K NEAREST NEIGHBORS

Projects: 1) Identifying Wine Color and 2) Optical Character Recognition

BEFORE WE BEGIN LET US DO A THOUGHT EXPERIMENT

I want to find somebody to spend a Saturday afternoon with and I am looking for somebody most similar to me (nearest neighbor) in terms of:

- Sex (coded as 0 for female, and 1 for male)
- Age (coded in years)
- Outdoor sports interest (coded from 0 (no interest) to 10 (enthusiast))»

(all categories matter the same to me)

LET US DO THE CALCULATION FOR A SIMILARITY SCORE

(AVERAGE ABSOLUTE DIFFERENCES)

Sake of argument: I am male (**1**), **50** years, outdoor sports score **7**:

- first candidate a student
  .
- second candidate an athletic outdoor (score=**9**) women (**0**) **51** years old
- third candidate an athletic outdoor (score=**9**), man (**1**) **53** years

LET US DO THE CALCULATION FOR A SIMILARITY SCORE

(AVERAGE ABSOLUTE DIFFERENCES – NORMALIZED TO 0 – 10)

Sake of argument: I am male (**1**), **50** years, outdoor sports score **7**:

- first candidate a student
  .
- second candidate an athletic outdoor (score=**9**) women (**0**) **51** years old
- third candidate is an athletic outdoor (score=**9**) man (**1**) **53** years»

OVERWIEW

In this session you will learn:

1. What is the underlying **idea of k-Nearest Neighbors**
2. How similarity can be measured with **Euclidean distance**
3. Why **scaling predictor variables** is important for some machine learning models
4. Why the **tidymodels package** makes it easy to work with machine learning models
5. How you can define a **recipe** to pre-process data with the `tidymodels` package
6. How you can define a **model-design** with the `tidymodels` package
7. How you can create a machine learning **workflow** with the `tidymodels` package
8. How **metrics** derived from a **confusion matrix** can be used to asses prediction quality
9. Why you have to be careful when interpreting *accuracy*, when you work with **unbalanced observations**
0. How a machine learning model can **process images** and how OCR (Optical Character Recognition) works»

ABOUT THE WINE DATASET

We will work with a publicly available wine dataset[1] containing 3,198 observations about different wines and their chemical properties.

Our goal is to develop a k-Nearest Neighbors model that can predict if a wine is red or white based on the wine's chemical properties.»

1. Cortez, Paulo, António Cerdeira, Fernando Almeida, Telmo Matos, and José Reis. 2009. "Modeling Wine Preferences by Data Mining from Physicochemical Properties." Decision Support Systems 47 (4): 547–53. https://doi.org/10.1016/j.dss.2009.05.016

RAW OBSERVATIONS FROM WINE DATASET

```
1 library(rio)
2 DataWine=import("https://lange-analytics.com/AIBook/Data/WineData.rds")
3 print(DataWine)
```

```
# A tibble: 3,198 × 13
   wineC…¹ acidity volat…² citri…³ resid…⁴ Chlor…⁵ freeS…⁶ total…⁷ Density    pH
   <chr>     <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl> <dbl>
 1 red        10.8   0.32     0.44     1.6   0.063      16      37   0.998  3.22
 2 white       6.4   0.31     0.39     7.5   0.04       57     213   0.995  3.32
 3 white       9.4   0.28     0.3      1.6   0.045      36     139   0.995  3.11
 4 white       8.2   0.22     0.36     6.8   0.034      12      90   0.994  3.01
 5 white       6.4   0.29     0.44     3.6   0.197      75     183   0.994  3.01
 6 red         6.7   0.855    0.02     1.9   0.064      29      38   0.995  3.3
 7 red        11.8   0.38     0.55     2.1   0.071       5      19   0.999  3.11
 8 white       6.7   0.25     0.23     7.2   0.038      61     220   0.995  3.14
 9 red         7.5   0.38     0.57     2.3   0.106       5      12   0.996  3.36
10 red         7.1   0.27     0.6      2.1   0.074      17      25   0.998  3.38
# … with 3,188 more rows, 3 more variables: sulphates <dbl>, alcohol <dbl>,
#   quality <dbl>, and abbreviated variable names ¹wineColor, ²volatileAcidity,
#   ³citricAcid, ⁴residualSugar, ⁵Chlorides, ⁶freeSulfurDioxide,
#   ⁷totalSulfurDioxide
```

»

OBSERVATIONS FROM WINE DATASET FOR SELECTED VARIABLES

SULFOR DIOXIDE AND ACIDITY

Note we use `clean_names("upper_camel")` from the `janitor` package to change all column (variable) names to UpperCamel.

```
library(tidyverse); library(rio);library(janitor)
DataWine=import("https://lange-analytics.com/AIBook/Data/WineData.rds") %>%
  clean_names("upper_camel") %>%
  select(WineColor,Sulfur=TotalSulfurDioxide,Acidity) %>%
  mutate(WineColor=as.factor(WineColor))
print(DataWine)
```

```
# A tibble: 3,198 × 3
   WineColor Sulfur Acidity
   <fct>      <dbl>   <dbl>
 1 red           37    10.8
 2 white        213     6.4
 3 white        139     9.4
 4 white         90     8.2
 5 white        183     6.4
 6 red           38     6.7
 7 red           19    11.8
 8 white        220     6.7
 9 red           12     7.5
```

```
10 red            25     7.1
# … with 3,188 more rows
```

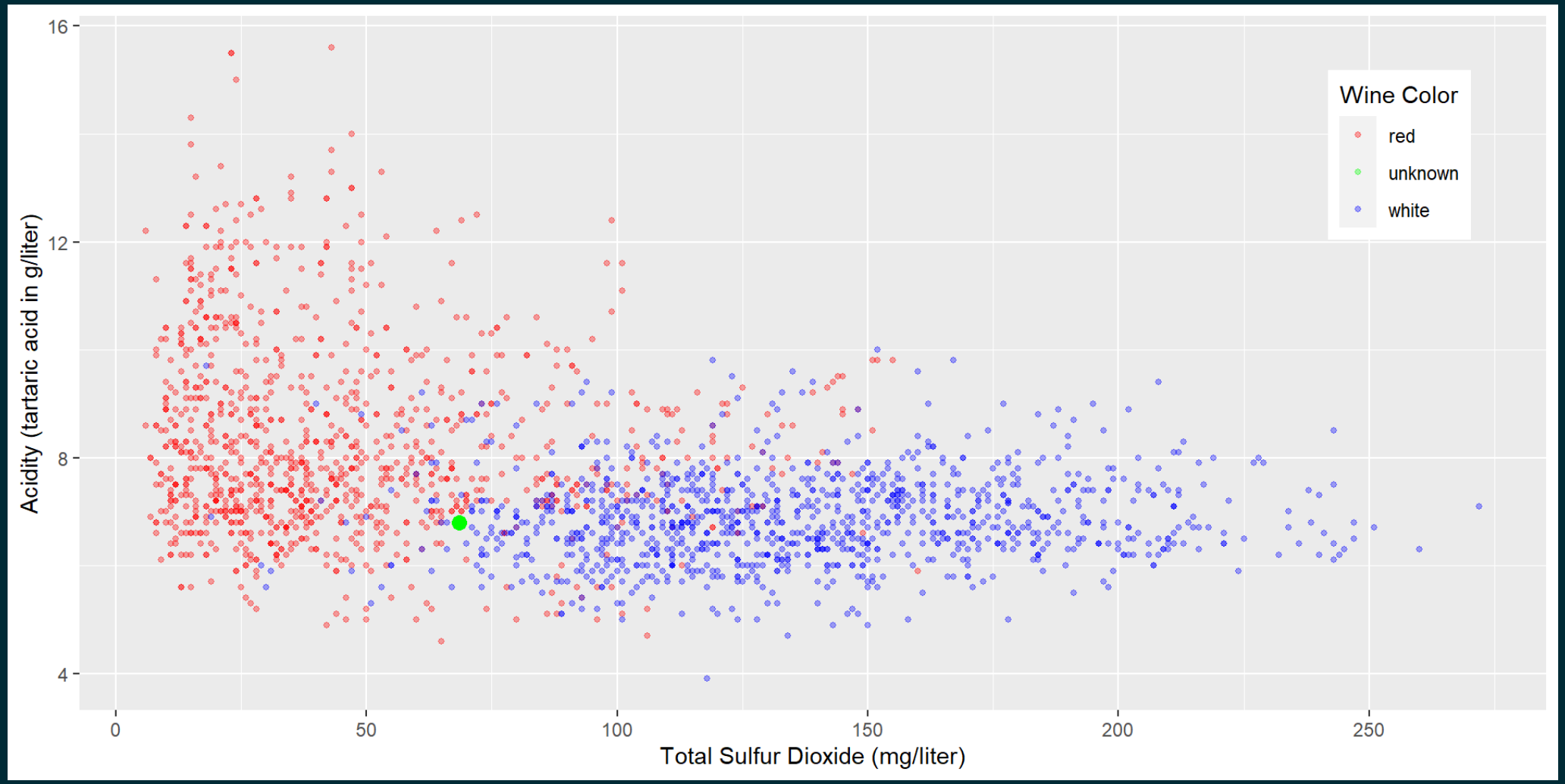
»

BEFORE STARTING WITH K NEAREST NEIGHBORS

LET US FIND SOME EYEBALLING TECHNIQUES THAT ARE RELATED TO VARIOUS MACHINE LEARNING MODELS»

EYE BALLING TECHNIQUES TO IDENTIFY RED AND WHITE WINES

TRY EYEBALLING THE DATA

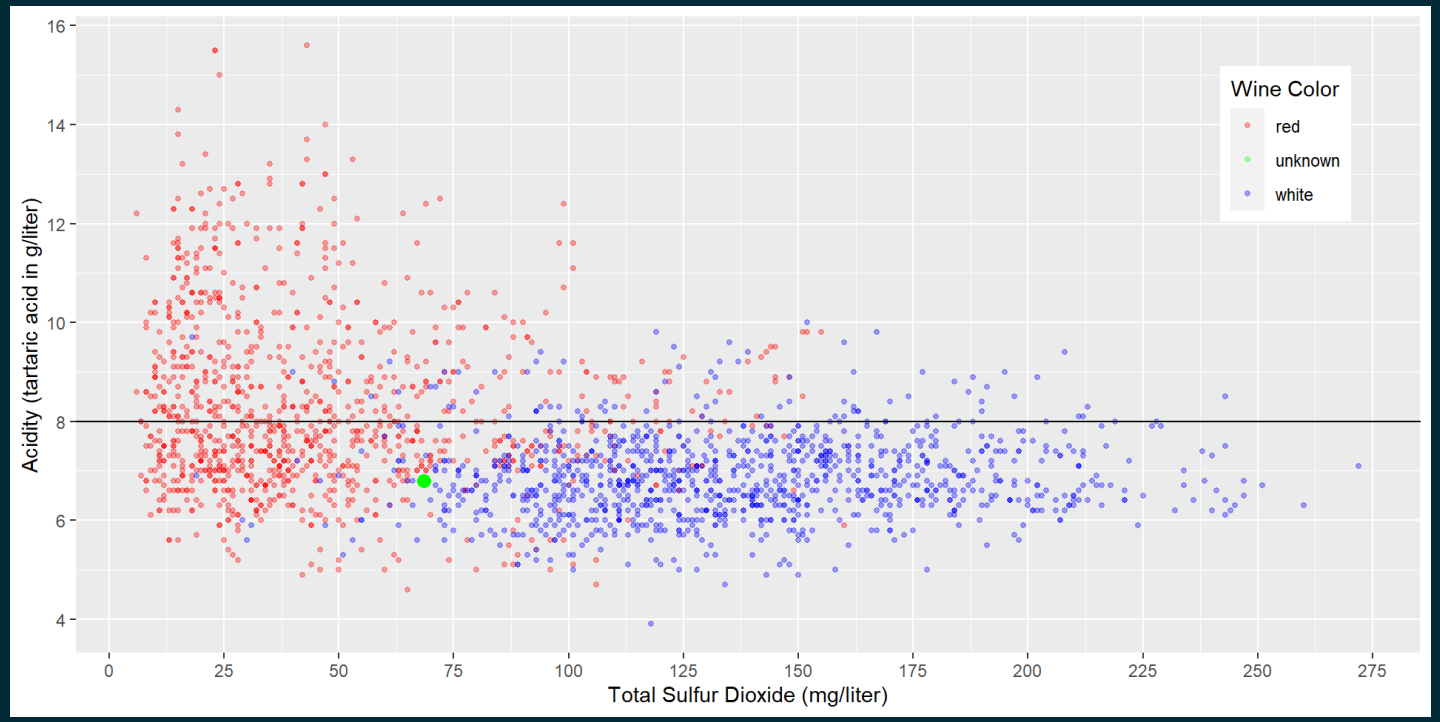


Acidity and Total Sulfur Dioxide Related to Wine Color

https://econ.lange-analytics.com/aibook/

EYE BALLING TECHNIQUES TO IDENTIFY RED AND WHITE WINES

HORIZONTAL BOUNDARY



Horizontal Decision Boundary for Acidity and Total Sulfur Dioxide Related to Wine Color

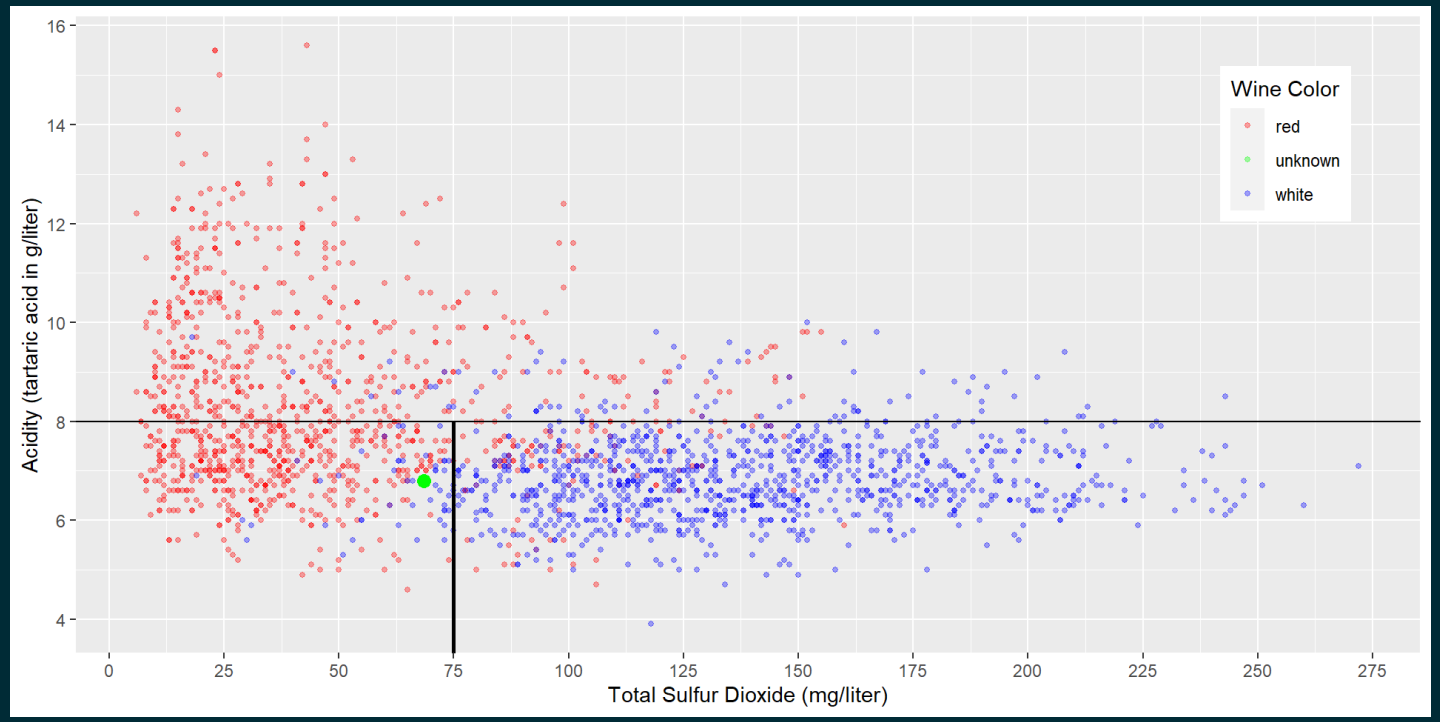
CONFUSION MATRIX

	Truth	
Prediction	Red Wine	White Wine
Red Wine	TP: 'half'	FP: 'few'
White Wine	FN: 'half'	TN: 'most'

https://econ.lange-analytics.com/aibook/

EYEBALLING TECHNIQUES TO IDENTIFY RED AND WHITE WINES

CREATING SUBSPACES LIKE SIMILAR TO A DECISION TREE



Sub-Space Boundaries for Acidity and Total Sulfur Dioxide Related to Wine Color

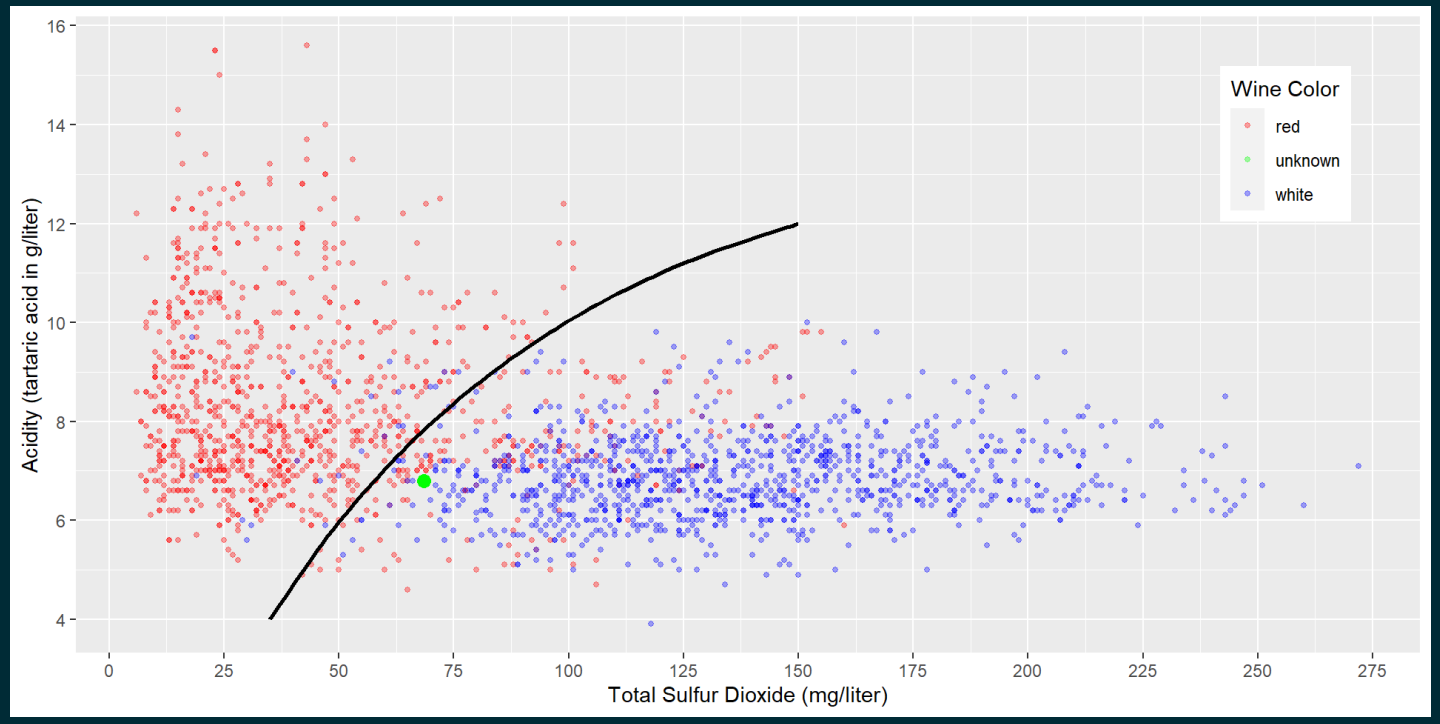
CONFUSION MATRIX

	Truth	
Prediction	Red Wine	White Wine
Red Wine	TP: 'most'	FP: 'few'
White Wine	FN: 'few'	TN: 'most'

https://econ.lange-analytics.com/aibook/

EYEBALLING TECHNIQUES TO IDENTIFY RED AND WHITE WINES

USING A NON-LINEAR DECISION BOUNDARY LIKE A NEURAL NETWORK



Curved Decision Boundary for Acidity and Total Sulfur Dioxide Related to Wine Color

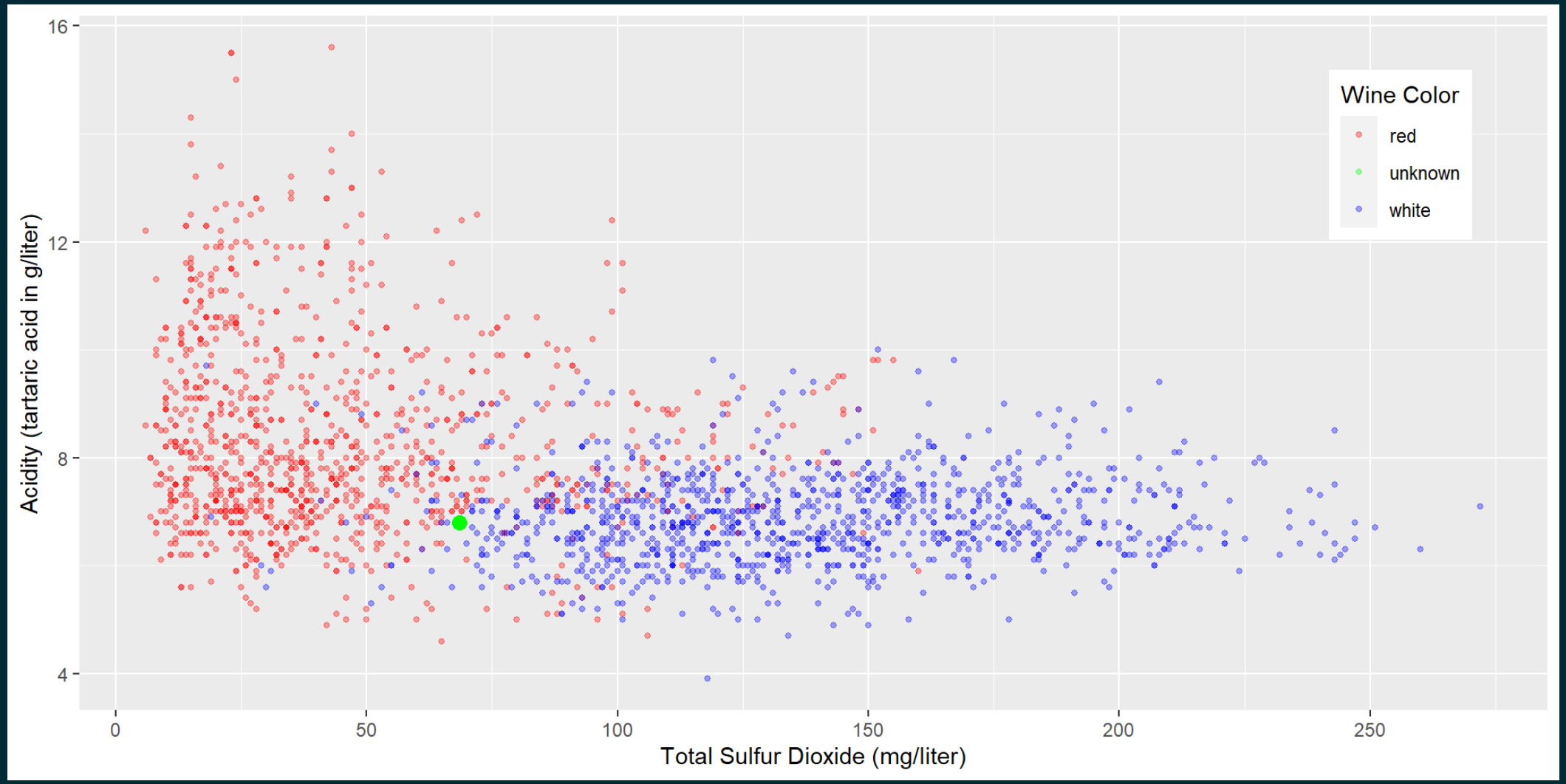
CONFUSION MATRIX

	Truth	
Prediction	Red Wine	White Wine
Red Wine	TP: 'most'	FP: 'few'
White Wine	FN: 'few'	TN: 'most'

https://econ.lange-analytics.com/aibook/

SO, HOW DOES K NEAREST NEIGHBORS WORK?

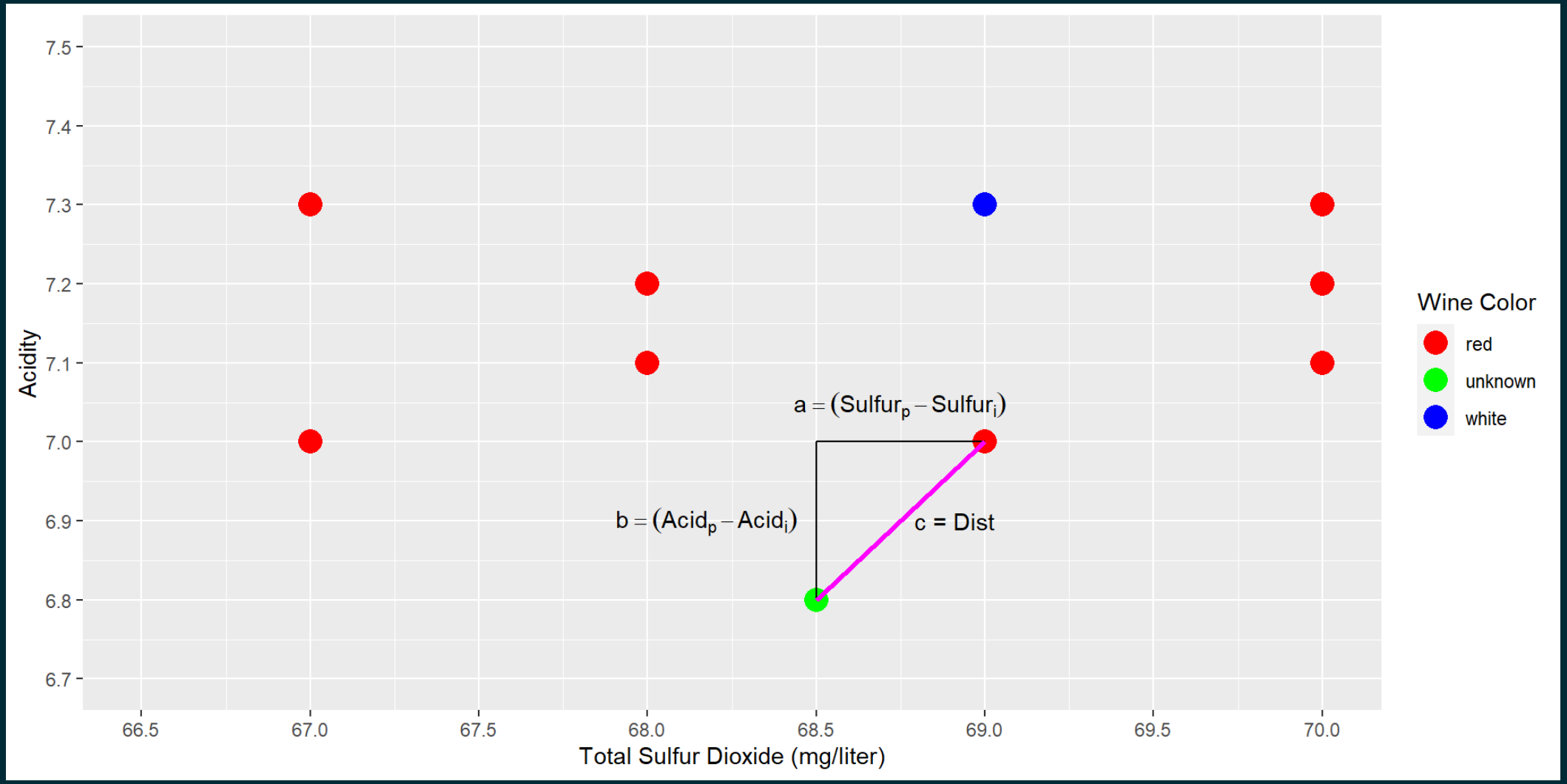
K NEAREST NEIGHBORS K=1



Acidity and Total Sulfur Dioxide Related to Wine Color»

https://econ.lange-analytics.com/aibook/

K NEAREST NEIGHBORS K=1



Predicting Wine Color with k-Nearest Neighbors (k=1)

https://econ.lange-analytics.com/aibook/

HOW TO CALCULATE EUCLIDEAN DISTANCE FOR TWO VARIABLES

Assume our observations have **two predictor variables** x and y. We compare the unknown point p to one of the points from the training data (e,g., point i):

$$Dist_i = \sqrt{(x_p - x_i)^2 + (y_p - y_i)^2}$$

»

HOW TO CALCULATE EUCLIDEAN DISTANCE FOR THREE VARIABLES

Assume our observations have **three predictor variables** x, y, and z. We compare the unknown point p to one of the points from the training data (e,g., point i):

$$Dist_i = \sqrt{(x_p - x_i)^2 + (y_p - y_i)^2 + (z_p - z_i)^2}$$

»

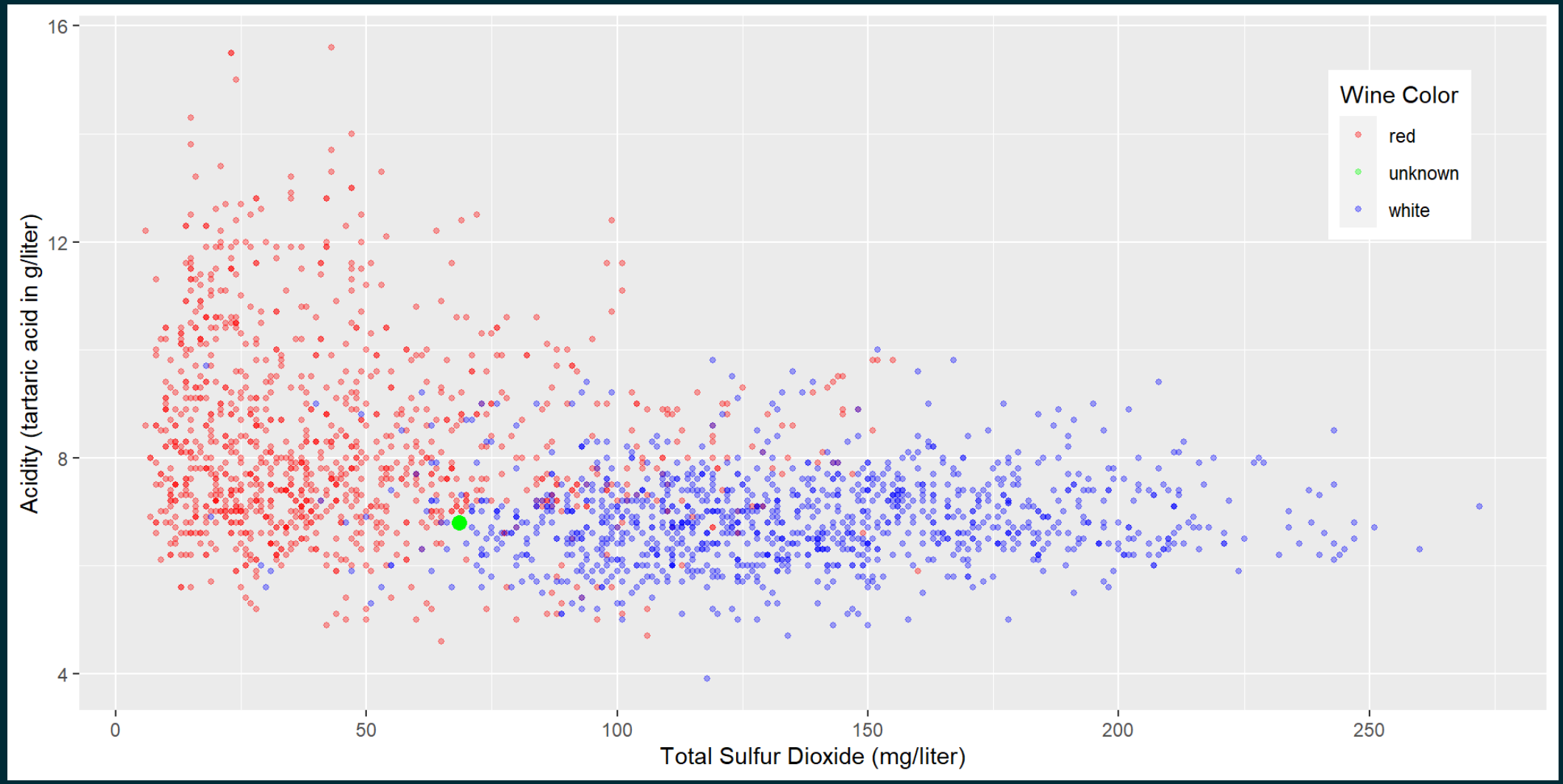
HOW TO CALCULATE EUCLIDEAN DISTANCE FOR N VARIABLES

Assume our observations have N predictor variables v_j with $j = 1 \ldots N$. We compare the unknown point p to one of the points from the training data (e,g., point i):

$$Dist_i = \sqrt{\sum_{j=1}^{N} (v_{p,j} - v_{i,j})^2}$$

»

K NEAREST NEIGHBORS K=4 (FOR A DIFFERENT UNKNOWN WINE)

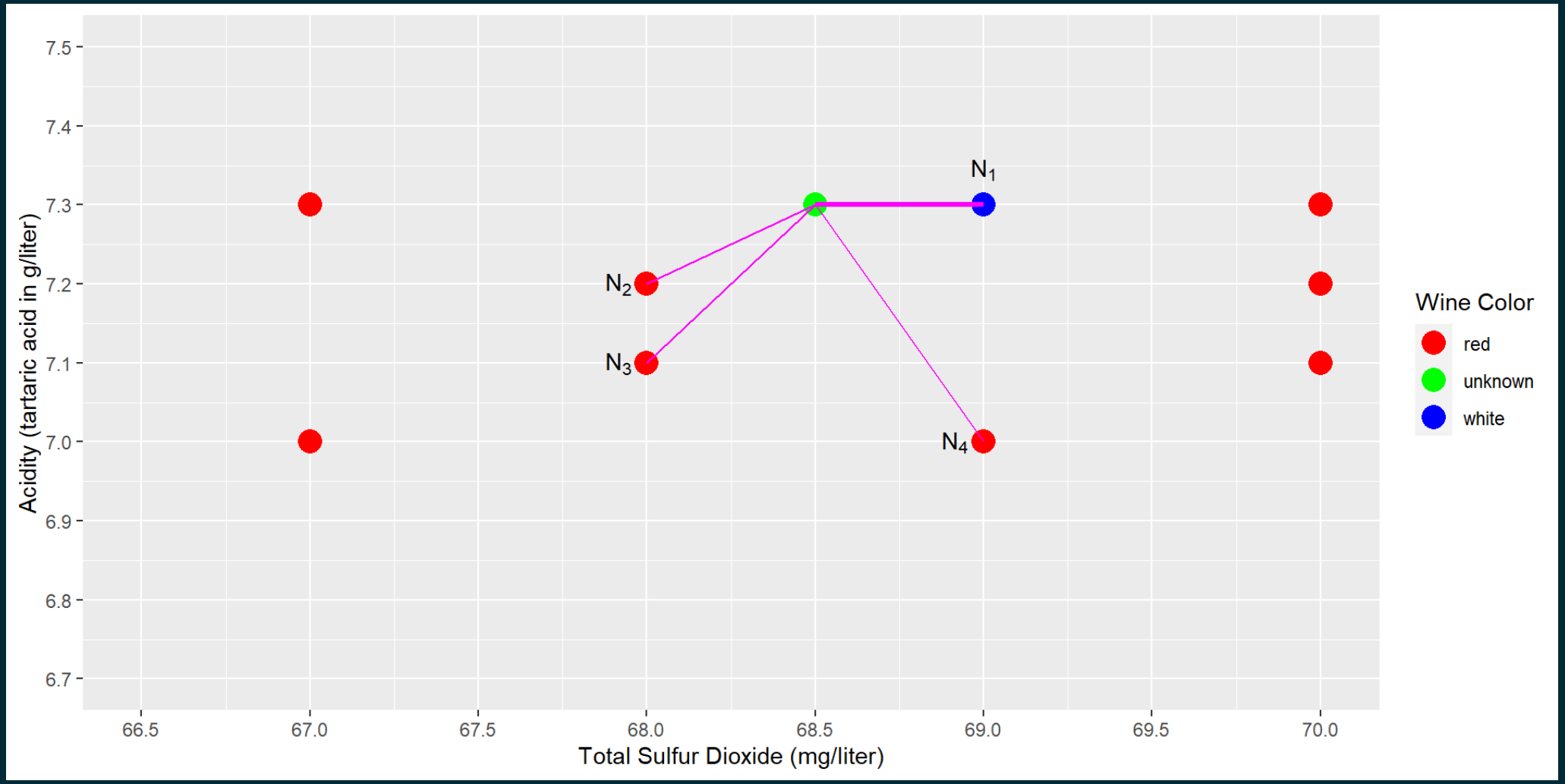


Acidity and Total Sulfur Dioxide Related to Wine Color

https://econ.lange-analytics.com/aibook/

K NEAREST NEIGHBORS K=4 (FOR A DIFFERENT UNKNOWN WINE)

4 NEAREST NEIGHBORS VOTE ON "RED" VS. "WHITE"

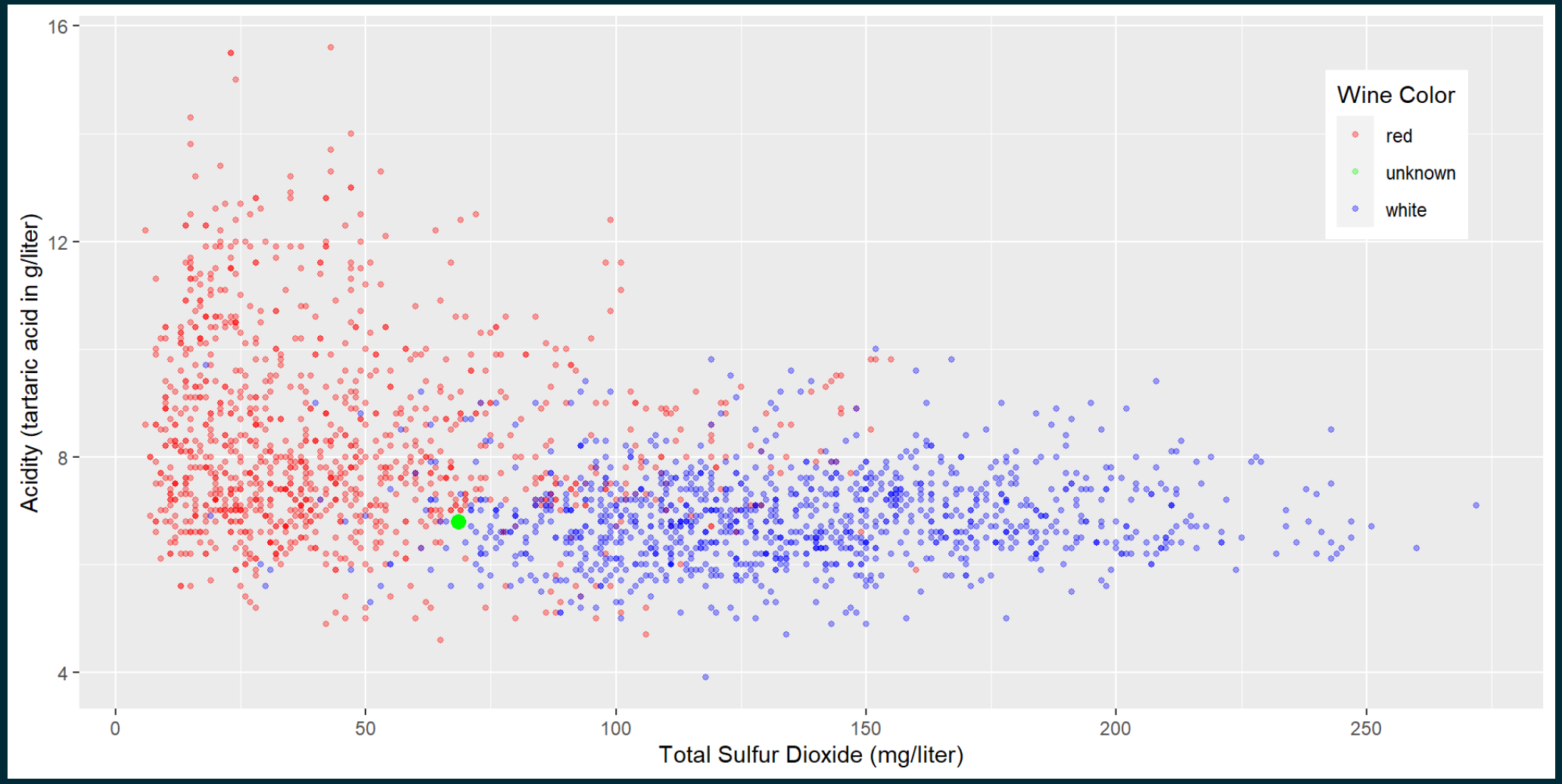


Predicting Wine Color with k-Nearest Neighbors (k=4)

https://econ.lange-analytics.com/aibook/

K NEAREST NEIGHBORS K=4 (FOR A DIFFERENT UNKNOWN WINE)

WATCH THE SCALE: G/LITER VS. MG/LITER. THAT DOES NOT LOOK RIGHT!



Acidity and Total Sulfur Dioxide Related to Wine Color »

https://econ.lange-analytics.com/aibook/

A FEW COMMON SCALING OPTIONS

- **Same units**

  Divide or multiply to get the same units. This is often not possible (e.g., `Height` and `Weight`). Or it is not feasible (e.g. `Alcohol` and `StrawberryJuice` content in spiked strawberry drink))

- **Rescaling**

  Generates a variable y that is scaled to a range between 0 and 1 based on the original variable's value x, its minimum x_{min} and its maximum x_{max}:

$$y = \frac{x - x_{min}}{x_{max} - x_{min}}$$

- **Z-Score Normalization**

  Z-score normalization uses the mean ($\overline{x}$) and the standard deviation (s) of a variable to scale the variable x to the variable z:

$$z = \frac{x - \overline{x}}{s}$$

»

TIME TO RUN K-NEAREST NEIGHBORS

LOADING DATA AND SELECTING VARIABLES

```
library(tidyverse); library(rio);library(janitor)
DataWine=import("https://lange-analytics.com/AIBook/Data/WineData.rds") %>%
         clean_names("upper_camel") %>%
         select(WineColor,Sulfur=TotalSulfurDioxide,Acidity) %>%
  mutate(WineColor=as.factor(WineColor))
print(DataWine)
```

```
# A tibble: 3,198 × 3
   WineColor Sulfur Acidity
   <fct>      <dbl>   <dbl>
 1 red           37    10.8
 2 white        213     6.4
 3 white        139     9.4
 4 white         90     8.2
 5 white        183     6.4
 6 red           38     6.7
 7 red           19    11.8
 8 white        220     6.7
 9 red           12     7.5
10 red           25     7.1
# … with 3,188 more rows
```

»

TIME TO RUN K-NEAREST NEIGHBORS

Generate Training and Testing Data (Splitting):

```
 1 library(tidymodels);
 2 set.seed(876)
 3 Split7030=initial_split(DataWine,prop=0.7,strata = WineColor)
 4
 5 DataTrain=training(Split7030)
 6 DataTest=testing(Split7030)
 7 print(DataTrain)
```

```
# A tibble: 2,238 × 3
   WineColor Sulfur Acidity
   <fct>      <dbl>   <dbl>
 1 red           37    10.8
 2 red           38     6.7
 3 red           12     7.5
 4 red           25     7.1
 5 red          114     8
 6 red           66     7.6
 7 red           49     6.8
 8 red          110     7
 9 red           44     6.5
10 red           10    10.4
# … with 2,228 more rows
```

```
 1 print(DataTest)
```

```
# A tibble: 960 × 3
   WineColor Sulfur Acidity
   <fct>      <dbl>   <dbl>
 1 white         90     8.2
 2 red           19    11.8
 3 white        220     6.7
 4 red          131     7.8
 5 white        161     7  
 6 red           41     9.9
 7 white        156     7.8
 8 white        150     6.5
 9 red          102     7.9
10 red           11     5.8
# … with 950 more rows
```

TIME TO RUN K-NEAREST NEIGHBORS

CLICK HERE TO FIND A REFERENCE LIST FOR VARIOUS `Step_` COMMANDS

Recipe: Prepare Data for Analysis:

```
1 RecipeWine=recipe(WineColor~Acidity+Sulfur, data = DataTrain) %>%
2             step_naomit() %>%
3             step_normalize(all_predictors())
```

Or:

```
1 RecipeWine=recipe(WineColor~., data = DataTrain) %>%
2             step_naomit() %>%
3             step_normalize(all_predictors())
```

```
1 print(RecipeWine)
```

```
Recipe

Inputs:

      role #variables
   outcome          1
 predictor          2

Operations:
```

TIME TO RUN K-NEAREST NEIGHBORS

CLICK HERE TO FIND A REFERENCE LIST FOR VARIOUS `Step_` COMMANDS

Creating a Model Design:

```
1 ModelDesignKNN=nearest_neighbor(neighbors = 4, weight_func = "rectangular") %>%
2                set_engine("kknn") %>%
3                set_mode("classification")
4 print(ModelDesignKNN)
```

```
K-Nearest Neighbor Model Specification (classification)

Main Arguments:
  neighbors = 4
  weight_func = rectangular

Computational engine: kknn
```

TIME TO RUN K-NEAREST NEIGHBORS

Putting it all together in a **fitted** `workflow`:

```
1  WFModelWine=workflow() %>%
2               add_recipe(RecipeWine) %>%
3               add_model(ModelDesignKNN) %>%
4               fit(DataTrain)
5  print(WFModelWine)
```

```
══ Workflow [trained] ══════════════════════════════════════════════════════════
Preprocessor: Recipe
Model: nearest_neighbor()

── Preprocessor ────────────────────────────────────────────────────────────────
2 Recipe Steps

• step_naomit()
• step_normalize()

── Model ───────────────────────────────────────────────────────────────────────

Call:
kknn::train.kknn(formula = ..y ~ ., data = data, ks = min_rows(4,     data, 5), kernel =
~"rectangular")

Type of response variable: nominal
Minimal misclassification: 0.1000894
```

TIME TO RUN K-NEAREST NEIGHBORS

How to use the **fitted** `workflow` to predict the wine color for the wines in the testing dataset:

1. Start with observation $i = 1$ from `DataTest` (the first observation).
2. Take observation i from `DataTest` and use `Acidity` and `Sulfur` to calculate the Euclidean distance to **each** of the observations of `DataTrain`.
3. Isolate the 4 observations with the smallest Euclidean distance and use the majority of their wine color as a prediction for observation i from `DataTest` (in case of a par, decide randomly).
4. Increase i by one (i.e., take the next observation from `DataTest`) and go to step 2 (until all `DataTest` observations are processed).

TIME TO RUN K-NEAREST NEIGHBORS

Predicting with the fitted workflow using `predict()` (not exactly helpful!):

```
1 predict(WFModelWine, DataTest)
```

```
# A tibble: 960 × 1
   .pred_class
   <fct>
 1 white
 2 red
 3 white
 4 white
 5 white
 6 red
 7 white
 8 white
 9 red
10 red
# … with 950 more rows
```

TIME TO RUN K-NEAREST NEIGHBORS

Predicting with the fitted workflow using `augment()` which *augments* `DataTest` with the predictions:

```
1 DataPredWithTestData=augment(WFModelWine, DataTest)
2 head(DataPredWithTestData)
```

```
# A tibble: 6 × 6
  WineColor Sulfur Acidity .pred_class .pred_red .pred_white
  <fct>      <dbl>   <dbl> <fct>           <dbl>       <dbl>
1 white         90     8.2 white            0.25        0.75
2 red           19    11.8 red              1           0
3 white        220     6.7 white            0           1
4 red          131     7.8 white            0.25        0.75
5 white        161     7   white            0           1
6 red           41     9.9 red              1           0
```

HAVING A DATA FRAME WITH `truth` AND `esimate` WE CAN CALCULATE PERFORMANCE METRICS

Confusion Matrix:

```
1 ConfMatrixWine=conf_mat(DataPredWithTestData, truth = WineColor, estimate = .pred_clas
2 print(ConfMatrixWine)
```

```
          Truth
Prediction red white
     red   436    46
     white  44   434
```

READING THE CONFUSION MATRIX

```
            Truth
Prediction   Red Wine White Wine
  Red Wine   TP: 436  FP: 46
  White Wine FN: 44   TN: 434
```

- The **positive class** (wine is "red") is in the **first column**. 436 of the positives are classified correctly (TR: true positives), and 44 positives are incorrectly classified (FN: false negatives).
- The **negative class** (wine is "white") is in the **second column**. 44 negatives are incorrectly classified (FP: false positives), and 434 negatives are classified correctly (TN: true negatives).

Accuracy: Number of wines on diagonal/number of all wines:

```
1 accuracy(DataPredWithTestData, truth = WineColor, estimate = .pred_class)
```

```
# A tibble: 1 × 3
  .metric  .estimator .estimate
  <chr>    <chr>          <dbl>
1 accuracy binary         0.906
```

WARNING: BE CAREFUL WITH THE ACCURACY RATE

THE STORY OF DR. NEBULOUS'S GAMBLERS SYSTEM

Dr. Nebulous offers a **97% Machine Learning Gambling Prediction**. Here is how it works: Gamblers can buy a prediction for a fee of $5. Dr. Nebulous will then run his famous machine learning model and send a closed envelope with the prediction. The gambler is supposed to open the envelope in the casino, right before placing a bet of $100 on a number in roulette. The envelope contains a message that states either "You will win" or "You will lose", which allows the gambler to act accordingly by either bet or not bet.

Dr. Nebulous claims that a "clinical trial" of 1000 volunteers, who opened the envelope after they had bet on a number in roulette, shows an accuracy of 97.3%.

How could Dr. Nebulous have such a precise model?

WARNING: BE CAREFUL WITH THETHE ACCURACY RATE

THE STORY OF DR. NEBULOUS'S GAMBLERS SYSTEM

The trick is Dr. Nebulous's machine learning model uses the *naive prognosis*: It always predicts "You will lose".

Here is the confusion matrix from the 1,000 volunteers trial:

	Truth	
Prediction	Win	Lose
Win	0	0
Lose	27	997

Roulette has 37 numbers to bet on. Chance to win is: $\frac{1}{37} = 0.027$.

Out of the 1000 volunteers, 27 are expected to win, and 973 are expected to lose.

$$Accuracy = \frac{0 + 973}{1000} = 0.973$$

WARNING: BE CAREFUL WITH THE ACCURACY RATE

THE STORY OF DR. NEBULOUS'S GAMBLERS SYSTEM

	Truth	
Prediction	Win	Lose
Win	0	0
Lose	27	997

However, when we look at the correct positive and the correct negative rate separately, we see that Dr. Nebulous' accuracy rate (although correct) makes little sense.

- The correct negative rate (**specificity**) is 100%
- The correct positive rate (**sensitivity**) is zero (out of the 27 winners, all were falsely predicted as "You will lose").

This example shows: When interpreting the confusion matrix, you must look at accuracy, sensitivity, and specificity simultaneously

TIME TO RUN K-NEAREST NEIGHBORS 

`accuracy()`, `sensitivity()` and `specificity()` for the wine data:

```
accuracy(DataPredWithTestData, truth = WineColor, estimate = .pred_class)
```

```
# A tibble: 1 × 3
  .metric  .estimator .estimate
  <chr>    <chr>          <dbl>
1 accuracy binary         0.906
```

```
sensitivity(DataPredWithTestData, truth = WineColor, estimate = .pred_class)
```

```
# A tibble: 1 × 3
  .metric     .estimator .estimate
  <chr>       <chr>          <dbl>
1 sensitivity binary         0.908
```

```
specificity(DataPredWithTestData, truth = WineColor, estimate = .pred_class)
```

```
# A tibble: 1 × 3
  .metric     .estimator .estimate
  <chr>       <chr>          <dbl>
1 specificity binary         0.904
```

Can we improve by using all predictors.»

PROJECT: DESIGN A MACHINE LEARNING WORKFLOW FOR OPTICAL CHARACTER RECOGNITION »

https://econ.lange-analytics.com/aibook/

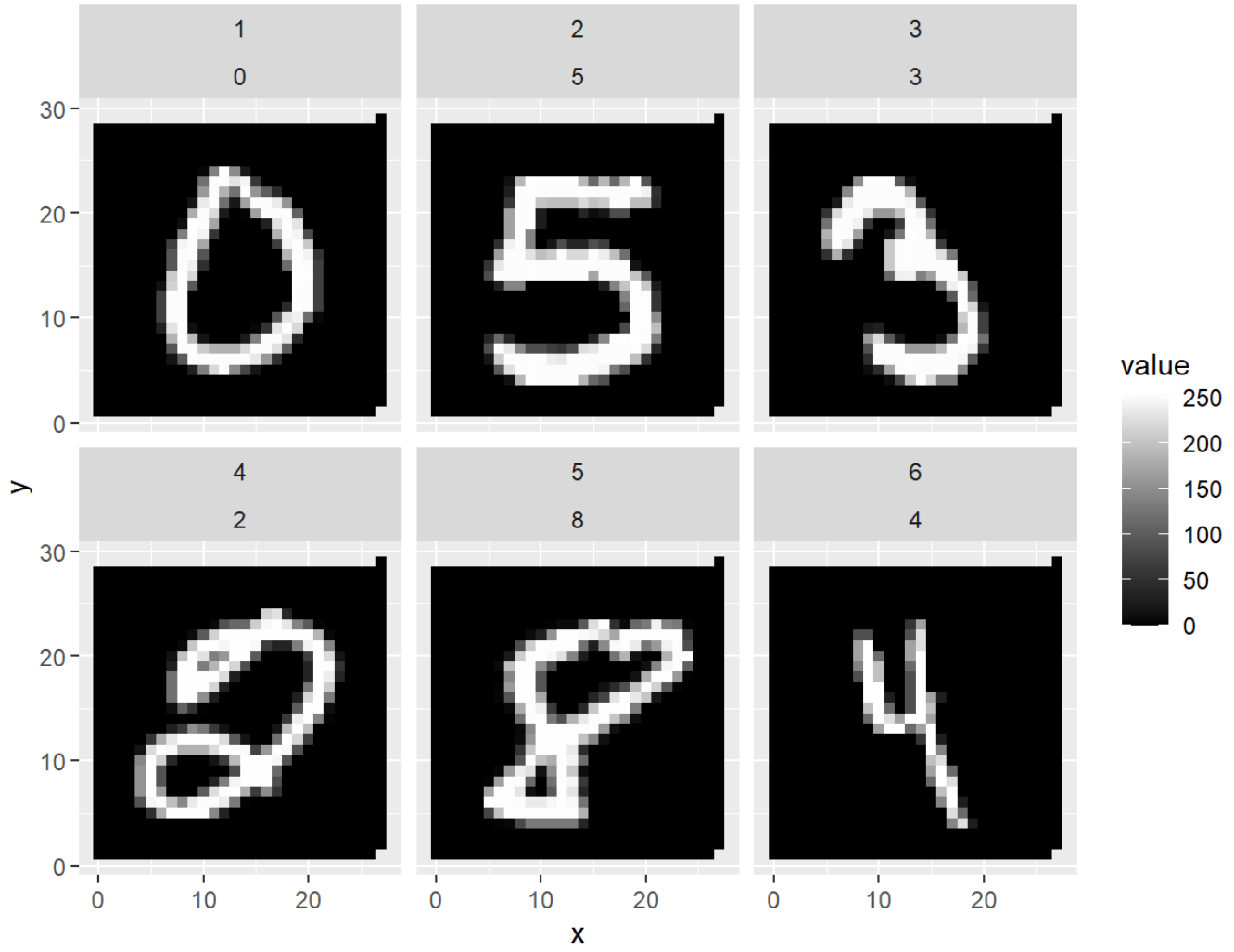
MNIST DATA SET

You will develop a machine learning model based on k-Nearest Neighbors to recognize handwritten digits from images.

You will use the MNIST dataset, a standard dataset for image recognition in machine learning (60,000 images for training and 10,000 images for testing). Developed by LeCun, Cortes, and Burges (2010) based on two datasets from handwritten digits obtained from Census workers and high school students.

We will use only the first 500 images of the original MNIST dataset to speed up the *k-Nearest Neighbors* model's training time.

VISUALIZATION OF THE FIRST SIX IMAGES FROM THE MNIST DATA SET



https://econ.lange-analytics.com/aibook/

HOW A IMAGE IS STORED IN THE MNIST DATASET

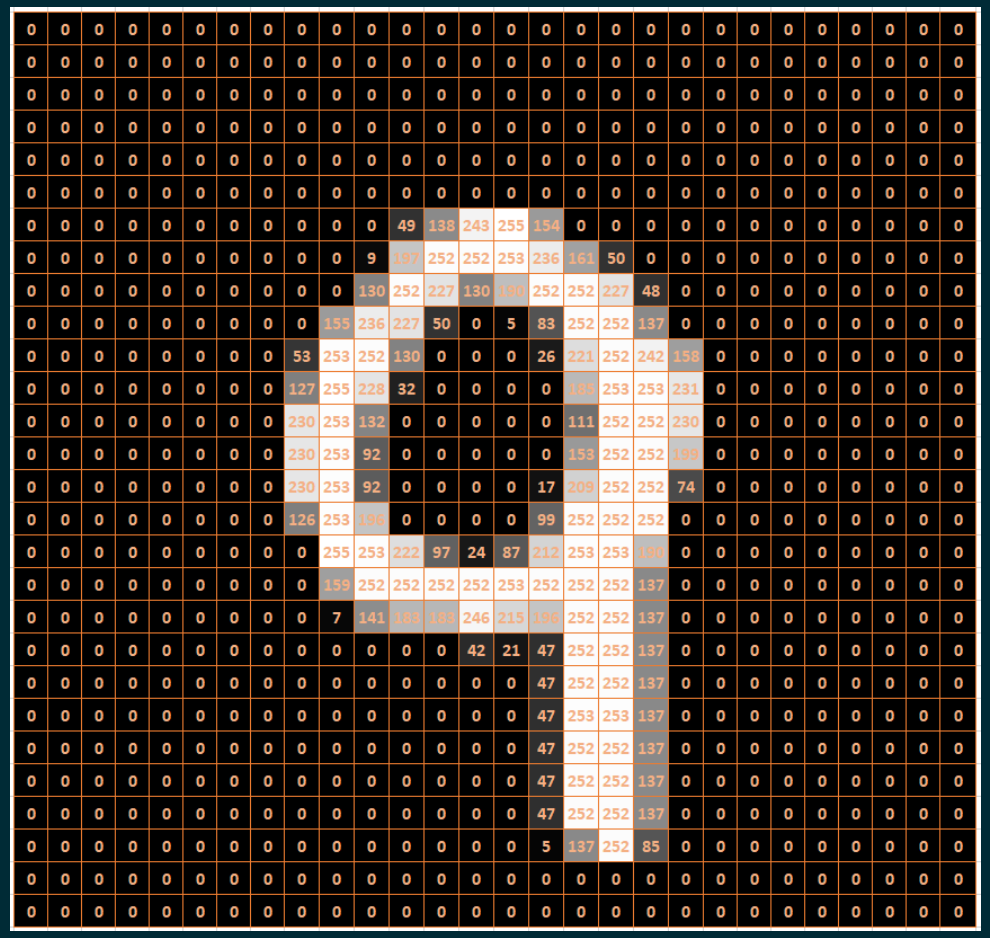


Image of a Handwritten Nine

The image has 28 rows and 28 columns. Each of the 784 cells (pixels) holds a value between 0 (black) and 255 (white)

https://econ.lange-analytics.com/aibook/

HOW A IMAGE IS STORED IN THE MNIST DATASET

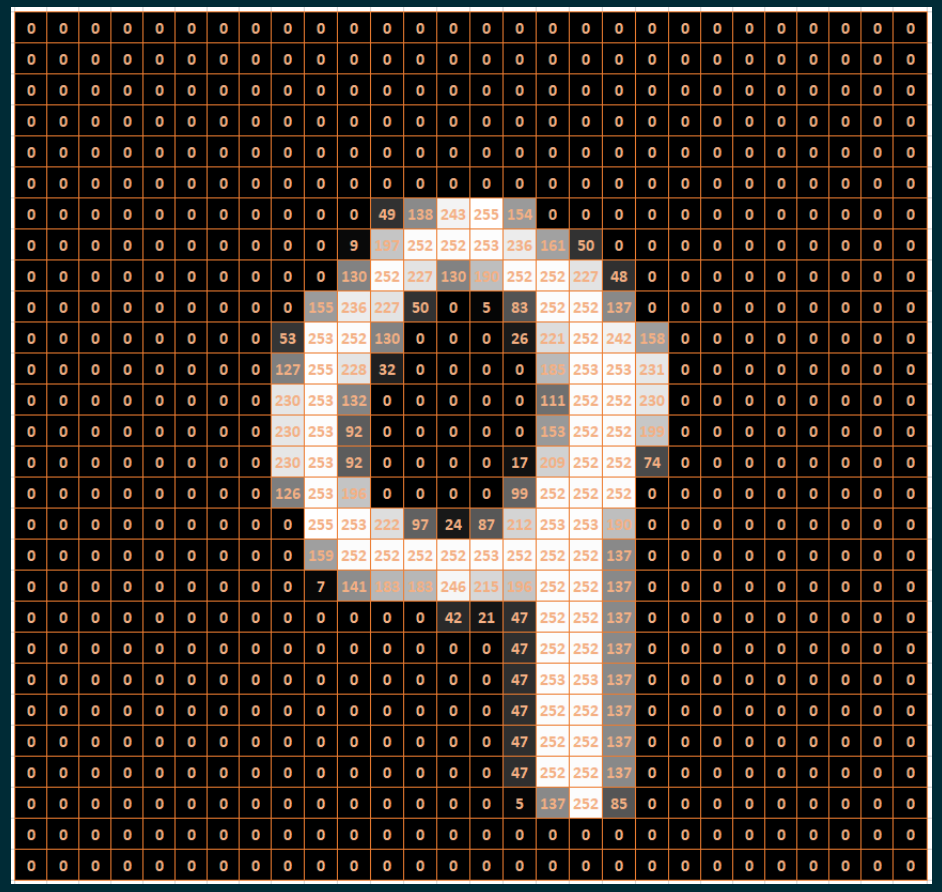


Image of a Handwritten Nine

- Pixel values for a single image are not stored in a table. Ohterwise we would end-up with a table containing tables.
- Pixel values are stored as one row for each image.
- Concatenating the 28 rows of an image into one row with 28*28=784 cells (pixels)

THREE ROWS FROM THE DATA FRAME OF THE MNIST DATASET

```
1  print(Mnist4PlotAndTable[1:3,1:784])
```

```
  Label Pix1 Pix2 Pix3 Pix4 Pix5 Pix6 Pix7 Pix8 Pix9 Pix10 Pix11 Pix12 Pix13
1     0    0    0    0    0    0    0    0    0    0     0     0     0     0
2     5    0    0    0    0    0    0    0    0    0     0     0     0     0
3     3    0    0    0    0    0    0    0    0    0     0     0     0     0
  Pix14 Pix15 Pix16 Pix17 Pix18 Pix19 Pix20 Pix21 Pix22 Pix23 Pix24 Pix25 Pix26
1     0     0     0     0     0     0     0     0     0     0     0     0     0
2     0     0     0     0     0     0     0     0     0     0     0     0     0
3     0     0     0     0     0     0     0     0     0     0     0     0     0
  Pix27 Pix28 Pix29 Pix30 Pix31 Pix32 Pix33 Pix34 Pix35 Pix36 Pix37 Pix38 Pix39
1     0     0     0     0     0     0     0     0     0     0     0     0     0
2     0     0     0     0     0     0     0     0     0     0     0     0     0
3     0     0     0     0     0     0     0     0     0     0     0     0     0
  Pix40 Pix41 Pix42 Pix43 Pix44 Pix45 Pix46 Pix47 Pix48 Pix49 Pix50 Pix51 Pix52
1     0     0     0     0     0     0     0     0     0     0     0     0     0
2     0     0     0     0     0     0     0     0     0     0     0     0     0
3     0     0     0     0     0     0     0     0     0     0     0     0     0
  Pix53 Pix54 Pix55 Pix56 Pix57 Pix58 Pix59 Pix60 Pix61 Pix62 Pix63 Pix64 Pix65
1     0     0     0     0     0     0     0     0     0     0     0     0     0
2     0     0     0     0     0     0     0     0     0     0     0     0     0
3     0     0     0     0     0     0     0     0     0     0     0     0     0
  Pix66 Pix67 Pix68 Pix69 Pix70 Pix71 Pix72 Pix73 Pix74 Pix75 Pix76 Pix77 Pix78
1     0     0     0     0     0     0     0     0     0     0     0     0     0
2     0     0     0     0     0     0     0     0     0     0     0     0     0
3     0     0     0     0     0     0     0     0     0     0     0     0     0
  Pix79 Pix80 Pix81 Pix82 Pix83 Pix84 Pix85 Pix86 Pix87 Pix88 Pix89 Pix90 Pix91
```

GO TO PROJECT IN BOOK 

BUILD YOUR OWN OCR SYSTEM.»

https://econ.lange-analytics.com/aibook/