I. Introduction

As much as we all love our wines, beers, ciders, vodkas, brandies, and all types of beverages, they can absolutely come at a steep cost, and the massive variety of seemingly identical options for each type of drink can be highly overwhelming for any casual consumer. Rather than try the costly trial-and-error method of buying new drinks and screening out which is the "best," an interested consumer can point to the advent of predicting machine learning methods to make an easy and more heavily-informed decision. For our project, we're analyzing whiskey ratings through machine learning methods in order to predict the best overall whiskey options available.

In this report, we've analyzed a data set containing numerous ratings of different types of whiskey in Section II; we've compared the results of two widely accepted machine learning methods in predicting the ratings of said whiskeys in Section III; we implemented these methods in Python and compared the resulting graphs to determine which is the more accurate method for our purposes in Section IV; and we came to a conclusion regarding our findings in section V.

II. Problem Formulation

Our problem is a question of how we can predict the overall score of whiskey based on its price. The data we're using comes from the site Whiskey Advocate, being scraped from the site in 2018. The data in question contains approximately 2,200 reviews of various whiskeys, containing information on each whiskey's name, alcohol percentage, review score, price per bottle, and text from the review summarizing the reviewer's thoughts on said whiskey. The scores were determined in four parts, following a common whiskey-rating system that scores a given whiskey based on four factors – nose, taste, finish, and body – each of which can reach a maximum of 25 points for a total of 100 points. For our purposes, we'll only be factoring in the final score out of 100, and we'll be using each whiskey's per-bottle price in our predictions. We've also placed a cap of $800 on the prices of the whiskeys, due to a few massive outliers that skewed the predictions too much for our liking. This reduces the dataset to 2,075 data points in total.

III. Methods

As stated previously, we're utilizing a data set of roughly 2,075 data points, with each data point being a specific type of whiskey. Each of these data points (whiskeys) has the features of price (in USD) and category (see part II). The labels, or what we end up predicting through this process, are the scores for each whiskey, given in the data table. Thus, our goal is to see how accurately we can predict a given whiskey's score based on its price and category alone. The only preprocessing done was dropping some of the columns containing information we don't need, such as review texts and currency. We selected our features, the price and category of the whiskey, as a means of testing the general assumption that the price and quality of a given product are directly correlated, as well as analyzing the data to see which categories of whiskey typically received lower scores; the grain whiskeys and Single Scotch whiskeys often showed up towards the bottom of the ranked data table, while Blended Malt Scotch whiskeys were very prevalent towards the top.

Regarding model validation, we split our data into three categories – training data, validation data, and test data – using the sklearn library's train_test_split() function, which, for each of our feature arrays, we pass the array itself and the percentages, as decimals, that we'd like to use for each of the three aforementioned data types. We've chosen to use the relatively standard amount of 80% for the size of the training data, with the remaining 20% being split between the validation data and the test data, both of which are 10% of the original data.

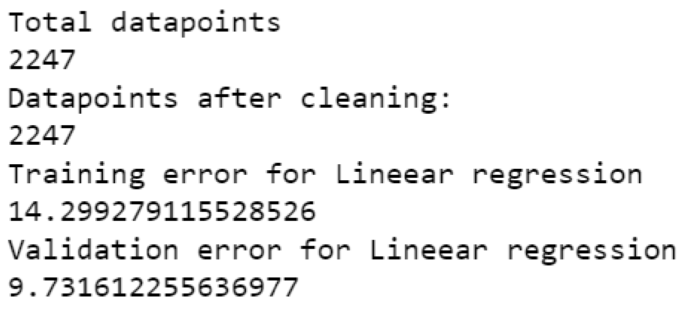
i. Linear Regression

To test our features, we selected Linear Regression as our first ML method. The reason for this choice is that we can easily and accurately obtain linear plots for our two features individually, obtaining straightforward cause-and-effect graphs. From there, it's easy to compare the results in these graphs to the actual scores given in the dataset. The use of Linear Regression deems this to be a supervised learning problem. Following this choice of ML method, we're using squared error as our loss function of choice. This is because squared error is typically utilized alongside Linear and Polynomial regression in machine learning practices; this loss function is also included in the Linear_Regression() code package in the Scikit-learn library for Python, via the function mean_squared_error(). This function takes in two parameters, the first being the actual whiskey scores we're trying to predict and the second being the predicted scores; naturally, the predicted scores are found through the Linear_Regression() package's predict() function, which takes in an array of features as its sole parameter. For our purposes, we're testing two separate feature arrays, one containing the prices of each whiskey, and the other containing the category of each whiskey; we're running a test for each of these arrays, where the array in question is fed into the predict() function, and then we use the mean_squared_error() function to compare the array of whiskey scores to the result of the predict() function.

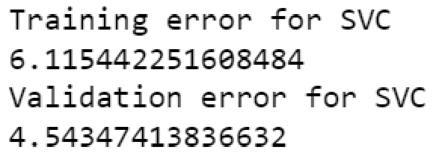
ii. Support Vector Classification

The second method we used to test our features was Support Vector Classification (SVC). We chose this method due to its use in regression problems and with linear data. In particular, SVC is used in classification problems, as its name would suggest; in our problem, we can consider the whiskey ratings to be the categories. Considering our relatively linear data, the use of linear maps on transformed data (our training set) should be fairly accurate as a result. The use of SVC deems this to be a supervised learning problem. Accordingly, we're using hinge loss as our loss function. This choice was made because not only is hinge loss commonly used in classification problems such as this one, but it emphasizes boundary points and determines error based on misclassified samples, or samples outside of said boundaries. The hinge loss function is included in Scikit-learn's make_pipeline() function. This function takes in two parameters, the StandardScaler() class and the SVC class, the latter using gamma='auto' as a parameter. We're using the same feature arrays as in the Linear Regression section, and using them in a similar fashion. As a side note, we originally considered using lasso or ridge regression, but we didn't see any noticeable changes in our tests, with some results being the exact same as with the previous Linear Regression/squared error configuration.

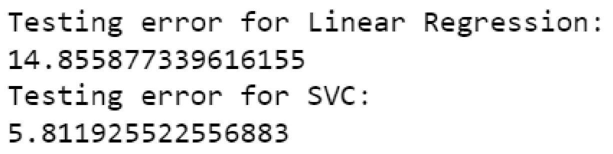
IV. Results



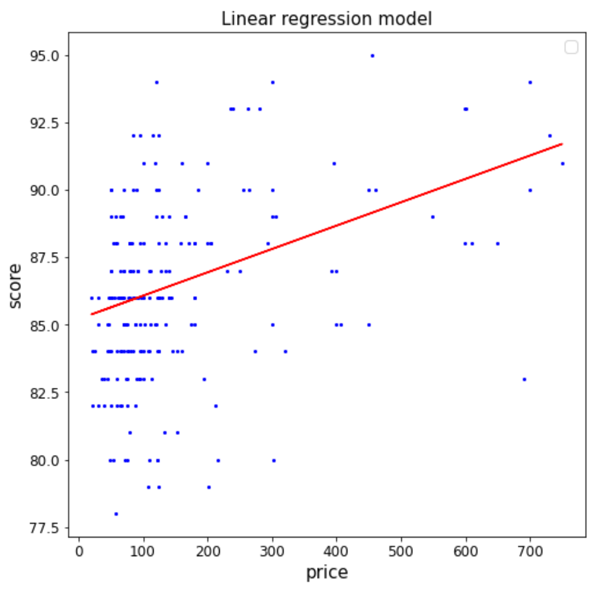
```
Total datapoints
2247
Datapoints after cleaning:
2247
Training error for Lineear regression
14.299279115528526
Validation error for Lineear regression
9.731612255636977
```

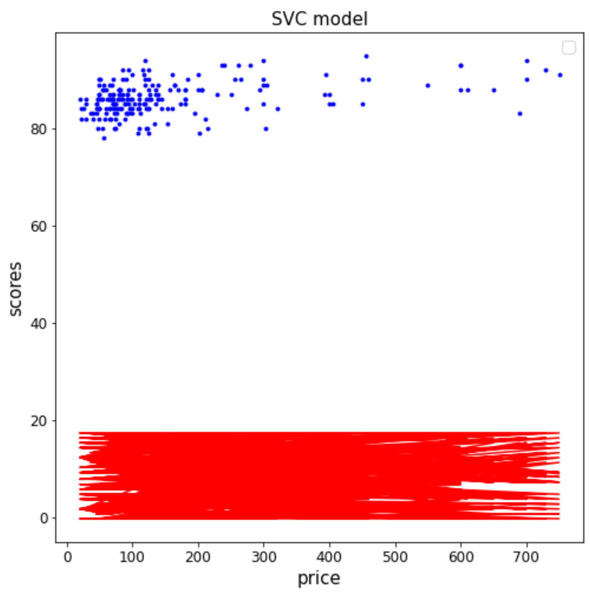


```
Training error for SVC
6.115442251608484
Validation error for SVC
4.54347413836632
```



```
Testing error for Linear Regression:
14.855877339616155
Testing error for SVC:
5.811925522556883
```





As seen above, the training, validation, and testing errors for the linear regression model were 14.299, 9.732, and 14.856 respectively, rounded to three decimal places. For the SVC model, these errors were 6.115, 4.543, and 5.812 respectively, also rounded to three decimal places. This immediately indicates that the SVC model provides considerably less error in its predictions than the linear regression model, which came as a surprise to us, as we had expected our data to deliver a smooth linear line-of-best-fit. Of course, the actual results of these predictions vary between runs, as there's always some amount of variation in the predictions; in addition, it's worth noting that we had trouble plotting our SVC model (as is clearly shown above), though the error results were computed without issue. Touching on our test set, this set was the remaining 10% of our data after splitting 80% into training and 10% into validation sets; it wasn't used for training the methods or validating our chosen method.

V. Conclusion

Judging by the results of these two predictive methods, the SVC model was the best fit for our ML problem, yielding the lowest error. Part of why this may be the case is that the linear regression model is heavily skewed by outliers and spread-out data sets, and even with the adjustments we made to our data, the linear model still couldn't fit all too accurately on our data, at least in comparison to a more nuanced model like SVC.

In a future application of this problem, the results could be much more accurate by factoring in more aspects of the whiskey; price is obviously not the only indicator of a whiskey's quality or its score, but the whiskey's ingredients, alcohol content, and style can also play a big role in defining what the whiskey tastes like and how "good" it is. A future experiment could benefit from factoring these in, as well as by creating more defined categories for the SVC model. For example, since SVC typically handles categorical data, dividing whiskeys into categories based on their specific “type” or “style” of whiskey could play a part in obtaining a more clear-cut graph of the data, or deliver more easily understandable error results.