Machine Learning Predictions for *WineEnthusiast* Review Scores

DSCI-631

Winter 2022

Group: 03

Team: Alec Peterson

Wine Reviews Dataset

129,971 wine reviews scraped from WineEnthusiast in 2017, available on Kaggle

Variable	Significance
price	Price of wine, in dollars ($)
description	Natural language text of review
country	Country of origin
province	Province within country (e.g. California)
region_1	Region within province (e.g. Napa Valley)
region_2	Sub-region, if applicable (e.g. California Other)
taster_name	Name of reviewer
taster_twitter_handle	Twitter handle of reviewer
title	Title for wine review (including year)
designation	Vineyard with the winery where the grapes that made the wine are from
variety	Grape variety (e.g. Pinot Noir, Red Blend)
winery	Winery name
points	Point score for review from 0 – 100, though scores ranged from 80 – 100 in practice.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 129971 entries, 0 to 129970
Data columns (total 13 columns):
 #   Column                 Non-Null Count   Dtype
---  ------                 --------------   -----
 0   price                  120975 non-null  float64
 1   description            129971 non-null  object
 2   country                129908 non-null  object
 3   province               129908 non-null  object
 4   region_1               108724 non-null  object
 5   region_2               50511 non-null   object
 6   taster_name            103727 non-null  object
 7   taster_twitter_handle  98758 non-null   object
 8   title                  129971 non-null  object
 9   designation            92506 non-null   object
 10  variety                129970 non-null  object
 11  winery                 129971 non-null  object
 12  points                 129971 non-null  int64
dtypes: float64(1), int64(1), object(11)
memory usage: 12.9+ MB
```

Features Used in Models

- `price` has good correlation with points
- `description` reflects sentiments of reviewer
- `country` did not have too many unique values and thought still added relevant info

- Individual tasters (as reflected by `taster_name`) might tend to give a certain range of scores or descriptions
- Some tasters accounted for a large proportion of reviews

- `variety` reflects the wine type, not too many unique values as well
- `points` is the label to be predicted

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 129971 entries, 0 to 129970
Data columns (total 13 columns):
 #   Column                 Non-Null Count   Dtype
---  ------                 --------------   -----
 0   price                  120975 non-null  float64
 1   description            129971 non-null  object
 2   country                129908 non-null  object
 3   province               129908 non-null  object
 4   region_1               108724 non-null  object
 5   region_2               50511 non-null   object
 6   taster_name            103727 non-null  object
 7   taster_twitter_handle  98758 non-null   object
 8   title                  129971 non-null  object
 9   designation            92506 non-null   object
 10  variety                129970 non-null  object
 11  winery                 129971 non-null  object
 12  points                 129971 non-null  int64
dtypes: float64(1), int64(1), object(11)
memory usage: 12.9+ MB
```

Unused / Not useful features

- Too many unique values that could not be easily mapped or grouped together
- `region_2` has too many nulls

- `taster_twitter_handle` redundant with `taster_name`

- Too many unique values or nulls
- `title` offered redundant information, and year extracted from `title` did not have significant correlation with `points`

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 129971 entries, 0 to 129970
Data columns (total 13 columns):
 #   Column                 Non-Null Count   Dtype
---  ------                 --------------   -----
 0   price                  120975 non-null  float64
 1   description            129971 non-null  object
 2   country                129908 non-null  object
 3   province               129908 non-null  object
 4   region_1               108724 non-null  object
 5   region_2               50511 non-null   object
 6   taster_name            103727 non-null  object
 7   taster_twitter_handle  98758 non-null   object
 8   title                  129971 non-null  object
 9   designation            92506 non-null   object
 10  variety                129970 non-null  object
 11  winery                 129971 non-null  object
 12  points                 129971 non-null  int64
dtypes: float64(1), int64(1), object(11)
memory usage: 12.9+ MB
```

NLP Feature Engineering

NLTK Positive and Negative Word Lexicons

Positive Words

```
0              a+
1          abound
2         abounds
3       abundance
4        abundant
             ...
2001     youthful
2002         zeal
2003       zenith
2004         zest
2005        zippy
```

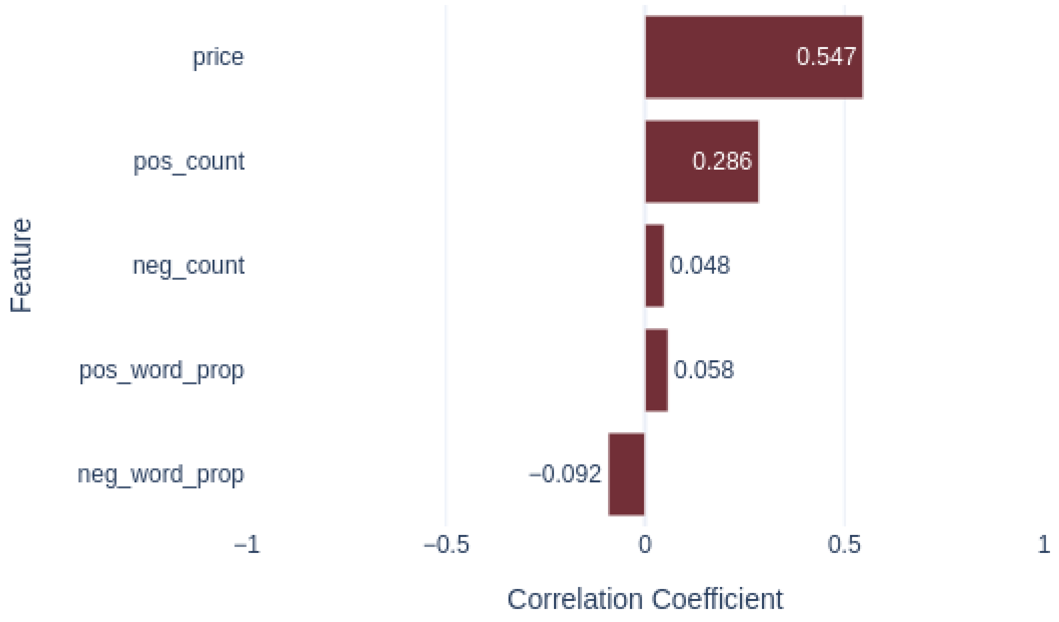
Negative Words

```
0          2-faced
1          2-faces
2         abnormal
3          abolish
4       abominable
               ...
4778          zaps
4779        zealot
4780       zealous
4781     zealously
4782        zombie
```

Lemmatization of `description` using `spacy`

	desc_lemmas
0	[ripe, fruity, wine, smooth, structured, firm,...
1	[tart, snappy, flavors, lime, flesh, rind, dom...
2	[pineapple, rind, lemon, pith, orange, blossom...
3	[like, regular, bottling, comes, rough, tannic...
4	[blackberry, raspberry, aromas, typical, navar...
...	...
117019	[notes, honeysuckle, cantaloupe, sweeten, deli...
117020	[citation, given, decade, bottle, age, prior, ...
117021	[drained, gravel, soil, gives, wine, crisp, dr...
117022	[dry, style, pinot, gris, crisp, acidity, weig...
117023	[big, rich, dry, powered, intense, spiciness, ...

Correlations for `price` and `description`-derived features



Linear Regression – Baseline Predictions

Features:

Numerical	Categorical
`price`	`country`
`pos_word_count`	`taster_name`
`neg_word_prop`	`variety`

Constraints:

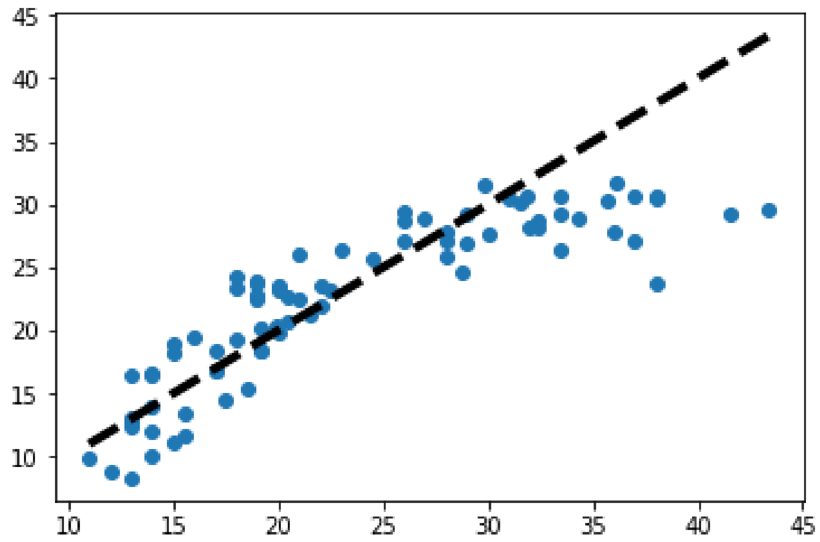
- Filter `price` to < $100:
  - Represents 90% of data (10% is $100 - $3300)
  - Correlation is strongest when filtered to this
  - Realistic upper limit for what someone might spend
- Remapped to "Other" to reduce problems when splitting:
  - countries with <10 wines
  - Varieties with count <20
- Split: 80 - 20 for Train – Test
- Stratify on `price` (0 - 20, 20 - 40, etc.)

Pipeline:

- Simple Imputer (Median)
- Standard Scaler
- One-Hot Encoder

Model: Linear Regression (unoptimized)

➔ **RMSE of 10-fold Cross-Validation (Benchmark): 2.31**



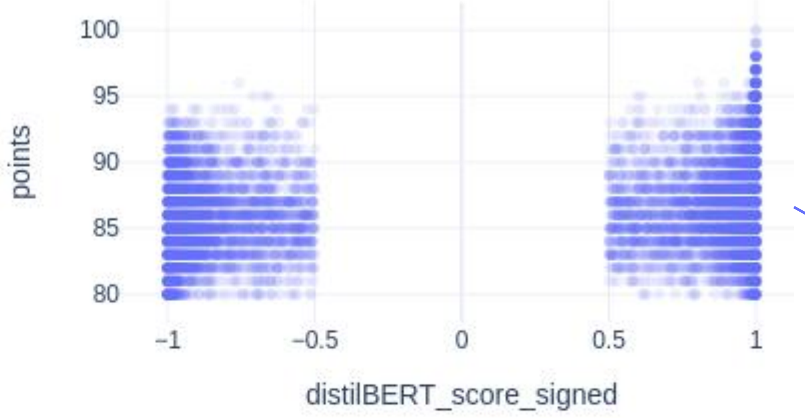
NLP Feature Engineering with 🤗

- Needed to better capture sentiment, context, and nuance of `description` natural language
- From research, best performance would likely come from pre-trained transformer models
- Hugging Face has publicly available models via the `transformers` module for sentiment analysis, with the ones used in this project derived from Bidirectional Encoder Representations from Transformers (BERT) models:

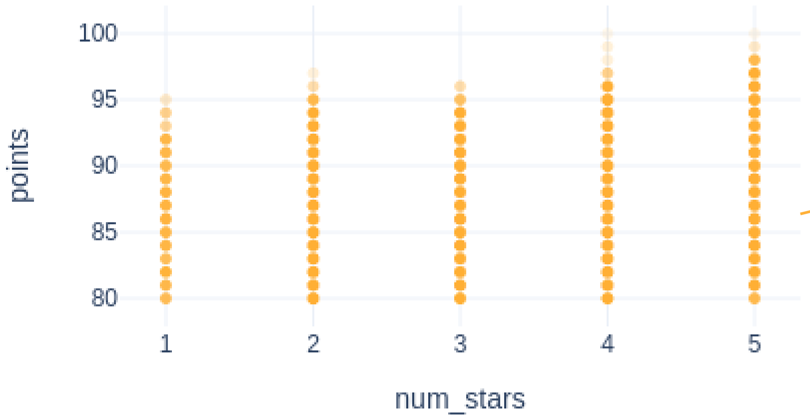
Model	Output
distilbert-base-uncased-emotion	"Positive" or "Negative" label and associated score
bert-base-multilingual-uncased-sentiment	"Score out of 5" label (like the number of stars for a customer review), and associated score

NLP Features – Correlations with `points`

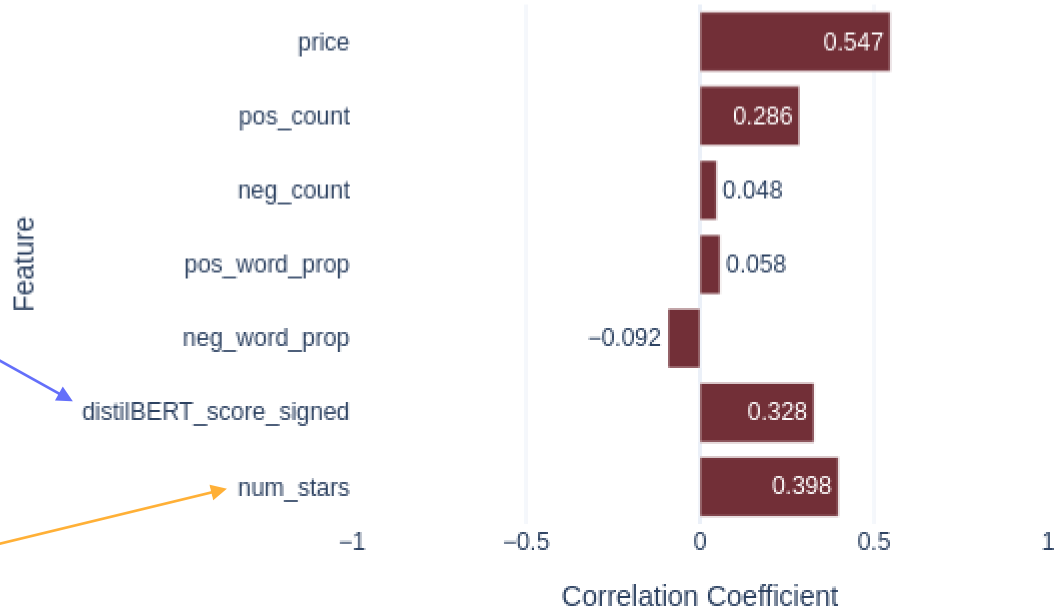
Remapping 0 – 1 score for “Negative” labels from [distilbert-base-uncased-emotion]()



Number of stars from [bert-base-multilingual-uncased-sentiment]()



`points` correlations with `price` and NLP features:



NLP Features – Correlations with `points`

`points` vs.
`log1p(price * num_stars)`

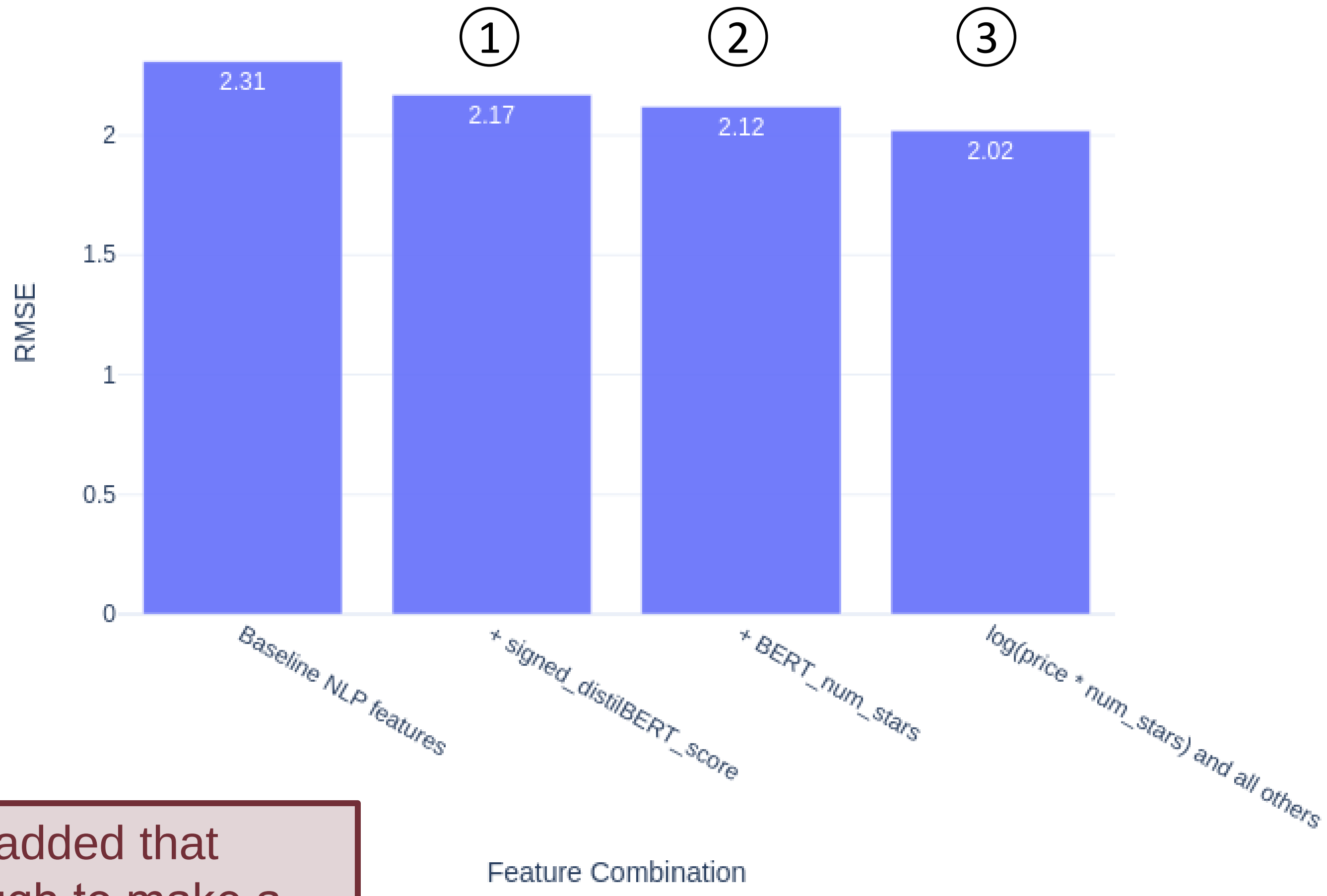
- `price * num_stars` seemed to make sense as a feature to magnify the effect of both features
- `log()` adds unique relation to reduce redundancy, determined from slightly stronger correlation observed between `log(price)` and `points`

`points` correlations with `price` and NLP features:

Linear Regression (unoptimized) with Various NLP Feature Combinations

Feature Type	Feature
Numerical	price
	pos_word_count
	neg_word_prop
	1 distilBERT_score_signed
	2 num_stars
	log(price * num_stars)
Categorical	country
	taster_name
	variety

3



Notable improvement as more features added that better capture description…but not enough to make a difference once scores are rounded to nearest integer.

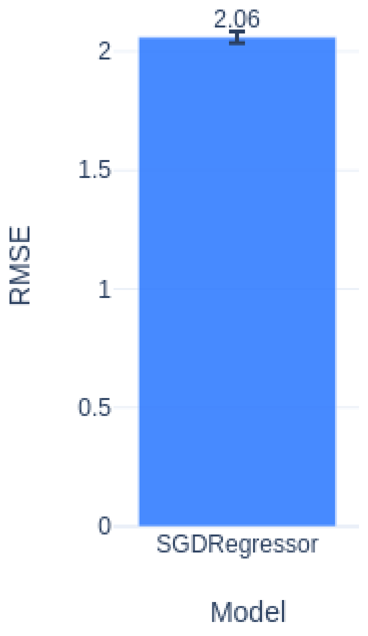
Machine Learning Models

ML Models Tested

- **Stochastic Gradient Descent (SGD) Regressor**
  - sklearn.linear_model.LinearRegression() does not have tuning parameters…
  - Linear models with ability add bias via Ridge, Lasso, Elasticnet penalties and other tuning parameters to potentially improve performance
- **Support Vector Machines (SVM) Regressor**
  - Capture potential nonlinearity
  - For this dataset, faster to test and tune than Random Forest (despite size)
- **Neural Network**
  - Further capabilities to capture nonlinearity
  - Implement learnings with Keras / Tensorflow
  - Personal opportunity to implement CUDA and personal computer's GPU ☺

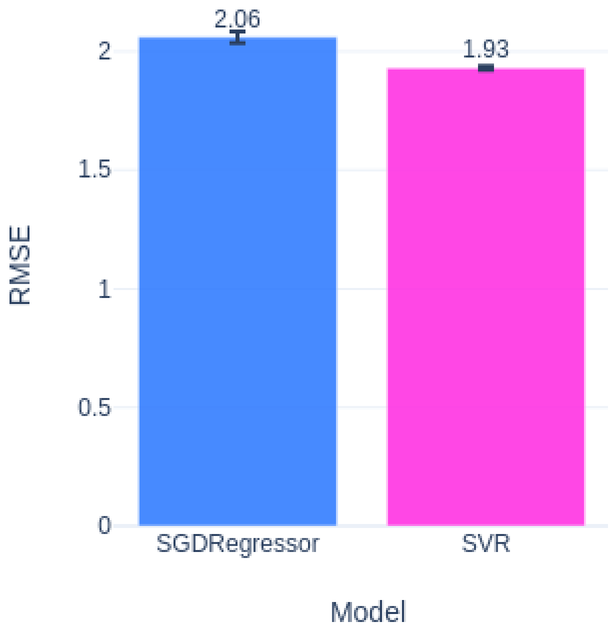
SGD Regressor

Model	Parameter	Tuned Value
SGDRegressor	alpha	0.001
	learning_rate	"constant"
	penalty	"elasticnet"



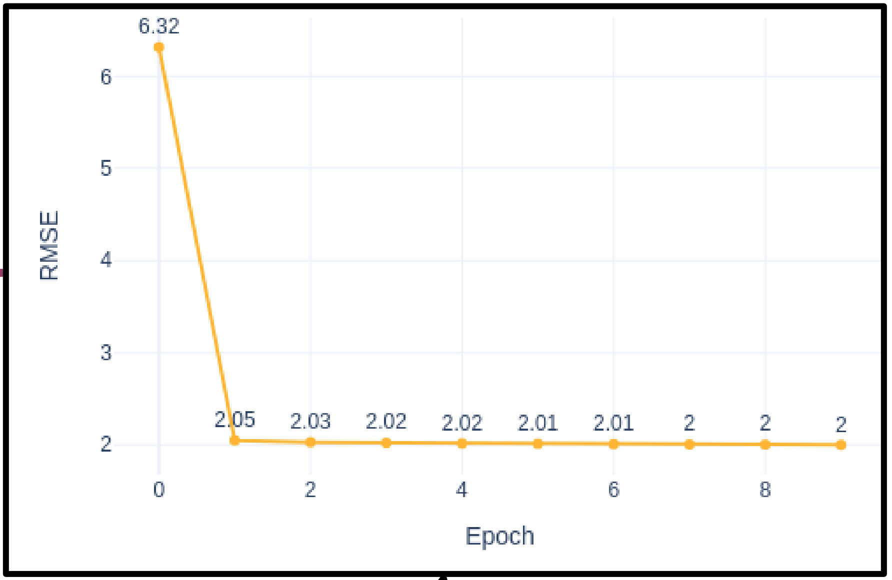
SVR

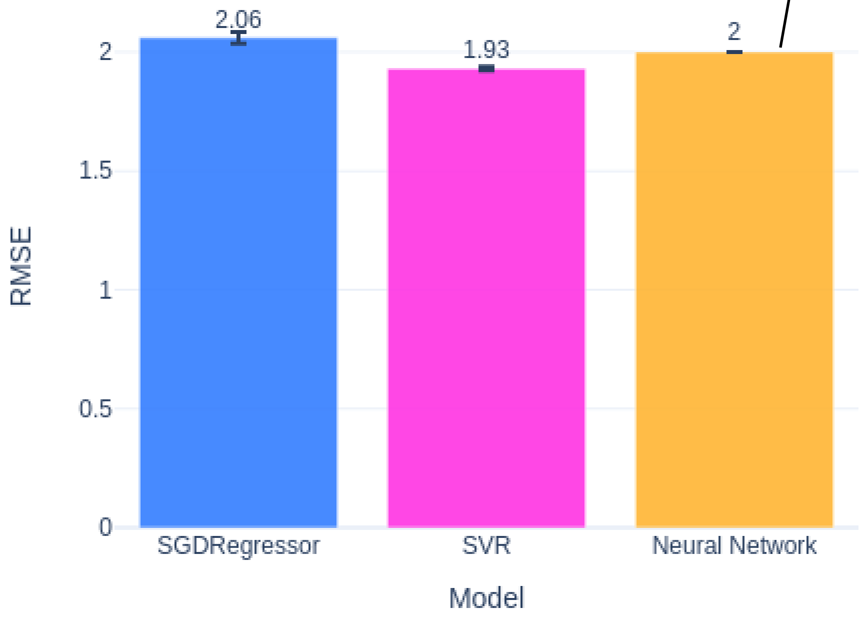
Model	Parameter	Tuned Value
SGDRegressor	alpha	0.001
	learning_rate	“constant”
	penalty	“elasticnet”
SVR	**C**	**1**
	Kernel	**“rbf”**



Neural Network

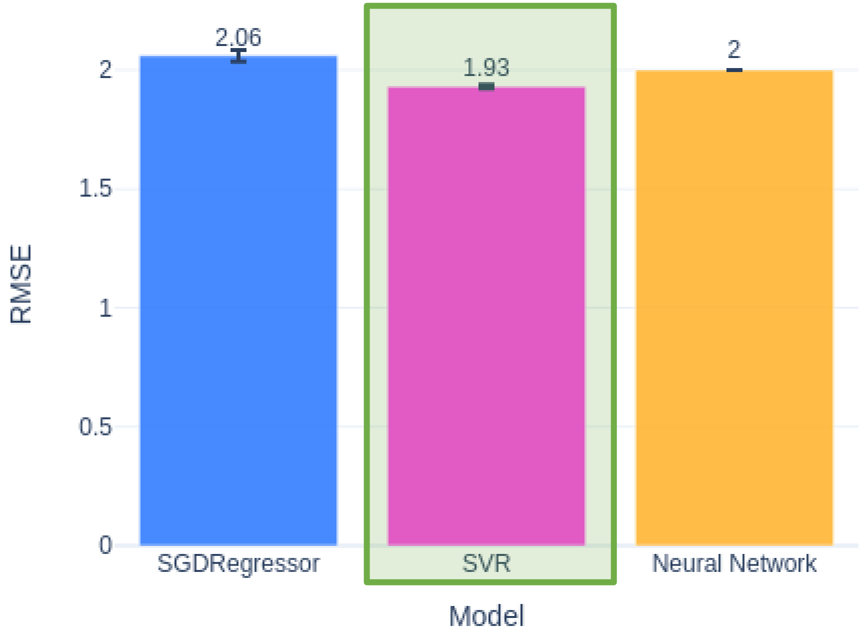
Model	Parameter	Tuned Value
SGDRegressor	alpha	0.001
	learning_rate	"constant"
	penalty	"elasticnet"
SVR	C	1
	Kernel	"rbf"
Neural Network (Dense)	**Hidden Layer**	
	# neurons	**50**
	activation	**"relu"**
	Output Layer	
	# neurons	**1**
	optimizer	**SGD (lr=0.001)**
	epochs	**10**



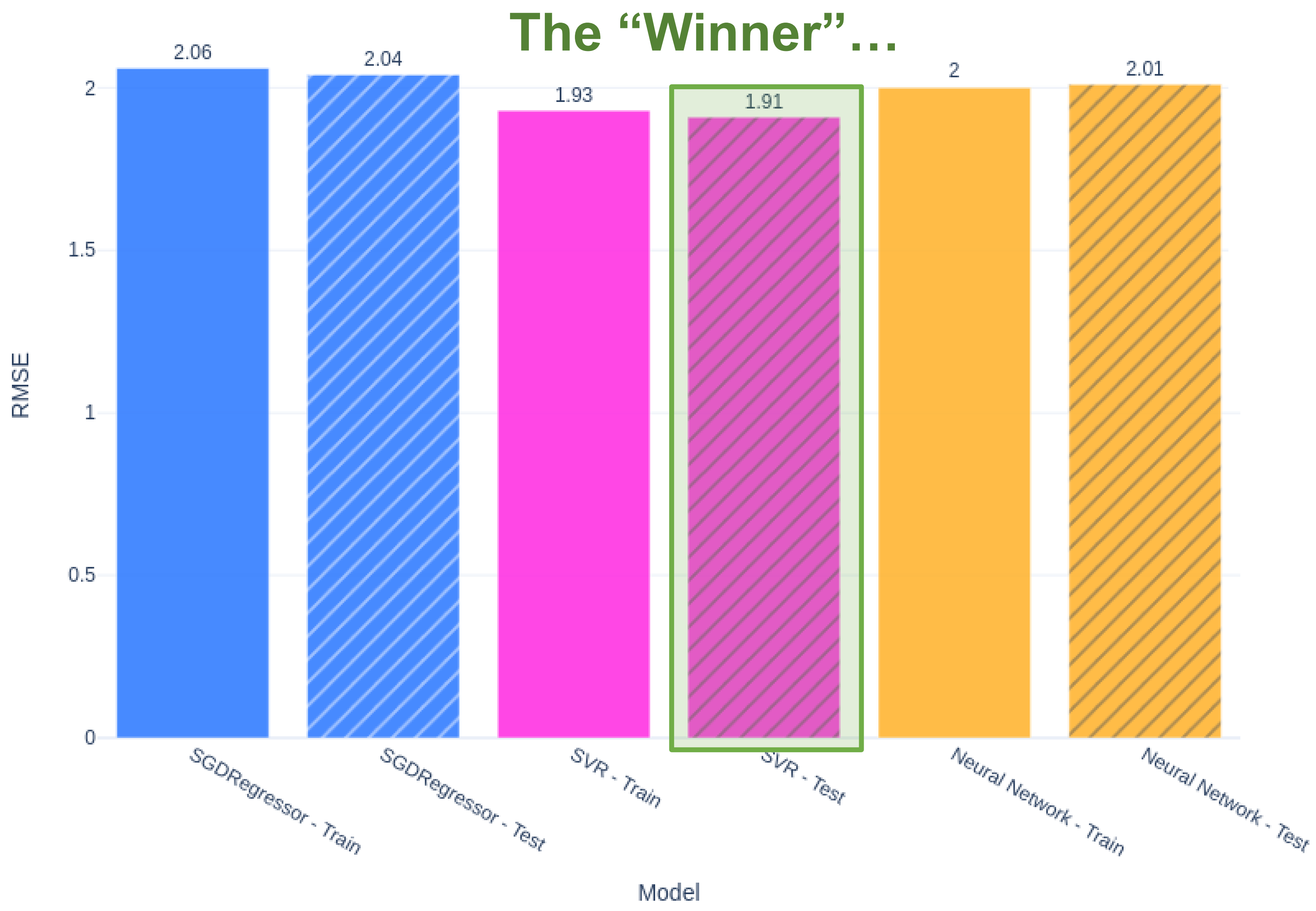


SVR is technically "best"

Model	Parameter	Tuned Value
SGDRegressor	alpha	0.001
	learning_rate	"constant"
	penalty	"elasticnet"
SVR	C	1
	Kernel	"rbf"
Neural Network (Dense)	Hidden Layer	
	# neurons	200
	activation	"relu"
	Output Layer	
	# neurons	1
	optimizer	SGD (lr=0.001)
	epochs	15



Test Set Performance



Conclusions, Limitations, Future Directions

- **Available data is perhaps too limited to achieve higher accuracy, though more sophisticated feature engineering may help**
  - Scores assigned are also subjective to the taster and perhaps hard to do on a 100-point (or even 20-point ranged) scale
  - Feature engineering I think will lead to significantly better performance.
- **Wine-specific lexicons**
  - Lower scoring reviews contained words for undesirable flavors (e.g. “chemical”, “vegetal”, “unripe”, “sugary”) that wouldn’t be captured in normal sentiment or emotion lexicons
  - Dictionaries with desired flavor notes for a given variety could be applied, with similarity to these “vectors” or embeddings represented by these words as a feature
- **Food & Beverage transformer models**
  - Transformer models trained on customer reviews for restaurants, bars or more specifically wine bars / vineyards / wine websites would improve model performance and sentiment scoring
  - Practice seems to be using manually labeled datasets to verify performance…
- **Implement Linear Regression or other linear model in practice**
  - The unoptimized linear regression had comparable RMSE to the more complex models, but computed much faster and with fewer resources than SVM or Neural Network
  - More interpretable and easily understood

Machine Learning Predictions for *WineEnthusiast* Review Scores

DSCI-631

Winter 2022

Group: 03

Team: Alec Peterson