Vino & Machine Learning: The Perfect Pairing

- Jessica, Kevin, Tim, Viktor, & Mandy

On this Journey

THE DATASET & ETL
POURING THE FOUNDATION

DATA ANALYSIS
UNCORKING INSIGHTS

PREPROCESSING
DECANTING DATA FOR ML

MACHINE LEARNING
SIPPIN' ON ALGORITHMS

FLASK BOTTLING UP THE RESULTS

Why Wine?... Wine Not!

Ever bought a bottle that you didn't really enjoy?

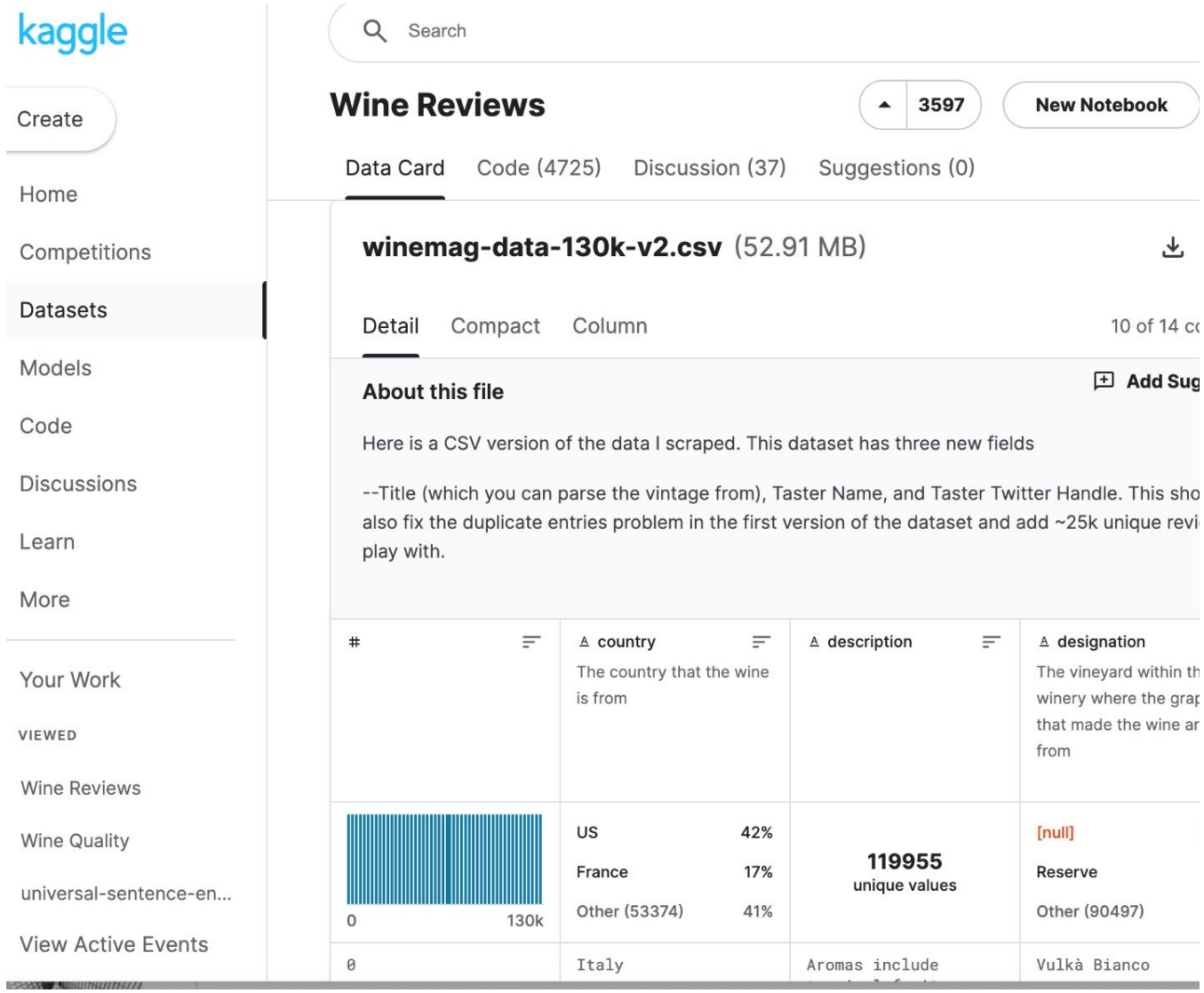
Imagine if ML could help you pick the perfect bottle within your budget?

That's what we explored!

- Where can you get the highest quality wine within your budget?
- What differences exist across regions and types of wine?

The Dataset

- Sourced from Kaggle
- Filled with wine reviews from Wine Enthusiast
- Started with a cellar of 130,000 entries
- Bursting with details on price, ratings, country, variety, and description
- Using our ETL process, refined our selection to 77,000



Added New Flavors

Type: We crafted a dictionary to sort wine varieties into types such as Red, White, Rose, and Sparkling.

Rating Category: We created a new column to classify wines according to their point-based ratings.

Vintage: We retrieved the vintage year from the title column.

-97 **Superb**
A great achievement

90-93 **Excellent**
Highly recommended

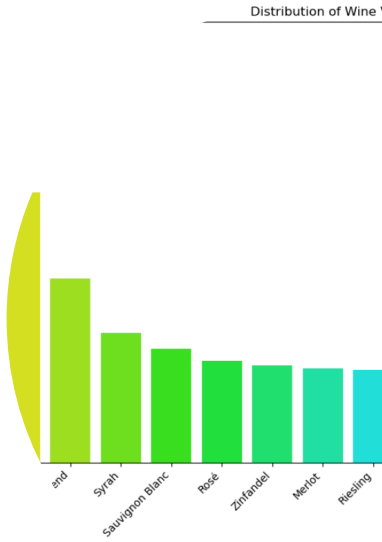
87-89 **Very Good**
Often good value; well recommended

26 **Good**
Suitable for everyday

```
                                    with counts less than or equ
                ety_counts = variety_counts[vari

               ariety counts after filtering (>= 100
             filtered_variety_counts)
```

```
ety counts after filtering (>= 1000):
riety
not Noir                    11693
hardonnay                   10070
Cabernet Sauvignon           8284
Red Blend                    7537
Bordeaux-style Red Blend     5123
Syrah                        3617
Sauvignon Blanc              3177
Rosé                         2839
Zinfandel                    2707
Merlot                       2630
Riesling                     2593
lalbec                       2417
ngiovese                     2369
bbiolo                       2325
e Blend                      1838
ling Blend                   1837
illo                         1773
le Red Blend                 1376
                             1284
                             1247
                             1195
+64
```



Uncorking Insights: Data Analysis

10 Things You Should Know About Wine

Visit Our GitHub Page

We Love Wine!

Where do you want to go?:

10 Things You Should Know About Wine

Tableau Dashboard

Make a Prediction

Git Hub Repo

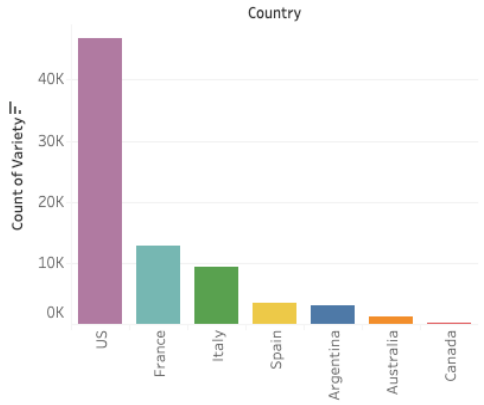
0:00 / 0:19

Where do you want to go?:

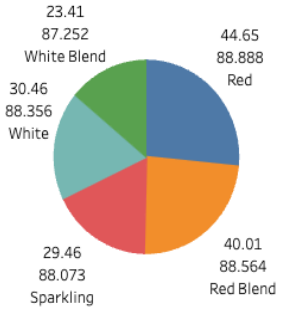
Tableau Dashboard Go

Wine Story Across The World

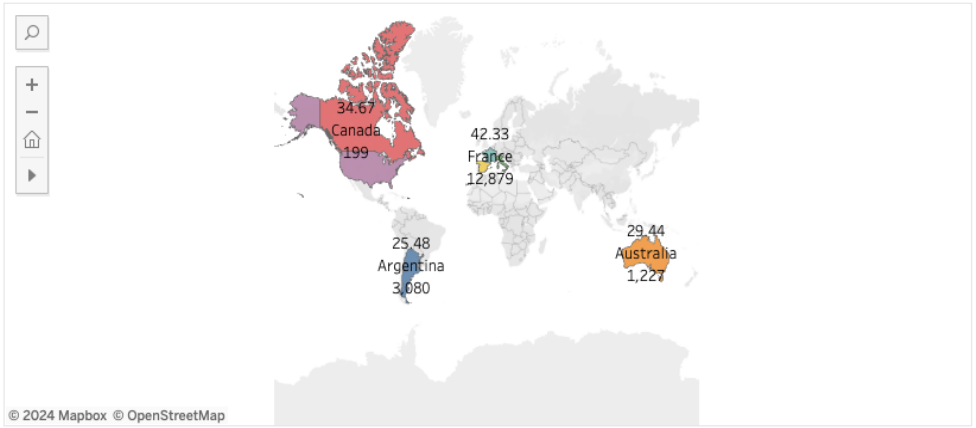
Wine Variety per Country



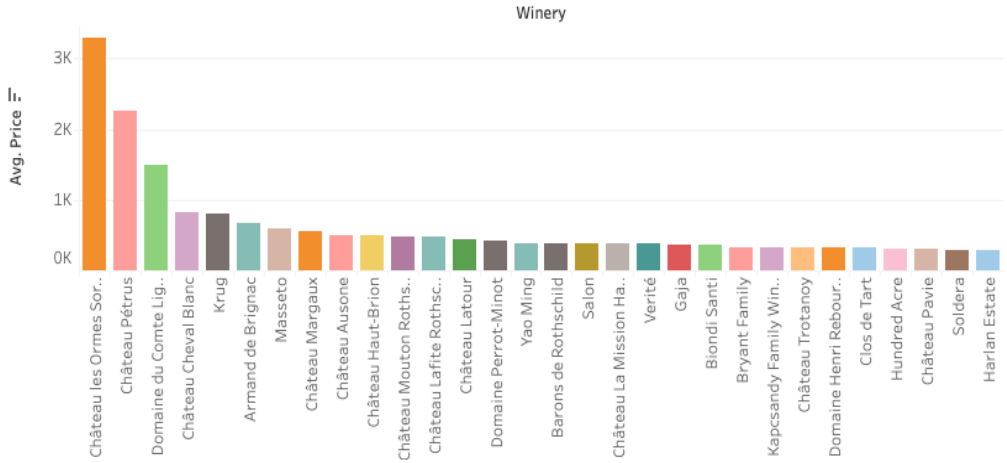
Average Price and Average Points per Wine Type



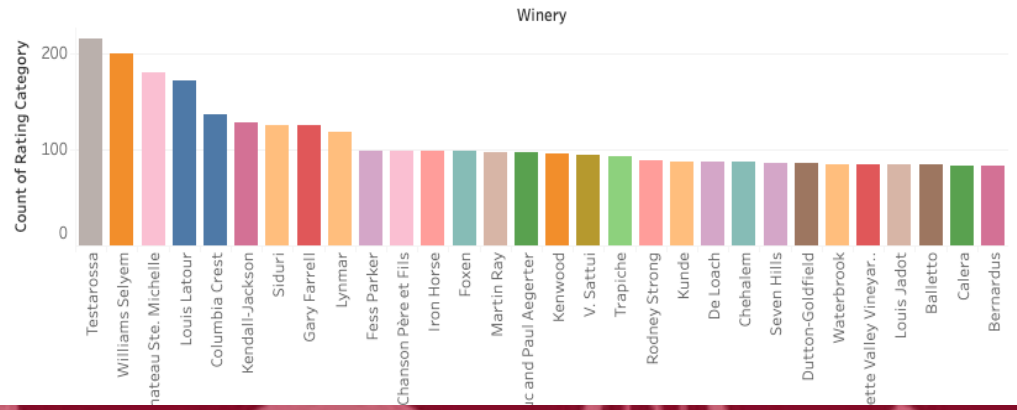
Countries Map



Average Price per Winery



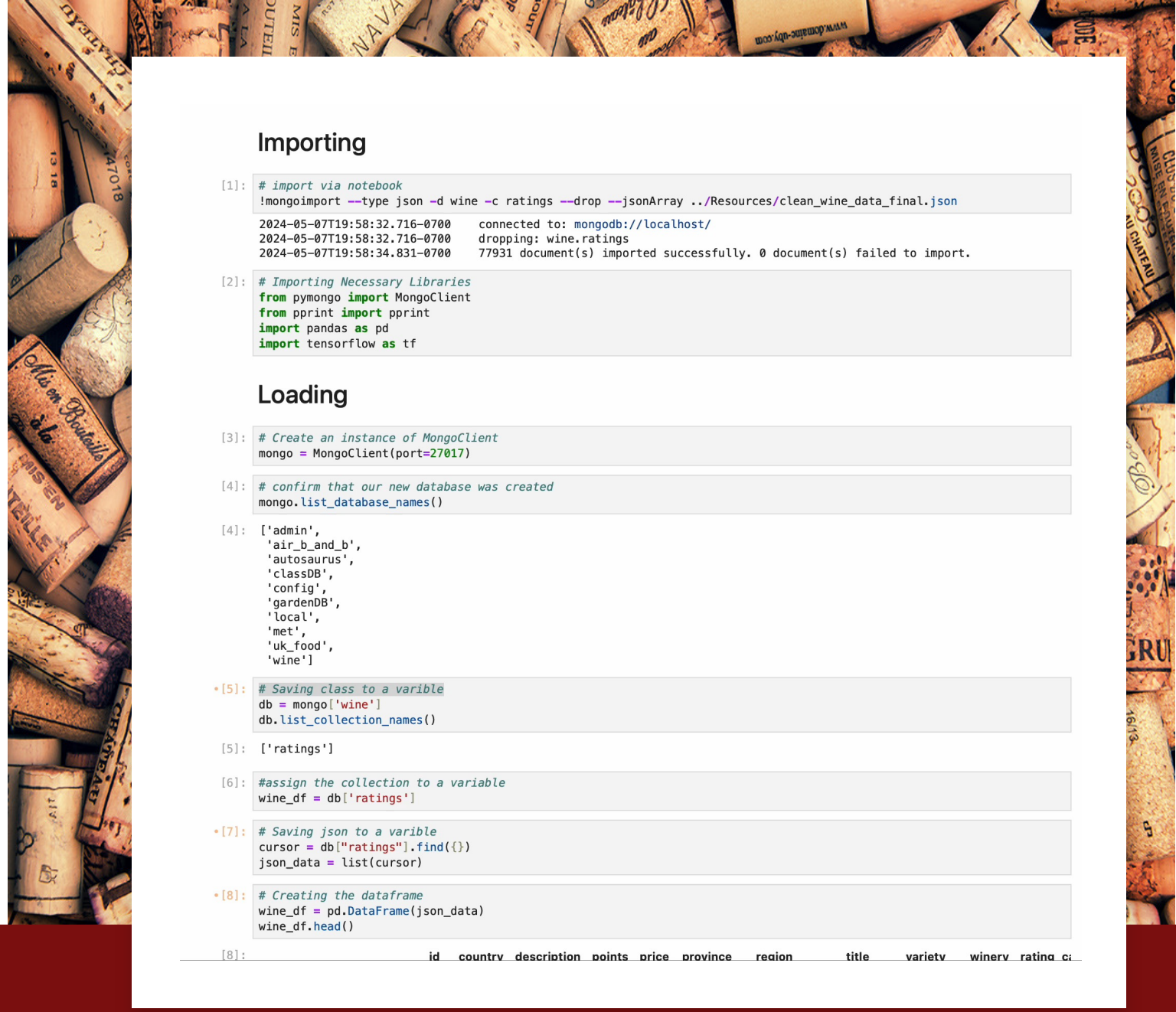
Rating Category per Winery



Decanting the Data:
Machine Learning

Pre-Processing

- **Dummy Encoding**:
  - Dummy coding is similar to one-hot encoding but can represent categorical variables with more than two categories. Instead of using just 0 and 1, dummy coding assigns different integers (e.g., 0, 1, 2, etc.) to represent each category.
  - Dummy encoding is used when there is an inherent order or hierarchy among the categories, and the dropped category serves as the reference level.
  - Dummy encoding is commonly used in linear regression, logistic regression, and decision tree-based models.



Importing

```
[1]: # import via notebook
!mongoimport --type json -d wine -c ratings --drop --jsonArray ../Resources/clean_wine_data_final.json
```

```
2024-05-07T19:58:32.716-0700    connected to: mongodb://localhost/
2024-05-07T19:58:32.716-0700    dropping: wine.ratings
2024-05-07T19:58:34.831-0700    77931 document(s) imported successfully. 0 document(s) failed to import.
```

```
[2]: # Importing Necessary Libraries
from pymongo import MongoClient
from pprint import pprint
import pandas as pd
import tensorflow as tf
```

Loading

```
[3]: # Create an instance of MongoClient
mongo = MongoClient(port=27017)
```

```
[4]: # confirm that our new database was created
mongo.list_database_names()
```

```
[4]: ['admin',
 'air_b_and_b',
 'autosaurus',
 'classDB',
 'config',
 'gardenDB',
 'local',
 'met',
 'uk_food',
 'wine']
```

```
[5]: # Saving class to a varible
db = mongo['wine']
db.list_collection_names()
```

```
[5]: ['ratings']
```

```
[6]: #assign the collection to a variable
wine_df = db['ratings']
```

```
[7]: # Saving json to a varible
cursor = db["ratings"].find({})
json_data = list(cursor)
```

```
[8]: # Creating the dataframe
wine_df = pd.DataFrame(json_data)
wine_df.head()
```

```
[8]:   id  country  description  points  price  province  region  title  variety  winery  rating_ca
```

DF manipulation

[9]:
```
# Drop the non-beneficial ID columns
wine_df = wine_df.drop(columns = ['_id','description','title', 'winery', 'region'])
wine_df.head()
```

[9]:

	country	points	price	province	variety	rating_category	type	vintage
0	Argentina	87	30	Other	Malbec	Good	Red	2010
1	US	87	12	California	Chardonnay	Good	White	2012
2	US	87	13	Michigan	Riesling	Good	White	2013
3	US	87	19	California	Cabernet Sauvignon	Good	Red	2011
4	Argentina	87	13	Mendoza Province	Malbec	Good	Red	2011

[10]:
```
# Create target
wine_df['target'] =wine_df['points']>=90
wine_df.tail()
```

[10]:

	country	points	price	province	variety	rating_category	type	vintage	target
77926	US	91	55	California	Pinot Noir	Excellent	Red	2014	True
77927	US	90	35	California	Zinfandel	Very Good	Red	2012	True
77928	France	90	28	Alsace	Pinot Gris	Very Good	White	2013	True
77929	US	90	75	Oregon	Pinot Noir	Very Good	Red	2004	True
77930	France	90	32	Alsace	Pinot Gris	Very Good	White	2012	True

[11]:
```
#Drop uneeded columns
wine_df = wine_df.drop(columns = ['points','rating_category', 'province', 'type'])
```

[12]:
```
# Review
wine_df.head()
```

[12]:

	country	price	variety	vintage	target
0	Argentina	30	Malbec	2010	False
1	US	12	Chardonnay	2012	False
2	US	13	Riesling	2013	False
3	US	19	Cabernet Sauvignon	2011	False
4	Argentina	13	Malbec	2011	False

[13]:
```
# Check for null
wine_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
```

Pre-Processing

- **One-Hot Encoding**:
  - Each binary variable represents one category and has a value of 1 if the observation belongs to that category and 0 otherwise.
  - One-hot encoding is typically used when the categorical variable has no inherent order or hierarchy among its categories.
  - One-hot encoding results in a sparse matrix with many zero values, especially when dealing with variables with a large number of categories.
  - One-hot encoding is commonly used in machine learning models like neural networks.

Sippin' on Algorithms: Data Modeling Optimization

Data Model implementation

Binary Classification Neural Network Model

Data Preprocessing:

- Split our preprocessed data into our features (X) and target (y) arrays
- The dataset is then split into training and testing sets using the train_test_split function from Scikit-Learn.

Data Standardization:

- We standardized or scaled the feature data using the StandardScaler function from Scikit-Learn to improve the convergence and performance of the neural network.

Neural Network Model Definition: We defined our deep learning model using;

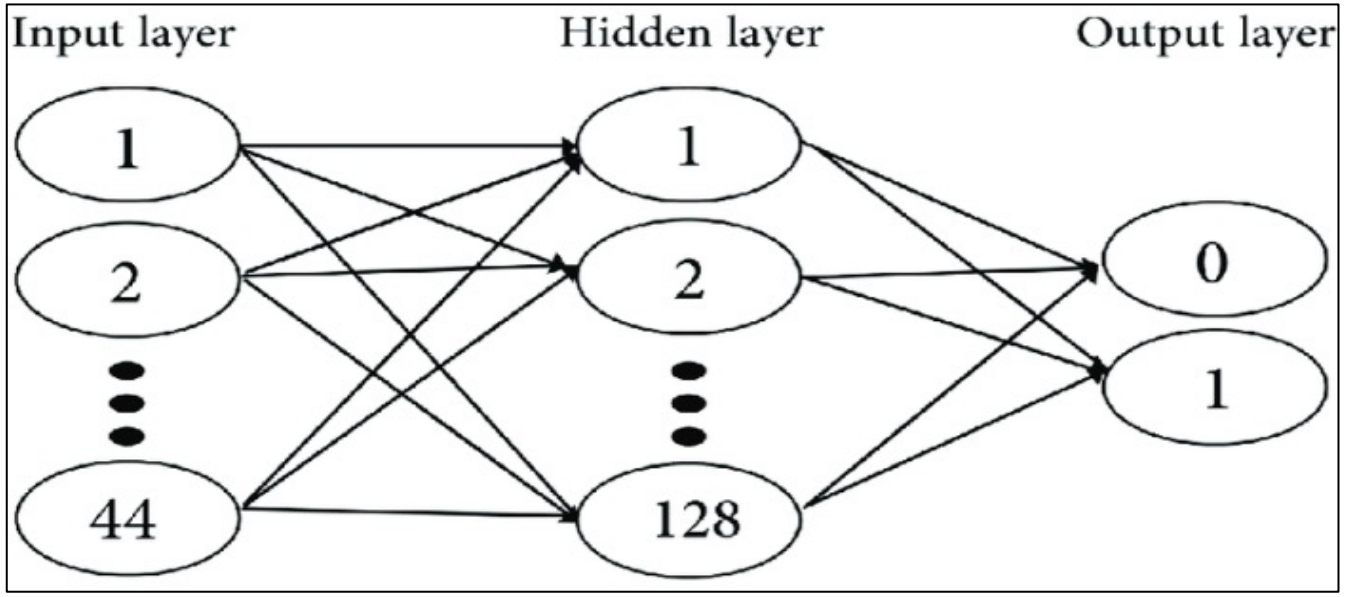
- A Sequential model which consists of:
  - An input layer (Dense) with 68 units (equal to the number of features in the dataset) and ReLU activation function.
  - Two hidden layers (Dense) with 16 units each and ReLU activation functions.
  - And an output layer (Dense) with 1 unit and a sigmoid activation function, which is suitable for our binary classification tasks.

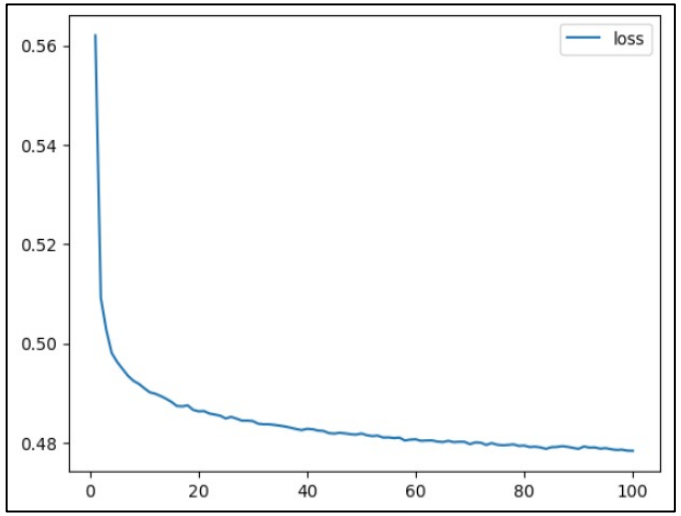
Model Compilation:

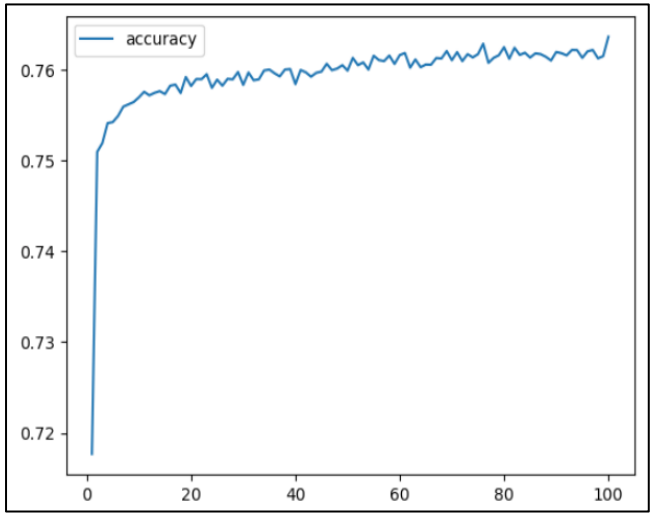
- The model is compiled using the compile function, specifying the loss function as "binary_crossentropy" (appropriate for binary classification), the optimizer as "adam", and the metric to monitor during training as "accuracy".

Model Training:

- The model is then trained using the fit function taking the scaled training features (X_train_scaled) and the corresponding target labels (y_train) as input, with the number of epochs set to 100.
- After training our model with the test dataset, it achieved a moderate level of performance with an accuracy of approximately 75.31% and a corresponding loss value of 0.5001.








```
# Evaluate the model using the test data
model_loss, model_accuracy = nn_model.evaluate(X_test_scaled,y_test,verbose=2)
print(f"Loss: {model_loss}, Accuracy: {model_accuracy}")
```

```
609/609 - 0s - 278us/step - accuracy: 0.7531 - loss: 0.5001
Loss: 0.5001154541969299, Accuracy: 0.7530667781829834
```

Bottling up Results: Flask

MVP & Iteration

- Basic app working locally
- Loading model and connecting dataset
- Making predictions based on test-data
- Making unique predictions
- Deploying Flask App
  - Tensorflow challenge

```
Prediction Function
f predict_rating(predict_row):
    loaded_model = tf.keras.models.load_model('static/resources
    result = loaded_model.predict(predict_row)  # Making a sing
    return result[0][0]

# Prediction accuracy function : compares passed values of pred
def pred_accuracy(prediction, target):
    if prediction == target:
        return "The model's prediction was correct!"
    else:
        return "The model got it wrong for this wine"
```

Possible Future Use

- To refine the wine predictive modeling over time by upskilling in ETL process so the predictions can supply more descriptive insights for the consumer when searching for wine.
- Incorporate a user form and have the model create a prediction based of the input.
- Improvements on accuracy from exploring different algorithms, such as gradient boosting or neural networks, which may offer greater predictive power.
- Further development could focus on creating a recommendation system to assist consumers in selecting wines based on personalized preferences and historical data.

Thank you & Cheers!

Credit: Images from Unsplash