THE UNIVERSITY OF CHICAGO

ADSP 31009/IP03 [44148]
Machine Learning & Predictive Analytics
Christopher Marasco

Good deal or bad?

A neural network regression model predicting wine prices and helping guide your wine-buying decisions

May 20th, 2024

The reason we're here: Problem Statement

Executive Summary:

The wine industry is a crowded market. Just ask yourself how easy it was to navigate your local grocer's section let alone a wholesaler's aisles upon aisles.

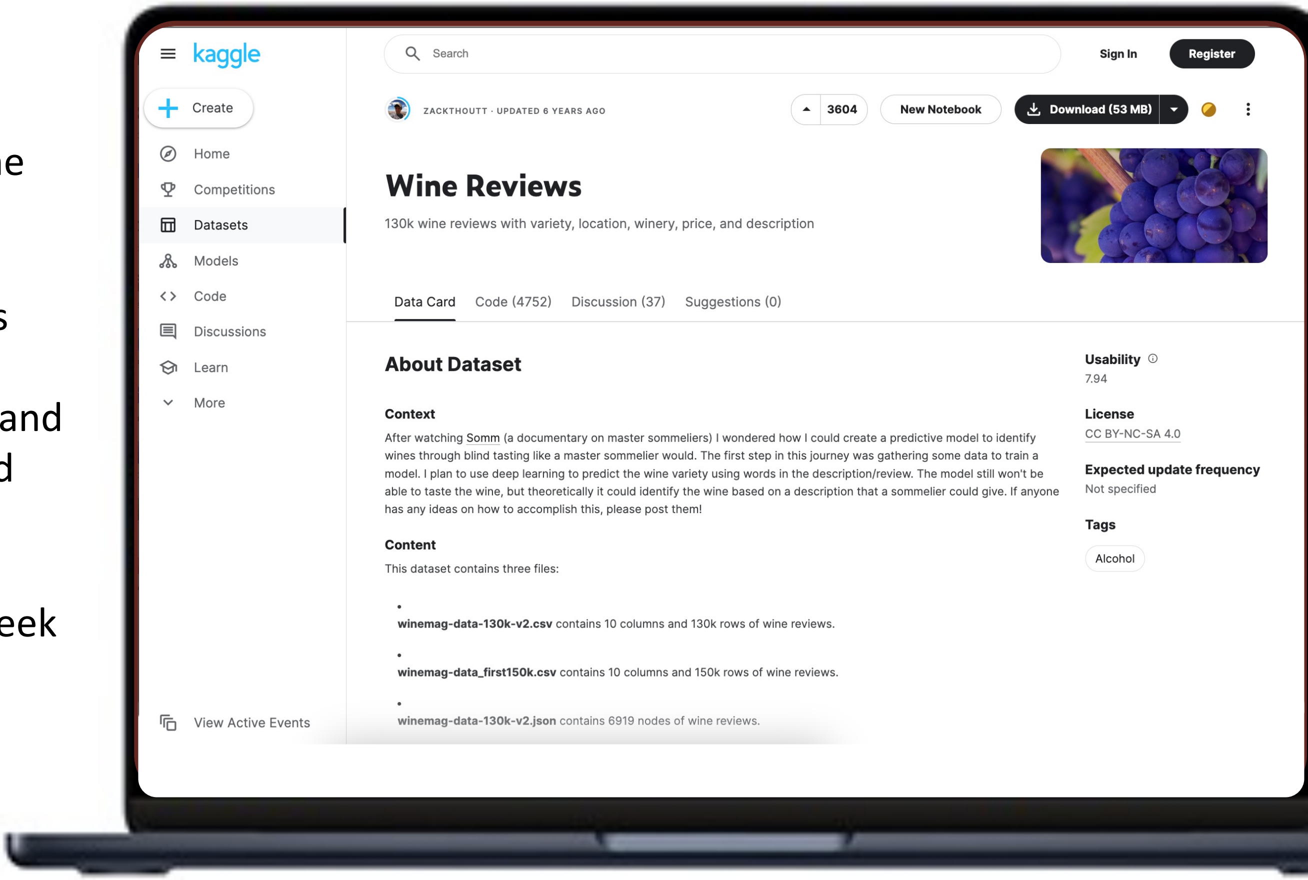
The reason for this is obvious, it's a 440+ billion-dollar global industry.

But with that crowding comes noise. We are here today to build a well-generalized deep learning model to help an average wine enthusiast buy good wine at good prices.

Assumptions, hypothesis, and data set

We pick up a thread from the Kaggle*

- Our publisher wonders if we can build an ML model to predict wine varietals through blind-tasting (for a machine this would be leveraging text descriptors)
- I chose to leverage the dataset, but tweak the hypothesis
- I would rather, like to use the data to predict wine price, and when my model is fit enough, look for outliers that would indicate a good buy (or a bad)
- The data was scraped from WineEnthusiast during the week of June 15th, 2017, so one incredibly huge assumption is that this data, though voluminous, is representative of a current market



Wine Reviews

130k wine reviews with variety, location, winery, price, and description

About Dataset

Context

After watching Somm (a documentary on master sommeliers) I wondered how I could create a predictive model to identify wines through blind tasting like a master sommelier would. The first step in this journey was gathering some data to train a model. I plan to use deep learning to predict the wine variety using words in the description/review. The model still won't be able to taste the wine, but theoretically it could identify the wine based on a description that a sommelier could give. If anyone has any ideas on how to accomplish this, please post them!

Content

This dataset contains three files:

- **winemag-data-130k-v2.csv** contains 10 columns and 130k rows of wine reviews.
- **winemag-data_first150k.csv** contains 10 columns and 150k rows of wine reviews.
- **winemag-data-130k-v2.json** contains 6919 nodes of wine reviews.

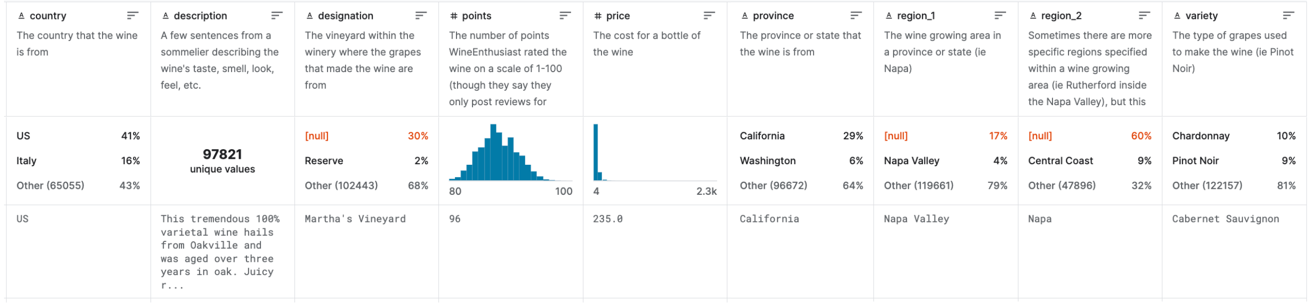
** https://www.kaggle.com/datasets/zynicide/wine-reviews*

Exploratory Data Analysis

Apart from removing wasteful features, and dropping some outliers, our dataset is fairly complete and simplistic

A sample of our original dataset

We dropped 'Description' as irrelevant for my exercise and 'designation' as redundant. Despite many nulls in the region columns, we wanted to keep those for context if possible.

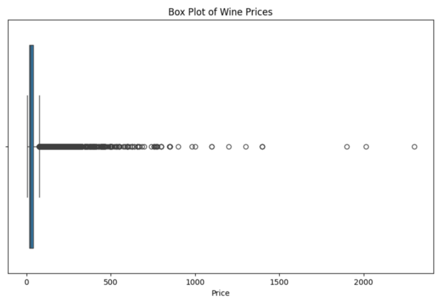


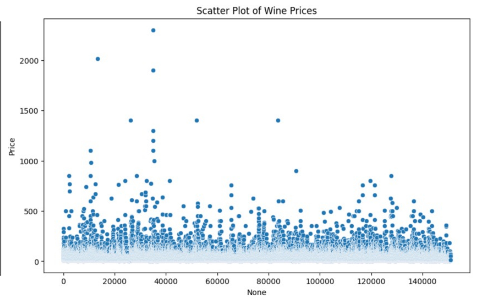
country	description	designation	points	price	province	region_1	region_2	variety
The country that the wine is from	A few sentences from a sommelier describing the wine's taste, smell, look, feel, etc.	The vineyard within the winery where the grapes that made the wine are from	The number of points WineEnthusiast rated the wine on a scale of 1-100 (though they say they only post reviews for	The cost for a bottle of the wine	The province or state that the wine is from	The wine growing area in a province or state (ie Napa)	Sometimes there are more specific regions specified within a wine growing area (ie Rutherford inside the Napa Valley), but this	The type of grapes used to make the wine (ie Pinot Noir)
US 41% Italy 16% Other (65055) 43%	97821 unique values	[null] 30% Reserve 2% Other (102443) 68%	80 100	4 2.3k	California 29% Washington 6% Other (96672) 64%	[null] 17% Napa Valley 4% Other (119661) 79%	[null] 60% Central Coast 9% Other (47896) 32%	Chardonnay 10% Pinot Noir 9% Other (122157) 81%
US	This tremendous 100% varietal wine hails from Oakville and was aged over three years in oak. Juicy r...	Martha's Vineyard	96	235.0	California	Napa Valley	Napa	Cabernet Sauvignon

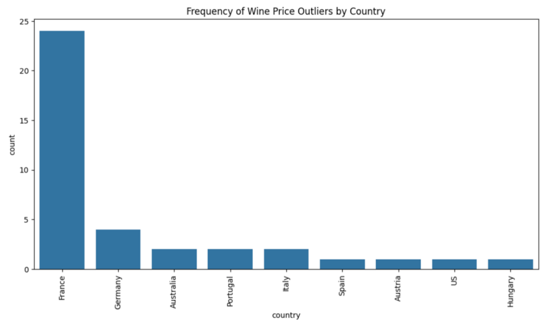
- country ✓
- description ✗
- designation ✗
- Points ✓
- price ✓
- province ✓
- region_1 ✓
- region_2 ✓
- variety ✓

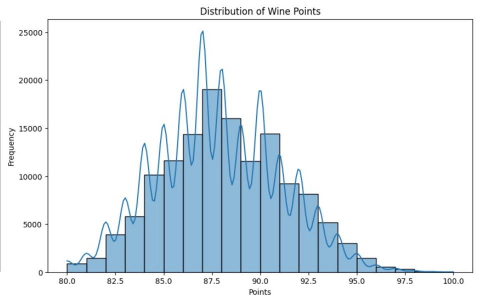
A review of our predictor and feature set

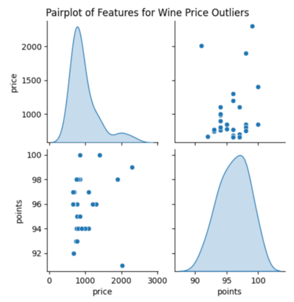
Immediately we see outliers not be worth leveraging if we want a more generalized model and functional recommendation tool (novice wine buyers aren't in market for $2000 Bordeaux)











Feature Engineering & Transformations

1 **Encode our categorical data**

Neural networks cannot work directly with categorical text data. They require numerical input.

We use Sklearns LabelEncoder to convert our strings to integer values

```
Fromsklearn.preprocessing
import LabelEncoder,
StandardScaler
```

2 **Normalize our target 'Price'**

We also need to normalize our target variable. Standardizing to improve model performance.

```
scaler =StandardScaler()
df['points'] =
scaler.fit_transform(df[
['points']])
```

3 **Split our data for train and test**

Splitting the data allows us to train the model on one subset of the data and evaluate it on another subset to see how well it generalizes to unseen data.

```
X_train, X_test, y_train,
y_test = train_test_split(X,
y, test_size=0.2,
random_state=42)
```

Model approach and fit

Keras

1: We chose to use **Keras** to generate our **neural network** because of it's

- Ease of Use /User-Friendly API
- Integration with TensorFlow
- Flexibility:
- Extensive Documentation and Community Support:

2: Fully Connected neurons

- We used two fully connected (Dense) layers with normalization and a dropout rate applied to each
- At each neuron we add a ReLU activation function to introduce some non-linearity

3: Compile our model

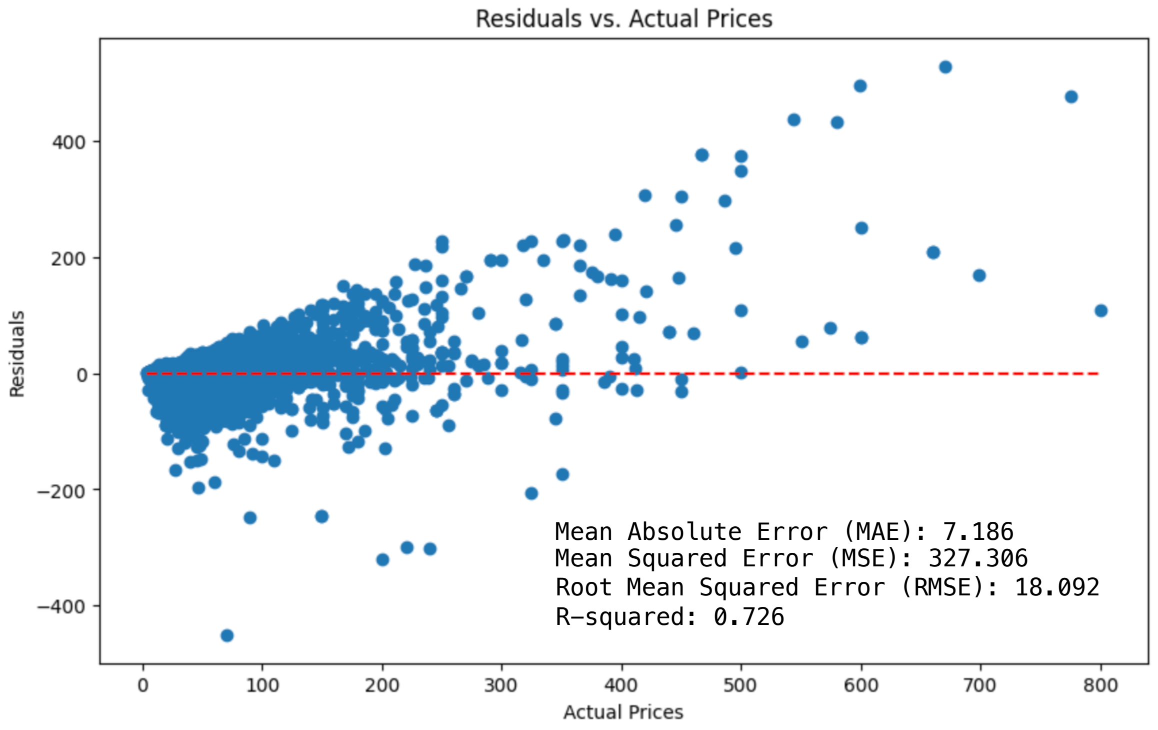
- Both input and output layers are complied with our objective function set to minimize our mean squared error

4: Run the model

- We added a validation split of 20%
- Ran 50 epochs
- on a batch of 32

5: Model Fit

Visually we're having a hard time with the higher priced wins, where we clearly see strong residual spread



```
Mean Absolute Error (MAE): 7.186
Mean Squared Error (MSE): 327.306
Root Mean Squared Error (RMSE): 18.092
R-squared: 0.726
```

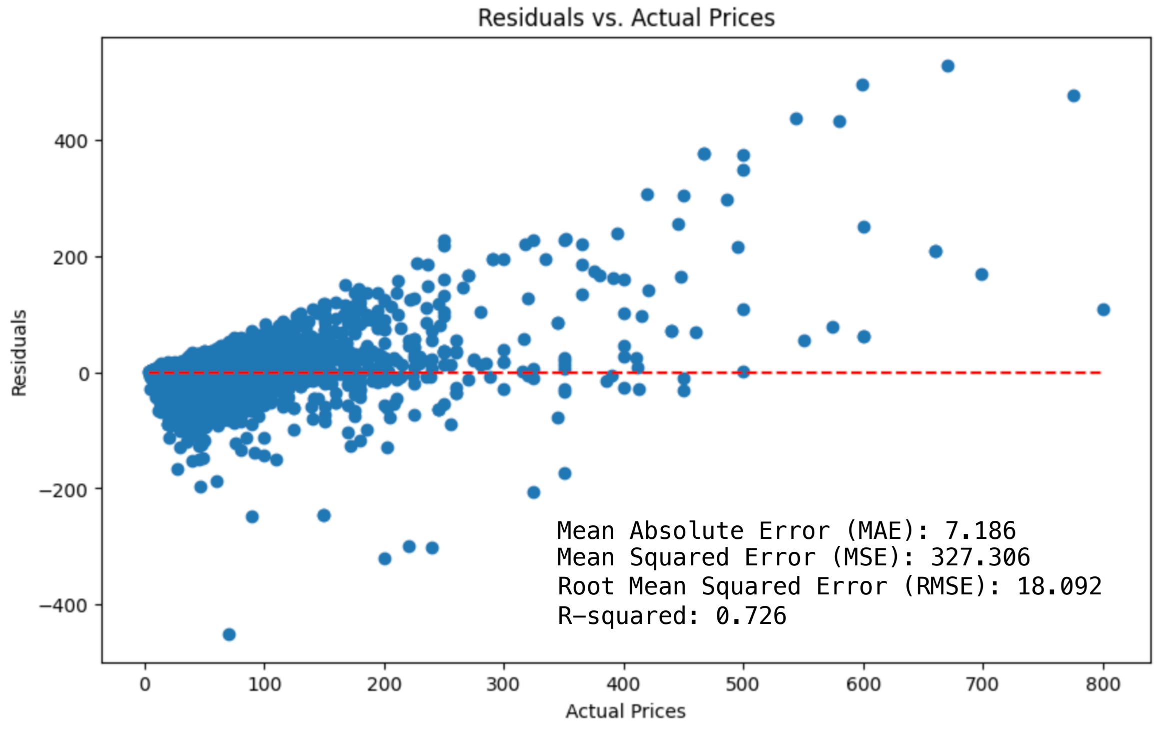
Results and Learnings

Following our baseline model we ran Optuna to hyper-tune our parameters based on the below

- optimizer: Chooses between 'adam' and 'rmsprop'.
- neurons: Chooses an integer between 64 and 256.
- dropout_rate: Chooses a float between 0.3 and 0.7.

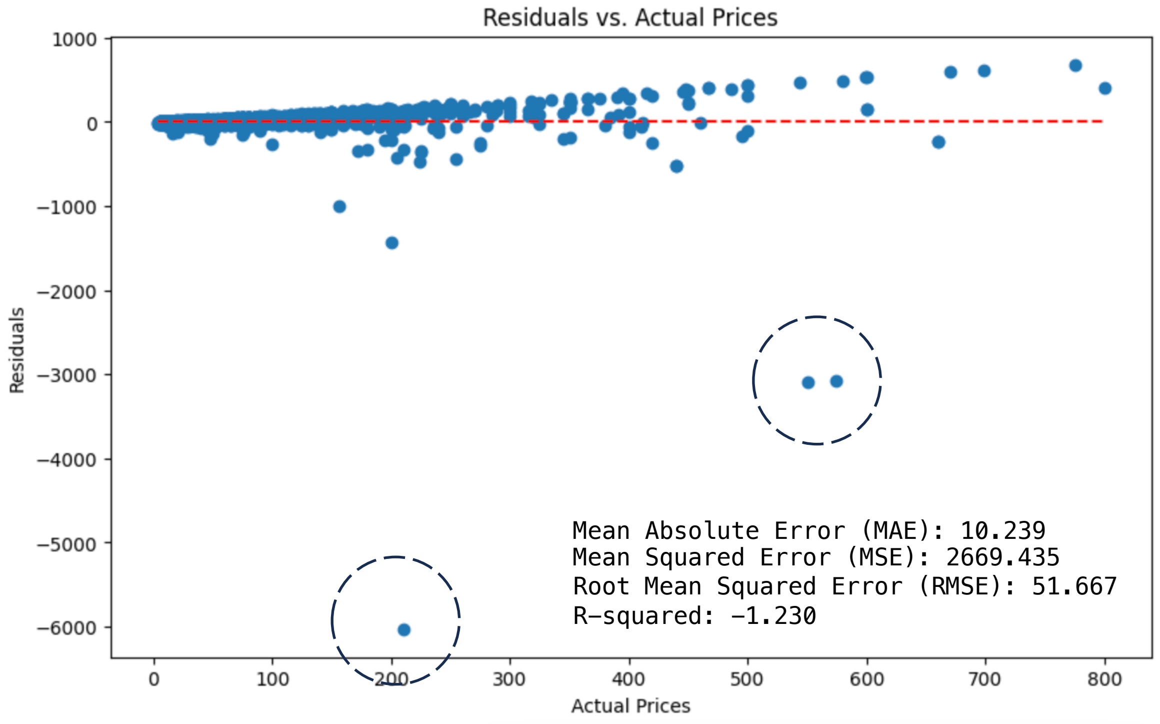
Despite some of the anomalies which visually "squash" our plot, on all accuracy dimensions, our hyper-tuned model under-performed our baseline model

Baseline Model (Champion)



```
Mean Absolute Error (MAE): 7.186
Mean Squared Error (MSE): 327.306
Root Mean Squared Error (RMSE): 18.092
R-squared: 0.726
```

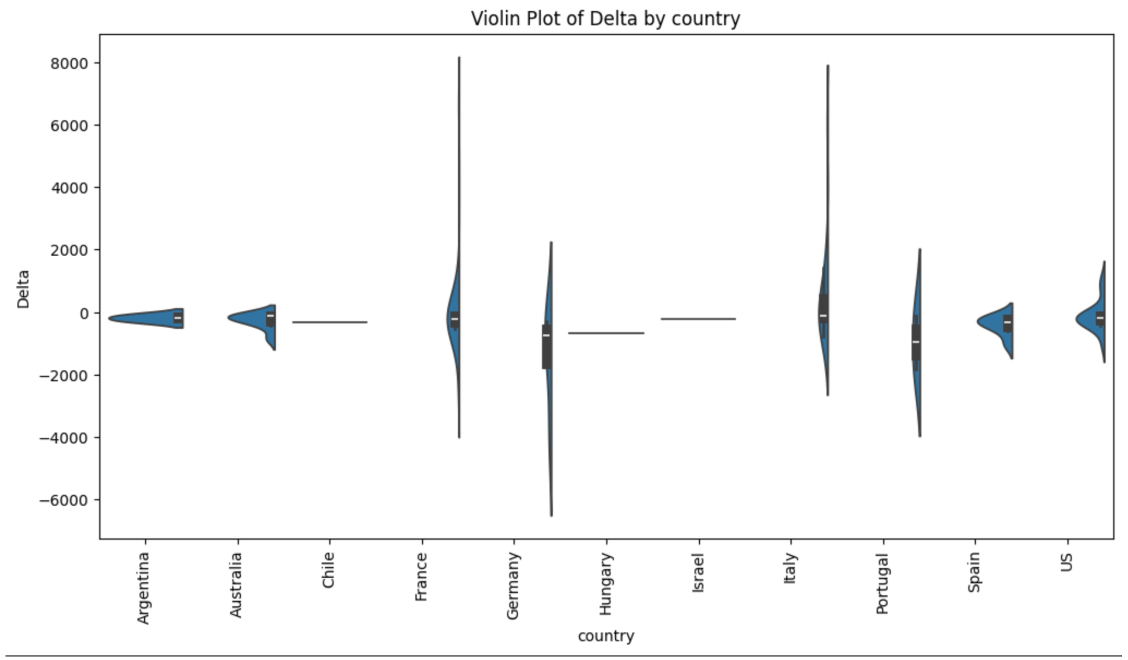
Hyper-Tuned Model



```
Mean Absolute Error (MAE): 10.239
Mean Squared Error (MSE): 2669.435
Root Mean Squared Error (RMSE): 51.667
R-squared: -1.230
```

So what’s a good buy?

- We attempted to normalize our results for objectivity ‘Average’
- From a pure volume game, we have a lot of good buys in the French and Italian producers
- Albania, however, based on capping our data frame to the less expensive wins, is showing the greatest single opportunity for savings!



country	Actual Price	Predicted Price	Delta	Wine count	Average
Albania	$20.00	$84.10	$64.10	1	$64.10
France	**$3,818.00**	**$5,376.08**	**$1,558.08**	**26**	**$59.93**
Lebanon	$50.00	$166.37	$116.37	2	$58.18
Italy	$1,536.00	$2,452.73	$916.73	16	$57.30
Brazil	$15.00	$66.19	$51.19	1	$51.19
Chile	$101.00	$252.65	$151.65	3	$50.55
US	$1,251.00	$2,222.19	$971.19	20	$48.56
Australia	$55.00	$197.58	$142.58	3	$47.53
Spain	$202.00	$432.84	$230.84	5	$46.17
Germany	$100.00	$326.62	$226.62	5	$45.32
Japan	$24.00	$69.14	$45.14	1	$45.14
Hungary	$25.00	$66.68	$41.68	1	$41.68
Argentina	$66.00	$107.58	$41.58	1	$41.58

Future Work

1. **More, smarter feature selection**
   - Features like region 2, which had upwards of 60% Null
2. **Target variable bin segmentation**
   - Because of the excessive increase in residuals as the price points of the wine get higher, I think it might be best if we break the model down into ”higher” and “lower” price points
3. **Hyper-parameter tuning, tuning**
   - Work out where in my hyper parameter tuning we fell apart (e.g. instead of using RMSProp and Adam as optimizers, I could have toggled between ReLU or GeLU
4. **“Good buy” bin segmentation**
   - Break down “good buy” by price point with unique thresholds
5. **Try a different model!**
   - ;alksjd;flakjsdf;lkajsd;flkj

THE UNIVERSITY OF
CHICAGO

THANK YOU