# Vintly

The Winemaker’s Quality Predictor

# The Problem

How can winemakers determine whether their wine will be considered high or low quality? In order to

1. Inform Pricing

    Without having a board of professional graders on hand, how can they make informed decisions?

2. Modify Production

    Add more or less of a certain compound for a current harvest

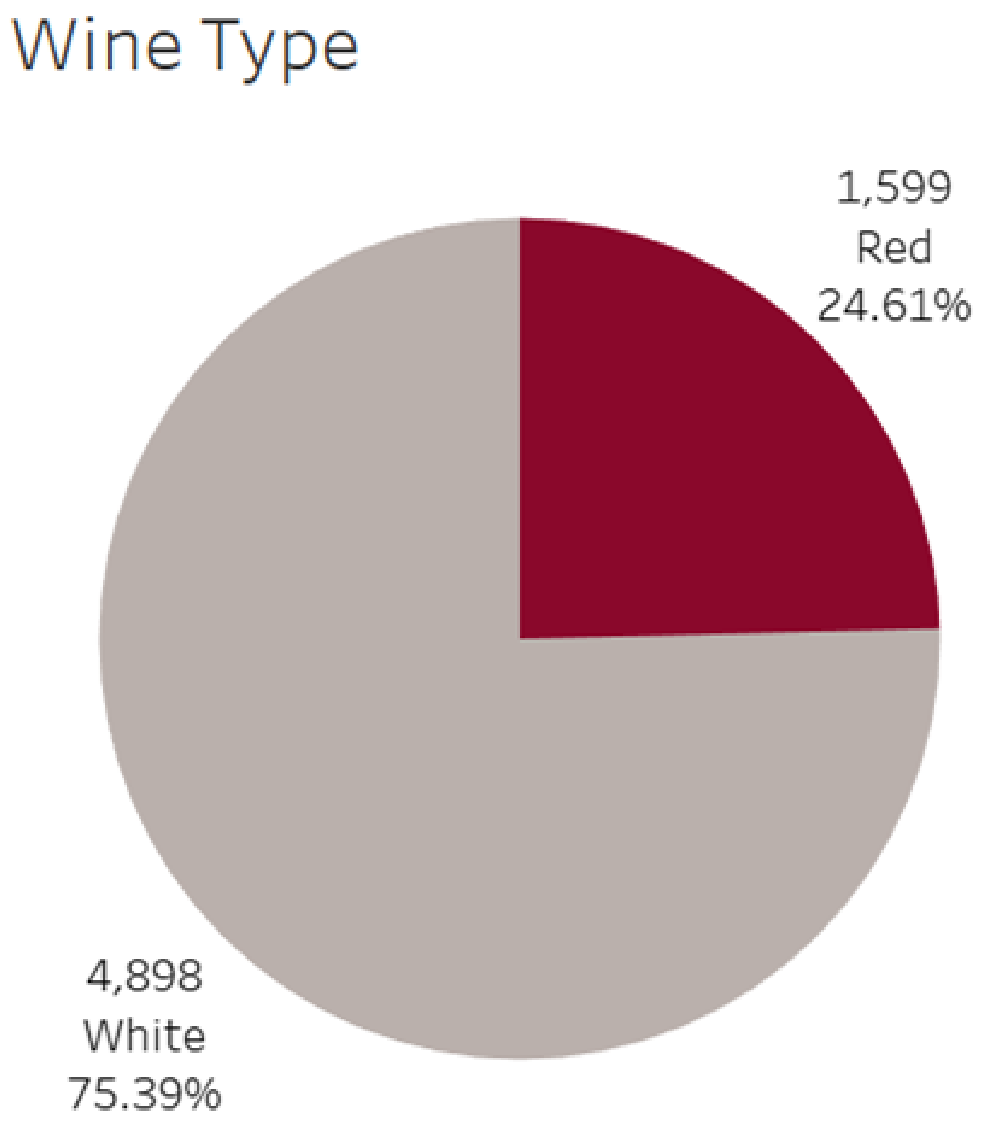
    - Citric acid
    - Chlorides (salt)

    Modify fermentation process for the future harvests

    - Ferment longer for less residual sugar
    - Expose to less oxygen to lower volatile acidity

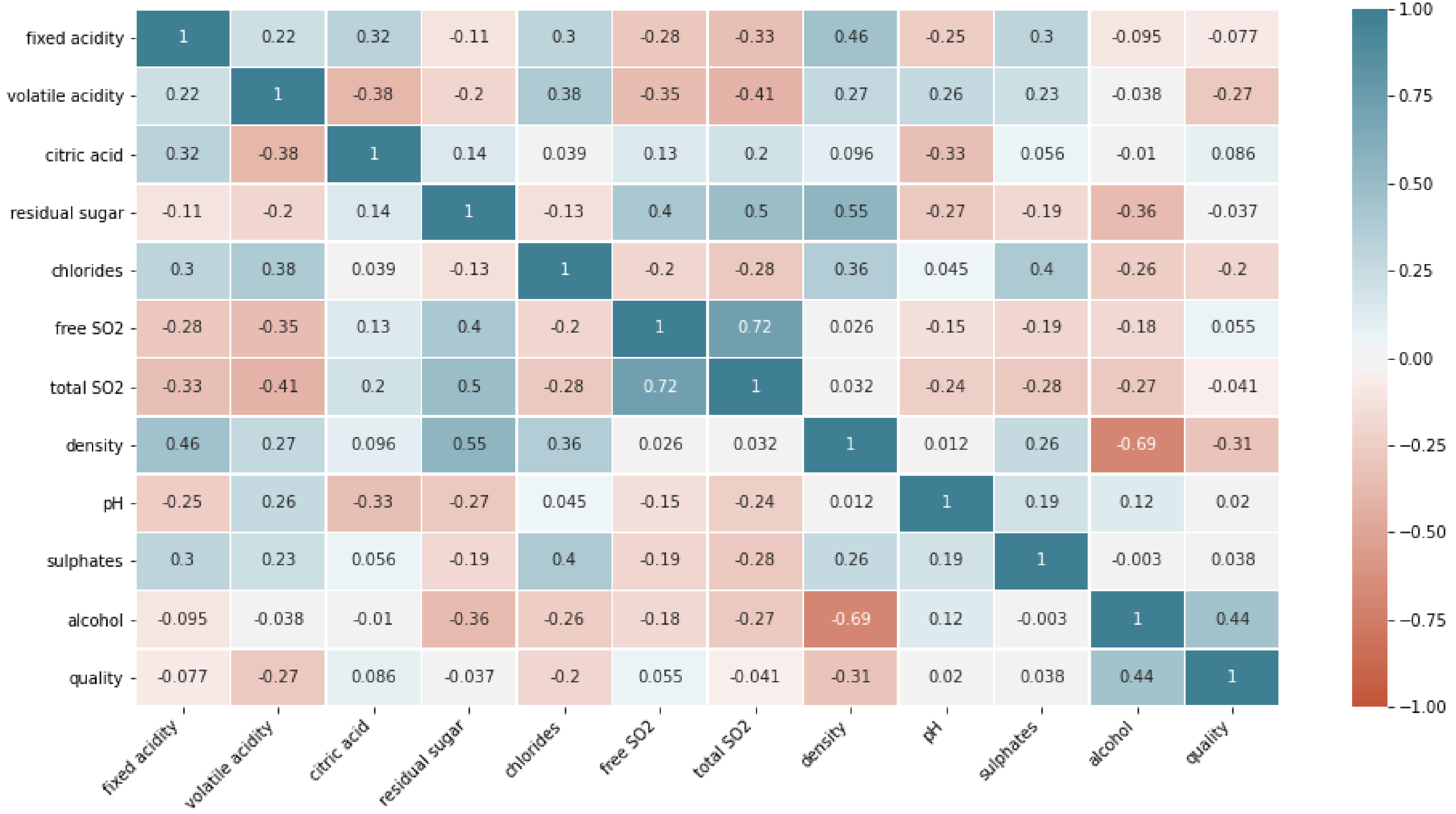
# The Data

- Over 6000 wines
- 11 physiochemical property inputs (Objective)
  - Fixed Acidity
  - Volatile acidity
  - Citric acid
  - Residual sugar
  - Chlorides
  - Free sulfur dioxide
  - Total sulfur dioxide
  - Density
  - pH
  - Sulphates
  - Alcohol
- 1 sensory output-based quality score 0-10 (Subjective)

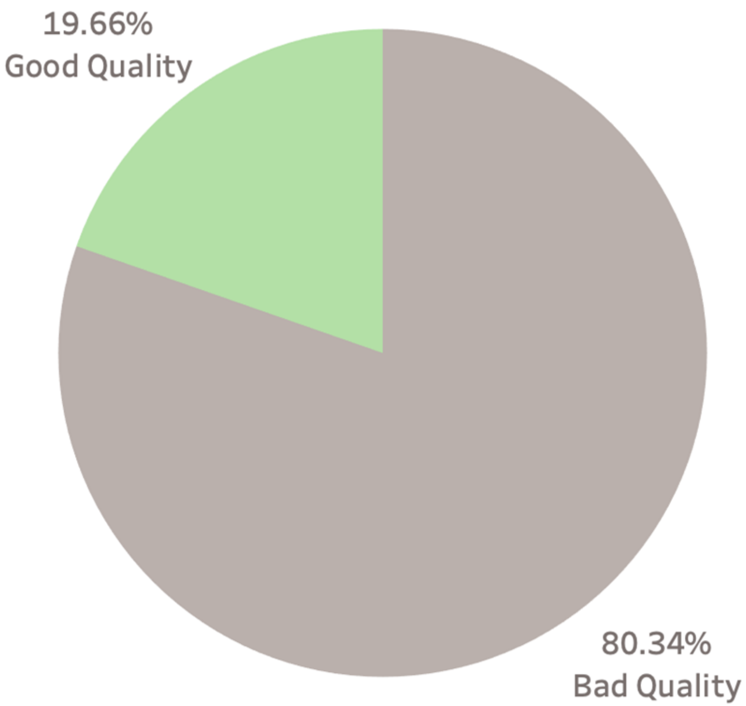



By the courtesy of:
P. Cortez, A. Cerdeira, F. Almeida, T. Matos and J. Reis. Modeling wine preferences by data mining from physicochemical properties. In Decision Support Systems, Elsevier, 47(4):547-553. ISSN: 0167-9236.

# Correlation Matrix



# The Challenge




**Class Imbalance**

Classification!

1. Quality score was sensory based therefore individual linear scores were less informative than a binary classification.
2. Good wines are always hard to find and are the minority. Imbalanced distribution of wine qualities makes classification, difficult.
   - False positives present in grading make our clients lose **reputation**
   - False negatives make them lose **money**

**It's always the trade-off faced when we boost the classification to either have less false positives or false negatives.**

# The Solutions

We developed three models with three different metrics each to allow a winemaker to evaluate their business goals and select the one most suited to them.

Models:

- Deep Learning
- Logistic Regression
- Scaled Vector Machine

Metrics:

- Precision
- Recall
- Accuracy

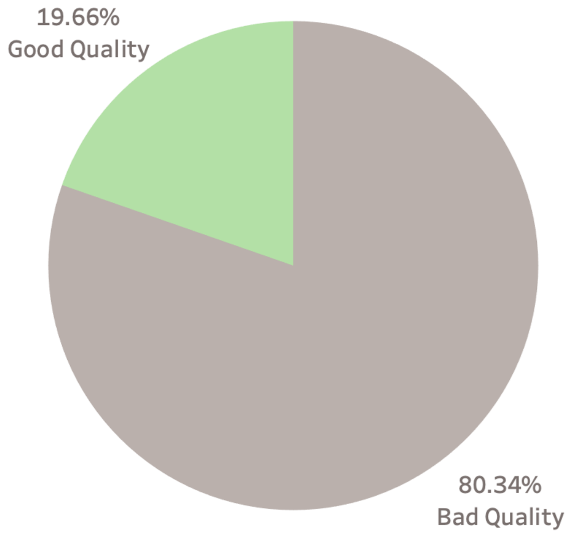
# Metrics

<u>Accuracy</u>

$$Accuracy = \frac{Number\ of\ Correct\ predictions}{Total\ number\ of\ predictions\ made}$$

The simplest of metric, accuracy works well when there are equal number of samples belonging to each class.

For our dataset, accuracy would be inflated to favor the majority class, bad wines, correctly labelling them more often than good wine correctly.



# Metrics

## Precision

**Also called Positive predictive value.**

The ratio of correct positive predictions to the total predicted positives.

Precision is a good measure when the cost of False Positive is high.

If a Bad Quality wine is incorrectly classified as Good Quality,

### Reputation is at stake.

## Recall

**Also called Sensitivity, Probability of Detection, True Positive Rate.**

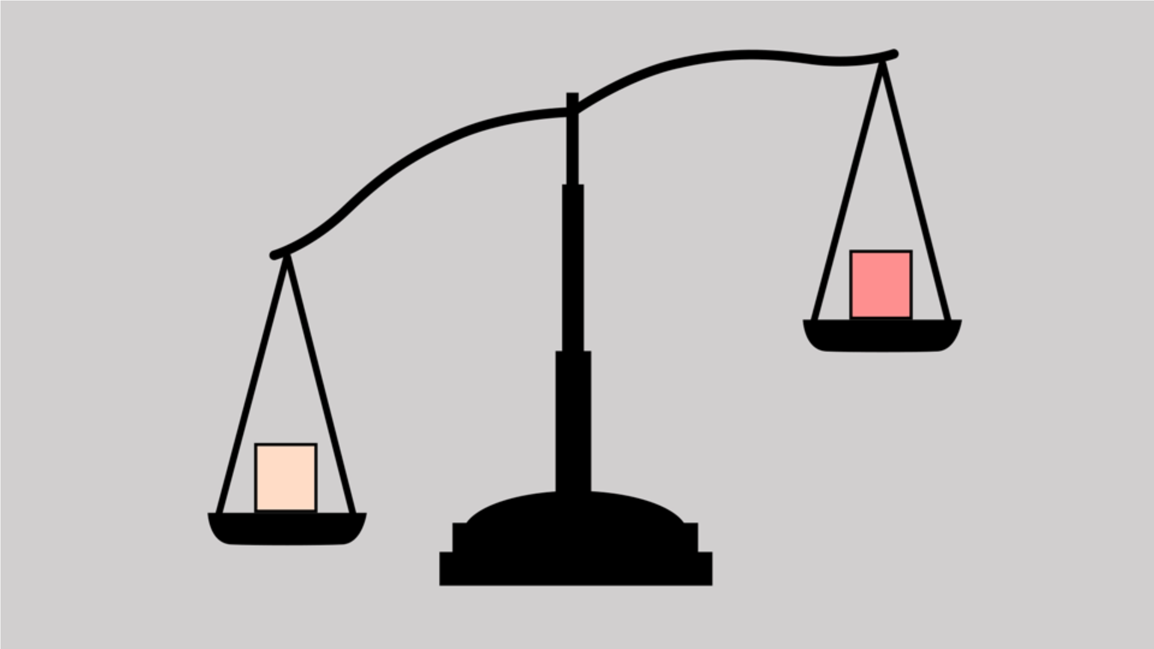
The ratio of correct positive predictions to the total positives examples.

Recall is a good measure when the cost of False Negative is high.

If a Good Quality wine is incorrectly classified Bad Quality,

### Money is at stake.

# False Negative vs False Positive

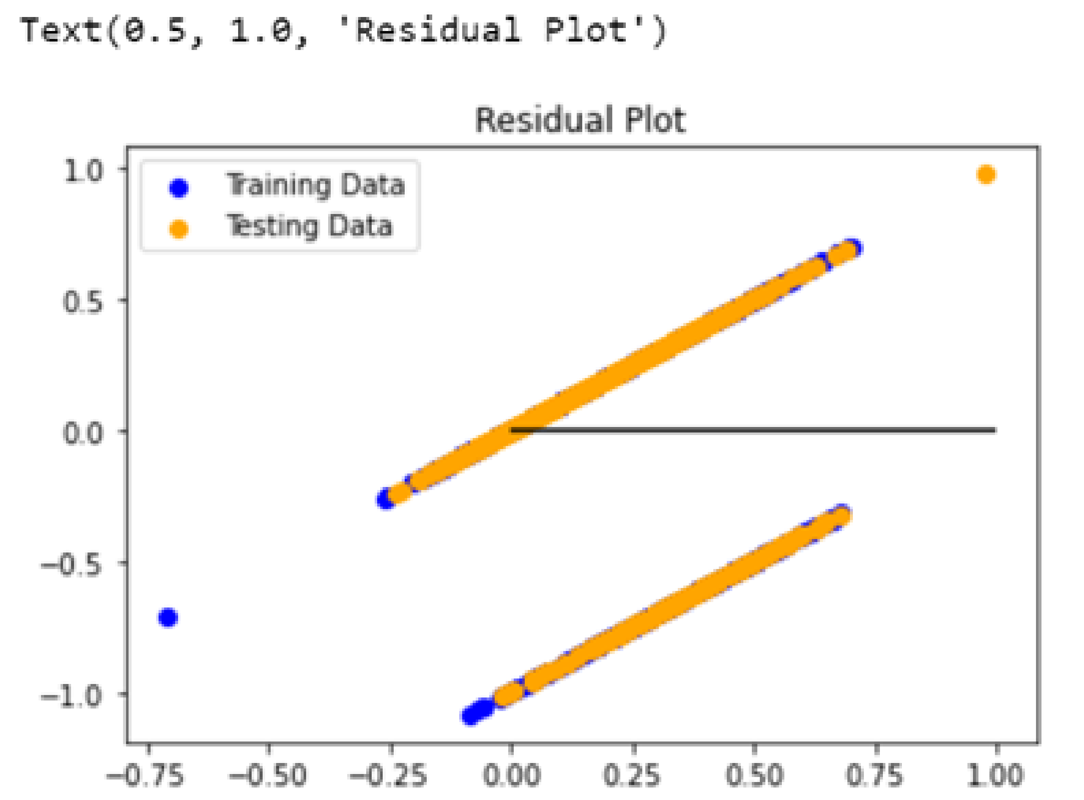


Falsely classifying a good wine as bad might result in losing money.

Falsely classifying a bad wine as good would hurt reputation. (Ultimately hurting a higher end brand that would result in overall loss of money).

# Multi-V Linear Model

- Given that the outcome of our data was linear, on a basis of 0-10 in quality ranking, we initially tried a multivariate linear regression.
- Quality was based on sensory input, we found that a multivariate linear model was not accurate, and determined that a logistic model of “good quality” vs “bad quality could be more effective.
- With the knowledge that what is subjectively considered a “good” wine vs “bad” in terms of pricing, we created bins for for everything above average being classified as “good” and average or below classified as “bad”.



# Deep Learning

```
#Dividing wine as good and bad by giving the limit for the quality
bins = (2, 6.5, 9)
group_names = ['bad', 'good']
df['quality'] = pd.cut(df['quality'], bins = bins, labels = group_names)
```

```
# Train the Model
model = Sequential()
model.add(Dense(units=200, activation='relu', input_dim=12))
model.add(Dense(units=200, activation='relu'))
# model.add(Dense(units=2, activation='sigmoid'))
model.add(Dense(units=2, activation='softmax'))
```

## Model Summary

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 200)               2600
_________________________________________________________________
dense_1 (Dense)              (None, 200)               40200
_________________________________________________________________
dense_2 (Dense)              (None, 2)                 402
=================================================================
Total params: 43,202
Trainable params: 43,202
Non-trainable params: 0
_________________________________________________________________
```

# *What is Smote?*

**Synthetic Minority Over-sampling Technique (SMOTE) is a machine learning algorithm that resolves class imbalance problem, such as when one class in the training set dominates the other.**

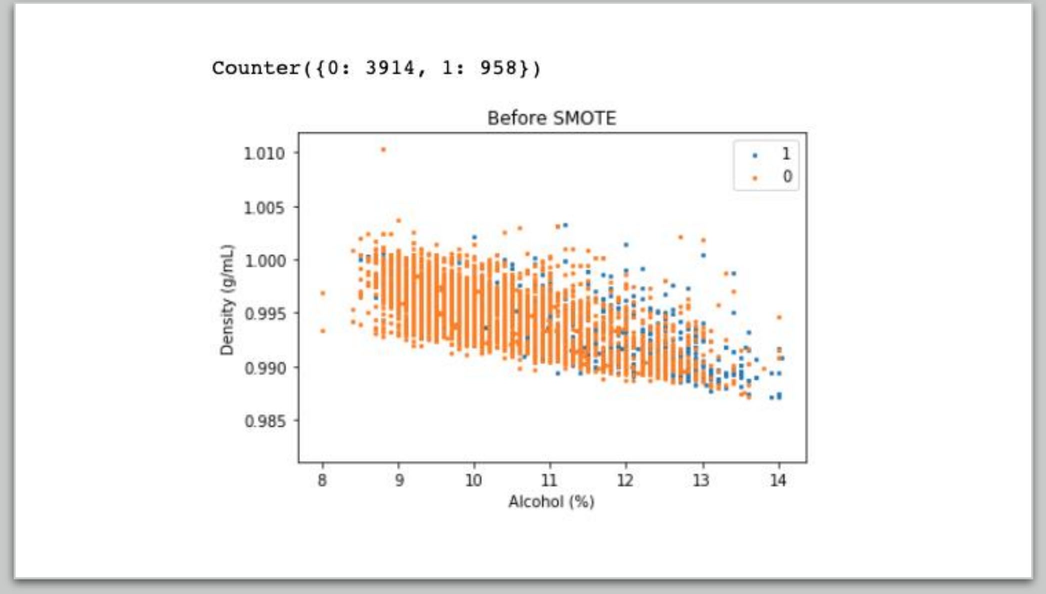
**Our data before SMOTE:**

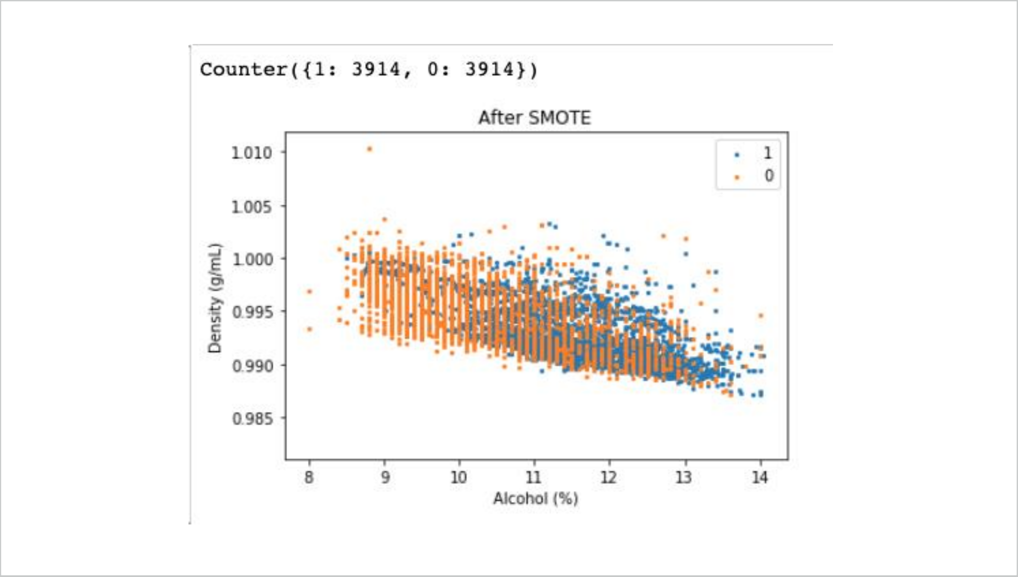
- 958 Good Quality Wines (minority)
- 3914 Bad Quality Wines (majority)

**Our data after SMOTE:**

- 3914 Good Quality Wines (minority)
- 3914 Bad Quality Wines (majority)

SMOTE imagines new, synthetic minority instances to balance the imbalance out.



# Logistic Regression

## Before SMOTE

| | precision | recall | f1-score |
|---|---|---|---|
| 0 | 0.84 | 0.95 | 0.89 |
| 1 | 0.54 | 0.24 | 0.33 |
| accuracy | | | 0.81 |
| macro avg | 0.69 | 0.59 | 0.61 |
| weighted avg | 0.78 | 0.81 | 0.78 |

## After SMOTE

| | precision | recall | f1-score |
|---|---|---|---|
| 0 | 0.92 | 0.71 | 0.80 |
| 1 | 0.39 | 0.76 | 0.51 |
| accuracy | | | 0.72 |
| macro avg | 0.66 | 0.73 | 0.66 |
| weighted avg | 0.82 | 0.72 | 0.74 |

Overall accuracy is slightly worse after SMOTE, however it was originally weighted to favor correct classification of bad wines and incorrect classification of good wines, due to the imbalance of samples in each class.

After SMOTE, good wine recall is dramatically improved, making it a better model for those more interested in not losing money by accidentally classifying a good wine as bad, as opposed to not hurting their reputation by classifying a bad wine as good.

SMOTE improves the overall accuracy of the Logistic Regression model by identifying the quality of wines, with improved accuracy in identifying the good wines as good = **Better pricing potential**

# Support Vector Machine

SVM address class imbalance by assigning different weights to positive and negative instances.

GridCV is part of the sklearn library, that automates the process of performing hyper parameter tuning in order to determine the optimal values for a given model.

Before GridCV

| | precision | recall | f1-score |
|---|---|---|---|
| 0 | 0.85 | 0.97 | 0.90 |
| 1 | 0.68 | 0.29 | 0.41 |
| accuracy | | | 0.83 |
| macro avg | 0.76 | 0.63 | 0.65 |
| weighted avg | 0.82 | 0.83 | 0.81 |

After GridCV

| | precision | recall | f1-score |
|---|---|---|---|
| 0 | 0.88 | 0.96 | 0.92 |
| 1 | 0.72 | 0.46 | 0.56 |
| accuracy | | | 0.86 |
| macro avg | 0.80 | 0.71 | 0.74 |
| weighted avg | 0.85 | 0.86 | 0.85 |

# Confusion Matrix

| | | Model Prediction | |
|---|---|---|---|
| | | Bad Quality (0) | Good Quality (1) |
| **Actual Wine Quality** | Bad Quality (0) | **True Negative** Correctly predicted Bad Quality | **False Positive** Incorrectly predicted Good Quality |
| | Good Quality (1) | **False Negative** Incorrectly predicted Bad Quality | **True Positive** Correctly predicted Good Quality |

# Testing Results

| **Deep Learning** (n=1,625) | Predicted: (Bad Quality) | Predicted: (Good Quality) |
|---|---|---|
| Actual: (Bad Quality) | 1,244 (77%) | 62 (3%) |
| Actual: (Good Quality) | 211 (13%) | 108 (7%) |

| **SVM** (n=1,625) | Predicted: (Bad Quality) | Predicted: (Good Quality) |
|---|---|---|
| Actual: (Bad Quality) | 1,269 (78%) | 37 (2%) |
| Actual: (Good Quality) | 173 (11%) | 146 (9%) |

| **Logistic Regression (SMOTE)** (n=1,625) | Predicted: (Bad Quality) | Predicted: (Good Quality) |
|---|---|---|
| Actual: (Bad Quality) | 924 (57%) | 382 (23%) |
| Actual: (Good Quality) | 76 (5%) | 243 (15%) |

SVM with GridCV

| | precision | recall | f1-score |
|---|---|---|---|
| 0 | 0.88 | 0.96 | 0.92 |
| 1 | 0.72 | 0.46 | 0.56 |
| accuracy | | | 0.86 |
| macro avg | 0.80 | 0.71 | 0.74 |
| weighted avg | 0.85 | 0.86 | 0.85 |

Deep Learning

| | precision | recall | f1-score |
|---|---|---|---|
| 0 | 0.86 | 0.94 | 0.90 |
| 1 | 0.63 | 0.39 | 0.48 |
| accuracy | | | 0.84 |
| macro avg | 0.75 | 0.67 | 0.69 |
| weighted avg | 0.82 | 0.84 | 0.82 |

Logistic with SMOTE

| | precision | recall | f1-score |
|---|---|---|---|
| 0 | 0.93 | 0.71 | 0.80 |
| 1 | 0.39 | 0.76 | 0.52 |
| accuracy | | | 0.72 |
| macro avg | 0.66 | 0.74 | 0.66 |
| weighted avg | 0.82 | 0.72 | 0.75 |

SVM had highest accuracy, (the best balance of Recall and Precision) and the best Precision. This model is better for those concerned with balance or reputation, (don’t want to sell bad wine labeled as good)

Logistic with SMOTE had the best Recall, but lowest Precision, better for preventing labeling of good wine as bad, therefore preventing loss of money.

# ***REPUTATION VS PRICE-RISK BENEFITS***

# QUESTIONS


LIFE IS TOO
SHORT TO
DRINK BAD
WINE
MONTGRAS
WWW.MONTGRAS.CL
FAMILIA DEICAS
NIETO SENETINER