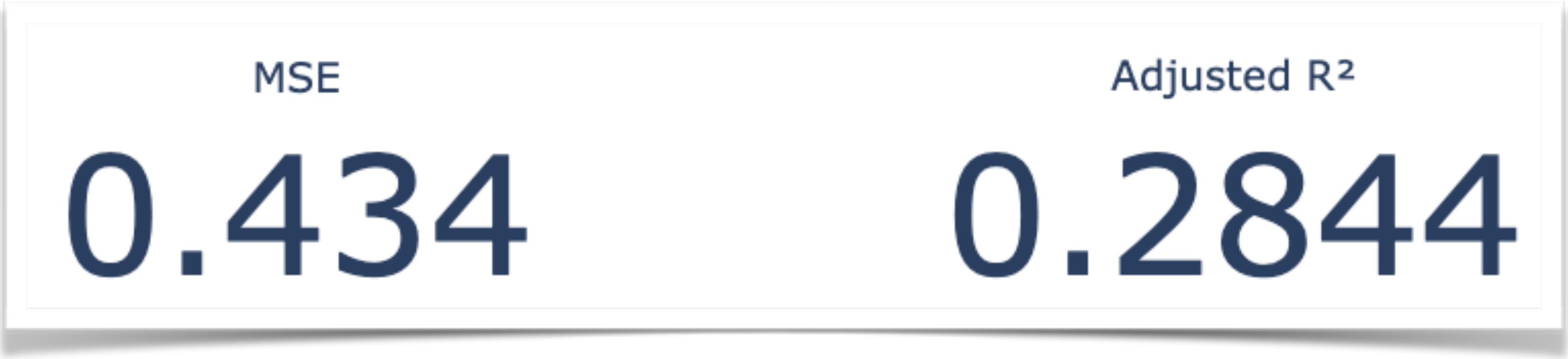
Multivariate linear regression (all features) with Stratified KFold Validation

- Training Set



MSE	Adjusted R^2
0.416	0.357

- Test Set

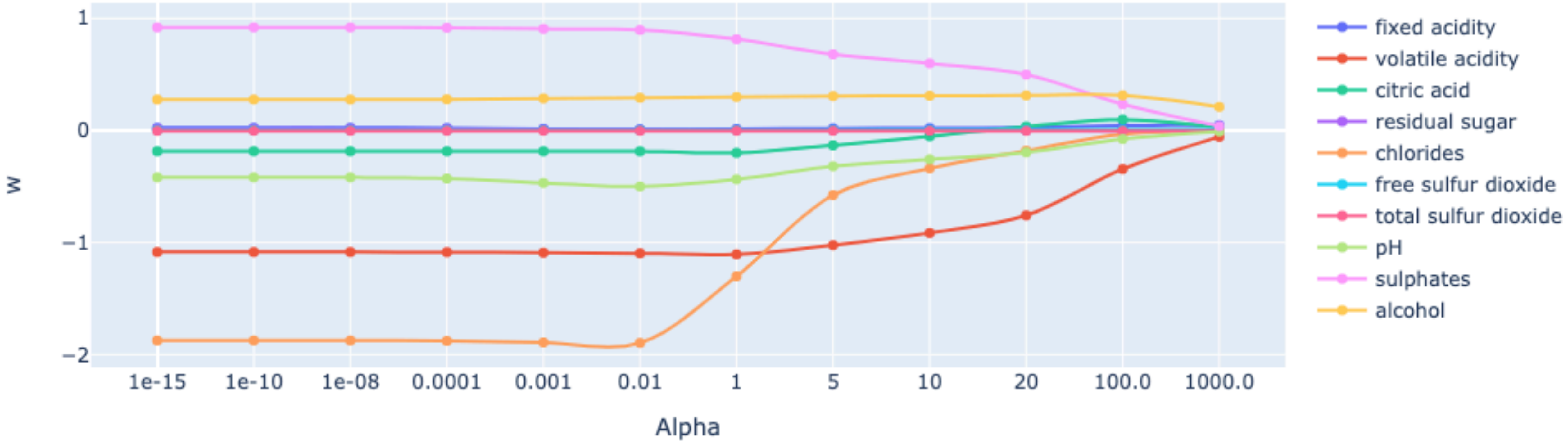


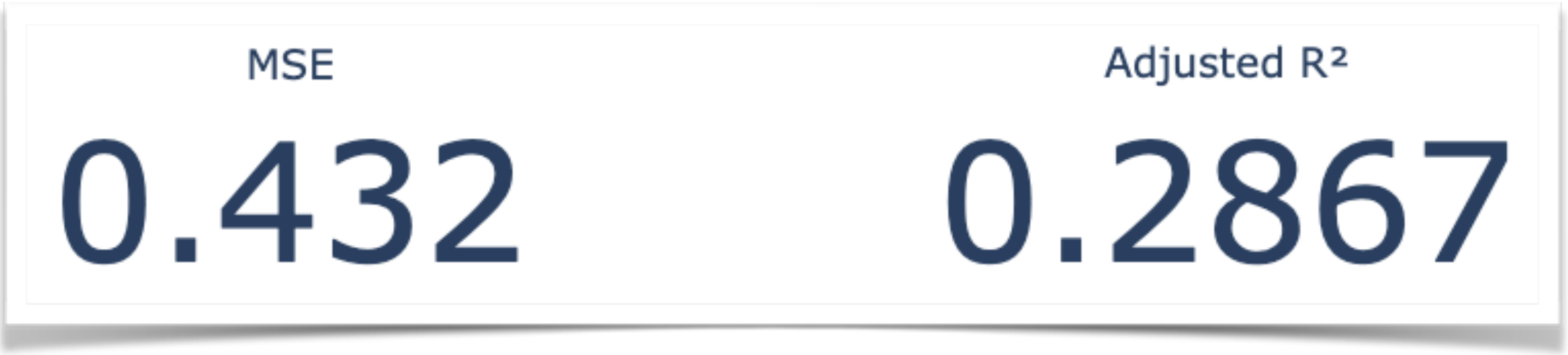
MSE	Adjusted R^2
0.434	0.2844

Regression

Multivariate linear regression with Ridge and Lasso Regularization (1)

- Ridge





MSE

0.432

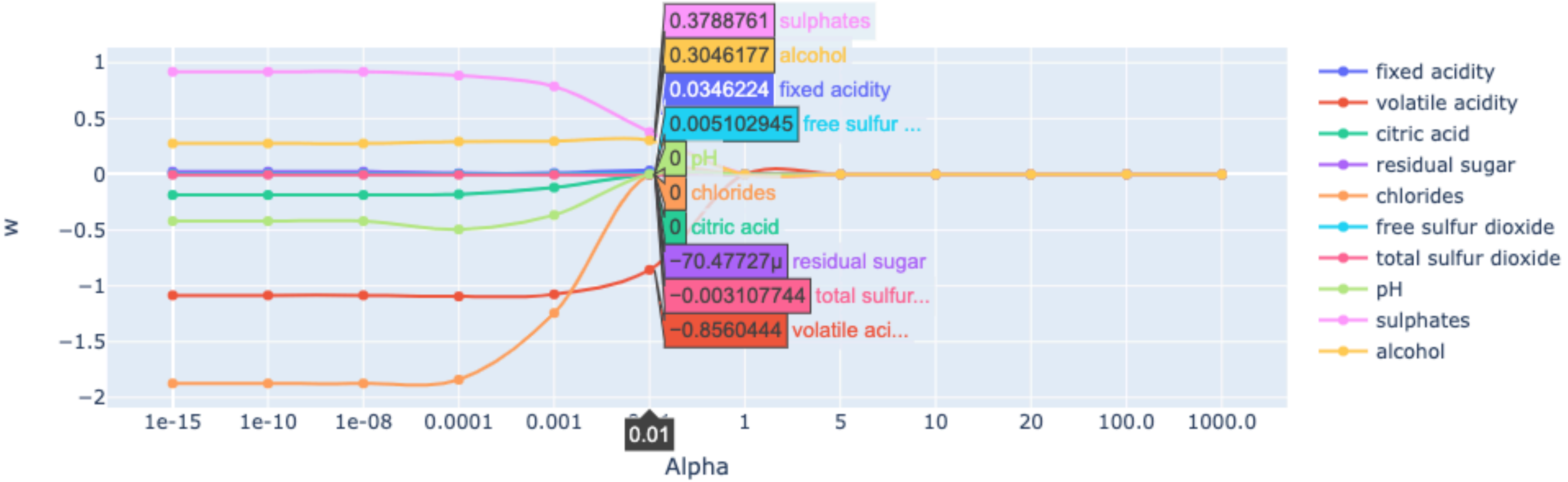
Adjusted R²

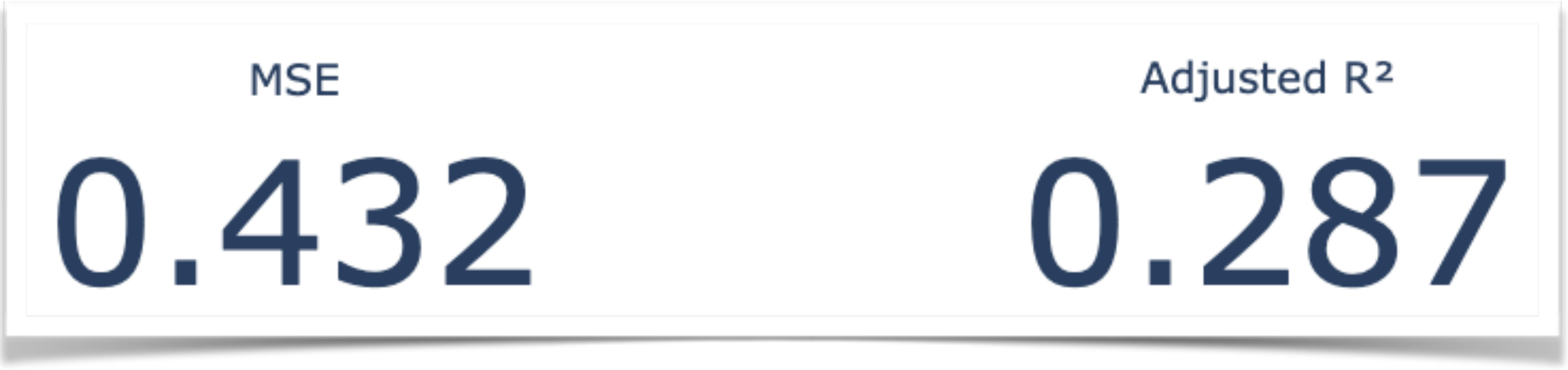
0.2867

Regression

Multivariate linear regression with Ridge and Lasso Regularization (2)

- Lasso





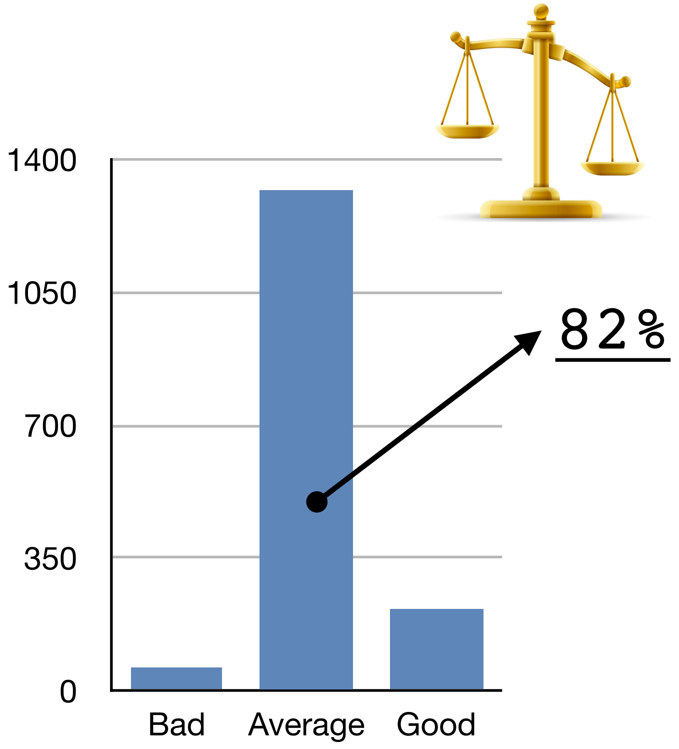
MSE	Adjusted R²
0.432	0.287

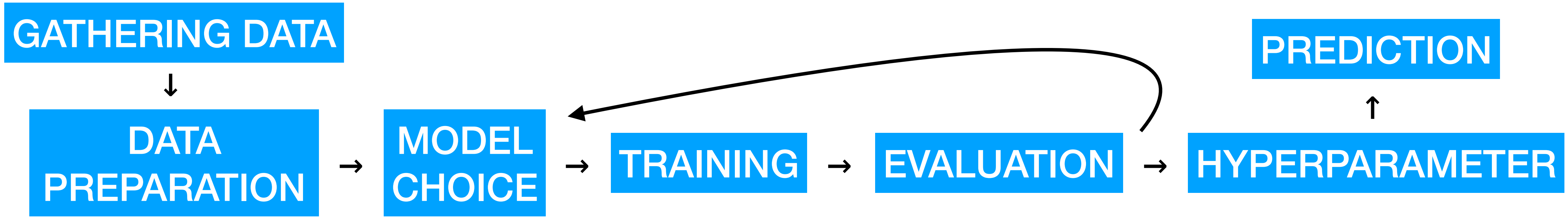
Classification

- Three classes: *Bad*, *Average* and *Good* wine

- Logistic Regression
  K Neighbors Classifier
  Support Vector Classifier (SVC)
  Gaussian Naive Bayes
  Gaussian Process Classifier
  Decision Tree Classifier
  Random Forest Classifier

- Playground:

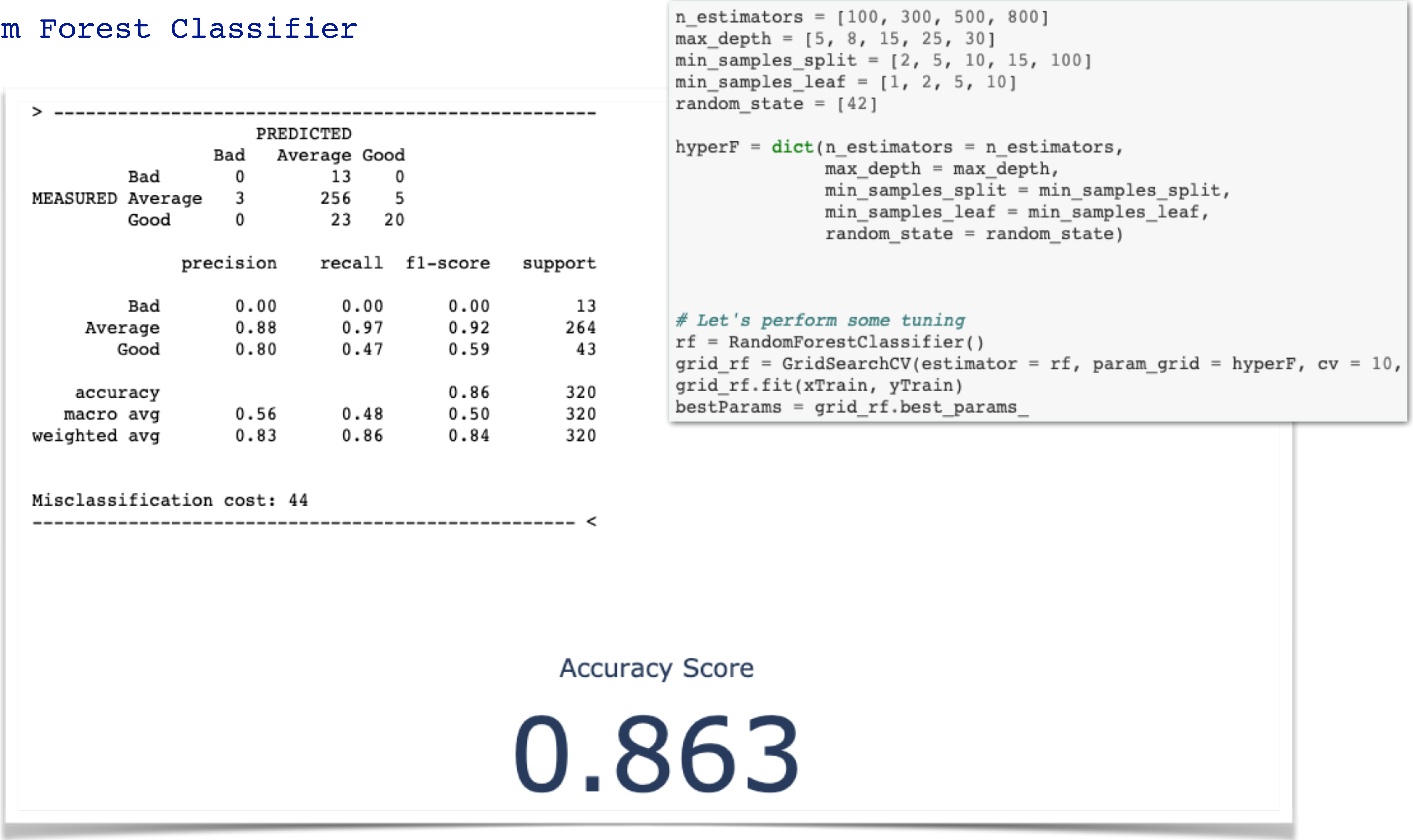




GATHERING DATA ↓ DATA PREPARATION → MODEL CHOICE → TRAINING → EVALUATION → HYPERPARAMETER ↑ PREDICTION

Classification

Random Forest Classifier



```
> ---------------------------------------------------
                   PREDICTED
               Bad   Average Good
         Bad     0        13    0
MEASURED Average 3       256    5
         Good    0        23   20

             precision    recall  f1-score   support

         Bad       0.00      0.00      0.00        13
     Average       0.88      0.97      0.92       264
        Good       0.80      0.47      0.59        43

    accuracy                           0.86       320
   macro avg       0.56      0.48      0.50       320
weighted avg       0.83      0.86      0.84       320


Misclassification cost: 44
--------------------------------------------------- <
```

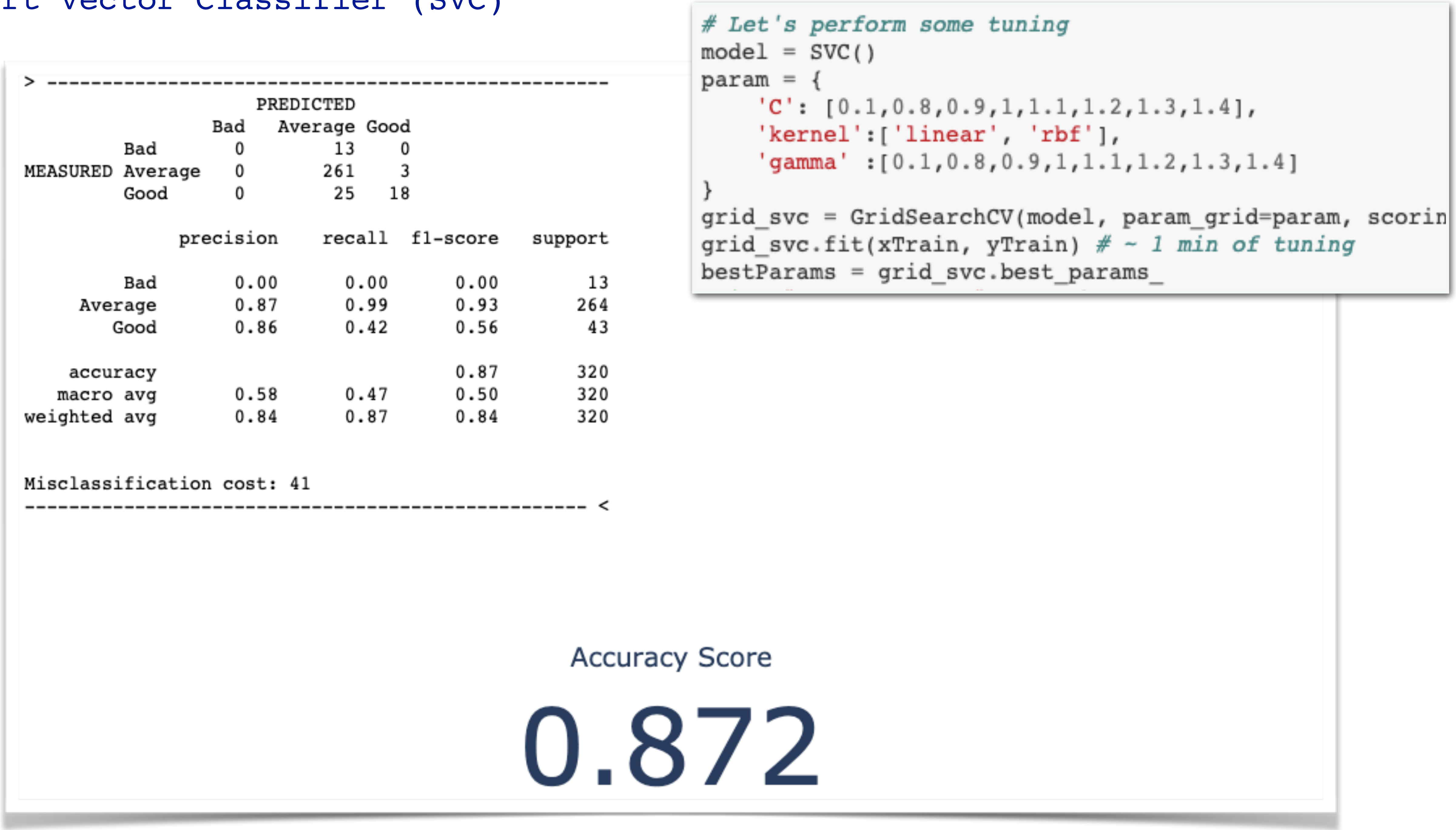
```
n_estimators = [100, 300, 500, 800]
max_depth = [5, 8, 15, 25, 30]
min_samples_split = [2, 5, 10, 15, 100]
min_samples_leaf = [1, 2, 5, 10]
random_state = [42]

hyperF = dict(n_estimators = n_estimators,
              max_depth = max_depth,
              min_samples_split = min_samples_split,
              min_samples_leaf = min_samples_leaf,
              random_state = random_state)


# Let's perform some tuning
rf = RandomForestClassifier()
grid_rf = GridSearchCV(estimator = rf, param_grid = hyperF, cv = 10,
grid_rf.fit(xTrain, yTrain)
bestParams = grid_rf.best_params_
```

Accuracy Score

0.863

Classification

Support Vector Classifier (SVC)



```
> ----------------------------------------------------
                    PREDICTED
                Bad   Average Good
         Bad      0        13    0
MEASURED Average  0       261    3
         Good     0        25   18

              precision    recall  f1-score   support

         Bad       0.00      0.00      0.00        13
     Average       0.87      0.99      0.93       264
        Good       0.86      0.42      0.56        43

    accuracy                           0.87       320
   macro avg       0.58      0.47      0.50       320
weighted avg       0.84      0.87      0.84       320


Misclassification cost: 41
---------------------------------------------------- <
```

Accuracy Score

0.872

```
# Let's perform some tuning
model = SVC()
param = {
     'C': [0.1,0.8,0.9,1,1.1,1.2,1.3,1.4],
     'kernel':['linear', 'rbf'],
     'gamma' :[0.1,0.8,0.9,1,1.1,1.2,1.3,1.4]
}
grid_svc = GridSearchCV(model, param_grid=param, scorin
grid_svc.fit(xTrain, yTrain) # ~ 1 min of tuning
bestParams = grid_svc.best_params_
```

Conclusion

- Quality entries not balanced
- Disappointing results but unrelated data
- Implementation of the topics covered in the course ✓
- Learning methods ✓

https://github.com/nicorbtt/RedWineMachineLearning