Stat 602 Final Project

Group 3: Tara Burton, Pam Davis, John Herbert, Kara Paulson, Snigdha Peddi

Introduction

Dry beans are widely produced all over the world. However due to their sensitivity to the climate and unavailability of certified beans for the cultivation a mixed variety of beans often result in unfair market prices. Previous studies have shown that computer vision and machine learning techniques can be used to measure morphological features of beans and accurately predict classes of beans from those features. For this study, multiple machine learning classification models were investigated to find a robust model that could expand on this area of study to predict the price of one pound of beans from various farms with accuracy. With these methods, farmers and consumers can feel more confident they are receiving a fair and equitable trade.

Data

The data for this problem included a training set with 3,000 observations of labeled bean classes. There are six varieties of beans and each bean type is represented 500 times within the dataset. The features include morphological characteristics of the beans. A detailed description of each feature can be found in the appendix A. An additional 3 samples of data are included, each set containing 1 pound of unlabeled classes of beans. Additional information was provided as to the price and weight of each bean class: Bombay ($5.56/lb, 1.92 g/seed), Cali (6.02/lb, 0.61 g/seed), Dermason (1.98/lb, 0.28 g/seed), Horoz (2.43/lb,0.52 g/seed), Seker (2.72/lb, 0.49 g/seed), Sira (5.40/lb,0.38 g/seed).

Model Building:

Before predicting the price per pound of beans, we first explored several models for accurately predicting the class of bean for each observation within the training/labeled data. We chose to fit neural network, multinomial logistic regression and support vector machine (SVM) models. Each model built used a 10-fold cross-validation for validation. The specific process of fitting each of these models is outlined in this section.

Model 1: Neural Network

Neural Network - All Variables

We first fit a neural network using scaled data. We used the caret package's train function and the nnet argument scale the data by converting each observation value in the training data set using $\frac{x-\mu}{\sigma}$, or the Min-Max Normalization. This also allowed us to run different decay rates, maximum iterations and number of nodes within the single hidden layer using the sigmoid function. The *nnet* model will only allow one hidden layer, therefore we were unable to test a more complex model with multiple hidden layers.

After hyper-parameterizing the model, we found that setting a decay rate of 0.001 produced 2 nodes within the hidden layer and returned the best results. The metrics that will be used to compare whether feature reduction are accuracy, sensitivity and specificity of the entire model. The overall Accuracy of the tuned model was 90.4% (Table 2).

Neural Network - Random Forest Variable Importance Features

We used a random forest for variable reduction to reduce the complexity of the neural network. The importance function from randomForest package measures the node impurity using Gini Index. A total decrease in impurity at each node from splitting on a variable is calculated and averaged over all trees. The variable with the largest Gini Index gets a higher score and is determined important to predict the classification.

According to the random forest, the variables *Solidity*, *Roundness*, and *SF4* have the least importance in the model with a large jump in scores from the 4th lowest feature, *Perimeter*. Therefore, we will test a neural network with the 3 least important features removed. The model with reduced variables performed slightly better with an overall with an accuracy rate of 90.6% (Table 4).

Model 2: Multinomial Logistic Regression Models

A Multinomial logistic regression was used to build a model using maximum likelihood estimation to evaluate the probability of each observation belonging to each class of bean. The *multinom* method of *nnet* package is used in the *train()* function of the *caret* package to build this model. We compared accuracy, sensitivity and specificity metrics from the two logistic regression models built on two different sets of variables

Multinomial - All Variables

The first model used all variables from the paper, including the additional calculated interaction terms. Using the 10-fold cross-validation method as with the neural networks, the model was fit to the training data. It performed with 90.3% accuracy (Table 9), but did not converge. This is a sign that the data was not fitting the model well and may be overfit.

Multinomial Logistic Regression - Variable Reduction

The variables for the second model underwent a selection process to eliminate some of the complexity of the model. The method used was to graphically explore what variables provided some separation between the classes. The first variable, area, appears to offer good separation of the Bombay and Cali seed varieties from the others. This variable is entered into the model. The other variables were selected based of the importance function using randomForest classifier. The final model is fit using Area, Minor Axis Length, Eccentricity and Compactness to explain which class the bean belonged.

When examining the bean class accuracy, the cross-validated model correctly predicted the class of bean approximately 90.2% of the time (Table 11).

Model 3: Support Vector Machine (SVM)

An SVM model was created using the same methodology as the first two models to determine a comparable cross-validated accuracy rate for all the training dataset. This model was fit in the caret package using the *svmlinear2* method.

SVM - All Variables

For the caret *train* model using SVM, the only hyper-parameter that can be adjusted is cost. We reviewed multiple values for cost and the best fit result was observed at 0.5. According to the results of our cross-validation, an accuracy score of 90.5% (Table 15) was acquired.

SVM - Random Forest Variable Importance Features

The overall accuracy of the random forest selected variable model was 90.5% (Table 17), the same as using all variables. Since there was no change to the model accuracy and also a reduction of complexity and potential noise, we chose the random forest importance test features to move forward in the analysis of SVM.

Accuracy of Class Predictions

Examining the accuracy of the bean classifications across the models resulted in similar results. The class accuracy generally improved in all three model types with the variable reduction model was fit. The actual results can be found in tables 5, 12, and 18.

Bombay and Cali beans are predicted correctly the most often. Sira and Dermason varieties are predicted correctly the least often between the bean predictions. The price difference between these varieties is quite large. When inaccurate predictions are made on these observations, the error is likely to increase at a higher rate than the other incorrect predictions.

Between Sample Distributions

Due to the equal distribution of bean class within the training data, we needed to be sure our models would maintain accuracy when faced with unequally distributed data. To accomplish this, a dataset of 3,000 observations from the training dataset was created by

randomly choosing probabilities that each class would be chosen. This resulted in a dataset with varying quantities of each class, which may be more generalizable to what would be expected with real data. The 3 models were then used to predict on this data and classification accuracy was calculated. Bootstrapping was used to resample 1,000 datasets using this method. All datasets had different class distributions, including some with 1 or more classes were not represented. The results are close to training accuracy with the models fit with balanced data (Table 25).

Predictions

After building the Neural Network, Multinomial Regression and SVM models, class predictions were made on the 3 one-pound bag samples.

Bean Class Predicted Sample Distribution

Our ultimate goal, to produce an estimated accuracy of the price of one pound of beans, required multiple predictions and steps. We first predicted the class of each observation using our 3 models. The observation was then labeled using the class with the maximum posterior probability. These predictions were used to determine the distribution of bean classes in each sample.

Sample A:

As with the accuracy of each bean type between the models, the distribution of predicted bean types were fairly homogenous across the sample A predictions. The sample was made up of approximately 3% Bombay, 46% Cali, 2% Dermason, 1.5% Horoz, 44% Seker and 3-4% Sira.

Sample B:

Across all of our models, the proportion of classes within Sample B is approximately comprised of 0% Bombay and Cali beans, 57-58% Dermason, ~0% Horoz, .17% Seker and 23-24% Sira. Sample B primarily consists of the Dermason and Sira beans and therefore we expected Sample B to have the largest margin of error between all the samples.

Sample C:

Sample C is comprised of mostly Horoz beans, making up 55-56% of the samples. Cali represents about 10%, Dermason 17%, Seker 1%, and Sira 17%. These are consistent across the 3 models. As with sample B, there is a decent proportion of Dermason and Sira beans which may skew the price estimates.

The actual proportions of each sample can be found in tables 6, 12, and 19.

Converting Actuals & Predictions in Price:

After predicting the class of each observation within the one-pound bag of beans, estimated price predictions were calculated. The estimated price of each observation was determined by multiplying the cross-validated posterior probability by the estimated price per seed of each

class of bean. From this predicted price, the sum of all observations were calculated to get the estimated price for the three samples. These price estimates can be found in Table 24.

Predicting 1 lb. Samples on Training Data Using Bootstrap w/ Replacement

To help measure our confidence in the accuracy of our predicted prices, we completed a simulation creating 1,000 one-pound bags of beans from training dataset via a stratified bootstrap. To accomplish this, one bean was randomly selected from the training data set. To better compare this with the sample, each draw was constrained to being picked based on the distribution of the predicted class of bean from the sample. For example, in sample A, each time a bean is drawn, there is a 2% chance of being a Bombay prediction, etc. Once the bean was selected, the predicted price of the bean (using the weighted price based off the posterior probabilities) was added to the total cost of the bag. During this process we also store the actual price of the labeled bean. Additionally, the weight of the bean, based on the predicted class, was added to the total weight of the bag. This bean was replaced back into the training data set and this process was repeated until the bag reached approximately one-pound. The total price of the sample was recorded.

Once the bootstrap was completed, the price of the 1,000 one-pound bags of beans were used to measure the 95% confidence interval of the price per pound of bean. We also calculated the MSE between the stored actual price of each observation and the predicted price of the observation. We calculated a 95% confidence interval on the squared errors of the bootstrap samples.

Model Selection for Test Predictions

Comparing the MSE for all 3 samples and all the samples together between all 3 of the models, neural network has the lowest error for samples A, B, and the combined samples of all the observations and SVM performed best on sample B. We would recommend making our predictions using the multinomial logistic regression even though the neural network and SVM had a lower error rates for the samples. The margin of difference in the performance is small, so the least complex model requiring the least amount of computational power was preferred.

Conclusion

Across all of our models, the accuracy of predicting the class of each bean observation was relatively stable. Due to the physical make-up of the beans, some of the varieties were more easily predicted than others. Unfortunately, the variables used to predict between the Dermason and Sira bean varieties did not provide great separation for an accurate prediction. Given the value difference with these beans, depending on the distribution of the of the class of beans within a sample, the price may vary widely. Additionally, although our bootstrap sample was normally distributed on the simulated data, the actual price of our predictions fell outside of the confidence intervals. Further exploration is needed to find a methodology used to estimate each price of a bean which may provide a more accurate estimated price.

Appendix

Table 1: Result Metrics of NNet - All Variables

	Variable	Importance
6	Eccentricity	410.9002
1	Area	372.3510
8	EquivalentDiameter	353.4616
3	MajorAxisLength	274.3295
13	SF1	242.3263
7	ConvexArea	186.1213
4	MinorAxisLength	135.1256
14	SF2	107.8561
5	AspectRatio	78.6286
15	SF3	76.2977
12	Compactness	74.0307
9	Extent	55.2089
2	Perimeter	48.8421
10	Solidity	30.7270
11	Roundness	28.1155
16	SF4	24.8426

	Metrics
size	2.0000
decay	0.0010
logLoss	0.2751
Accuracy	0.9040
Mean_F1	0.9041
Mean_Sensitivity	0.9040
Mean_Specificity	0.9808

Table 3: Accuracy by Class - All Variables

Class	Accuracy
BOMBAY	1.0000
CALI	0.9559
DERMASON	0.8631
HOROZ	0.9043
SEKER	0.9137
SIRA	0.7894

Table 4: Result Metrics of NNet - Random Forest Variables

	Metrics
size	2.000
decay	0.001
Accuracy	0.906
Mean_F1	0.906

	Metrics
Mean_Sensitivity	0.906
Mean_Specificity	0.981

Table 5: Accuracy by Class - Random Forest Variables

Class	Accuracy
BOMBAY	1.0000
CALI	0.9579
DERMASON	0.8611
HOROZ	0.9022
SEKER	0.9177
SIRA	0.7992

Table 6: Sample Prediction Bean Distribution

Class	A	B	C	All
BOMBAY	0.0284	0.0000	0.0000	0.0070
CALI	0.4652	0.0000	0.0998	0.1466
DERMASON	0.0206	0.5812	0.1670	0.3124
HOROZ	0.0129	0.0073	0.5560	0.1808
SEKER	0.4446	0.1748	0.0092	0.1897
SIRA	0.0284	0.2367	0.1680	0.1635

Table 7: Predicted Samples Results - Neural Network

Metric	A	B	C	All
Actual Mean Price	4.6463	3.0712	3.2649	10.6066
Predicted Mean Price	4.5379	3.2806	3.4395	10.8806
Lower Conf	4.4291	3.2173	3.3477	10.6182
Median	4.5366	3.2817	3.4399	10.8765
Upper Conf	4.6502	3.3430	3.5255	11.1797

Table 8: Predicted Samples Squared Error - Neural Network

Metric	A	B	C	All
Mean	0.0122	0.0448	0.0313	0.0774
Lower Conf	0.0046	0.0230	0.0154	0.0320
Median	0.0115	0.0437	0.0302	0.0748
Upper Conf	0.0227	0.0736	0.0536	0.1357

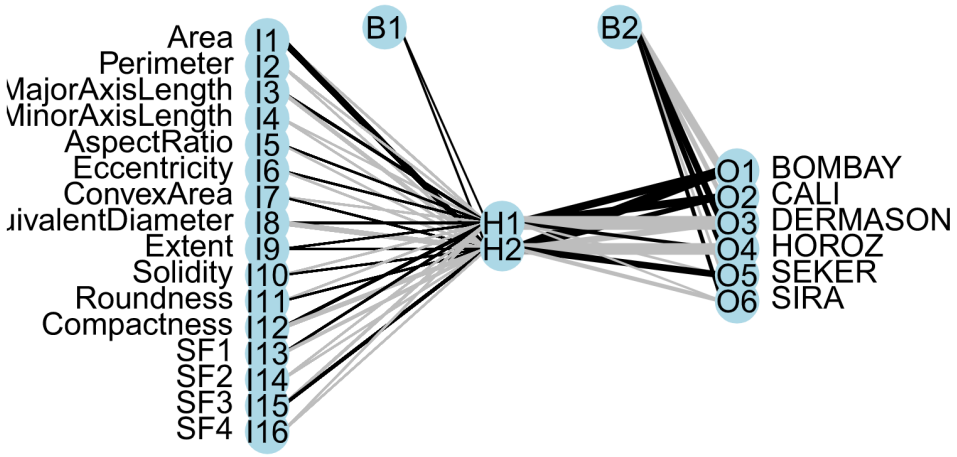

Figure 1: Neural Network - All Variables

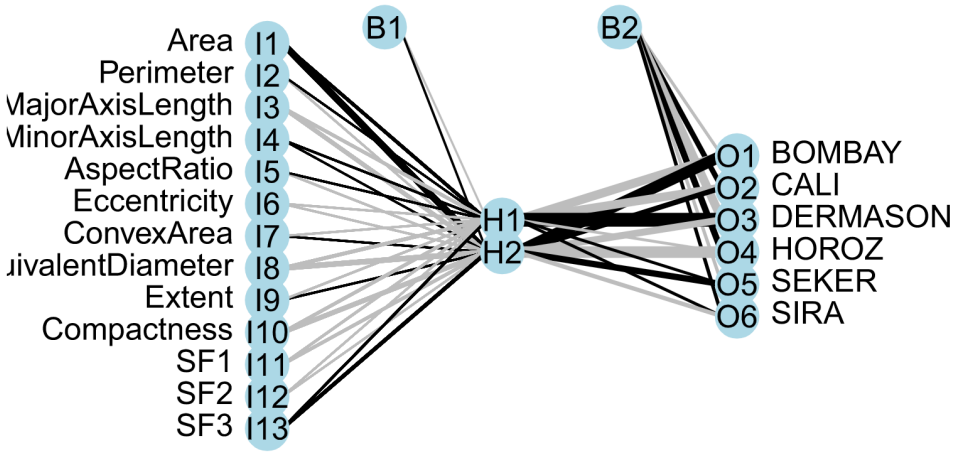

Figure 2: Neural Network - Random Forest Features

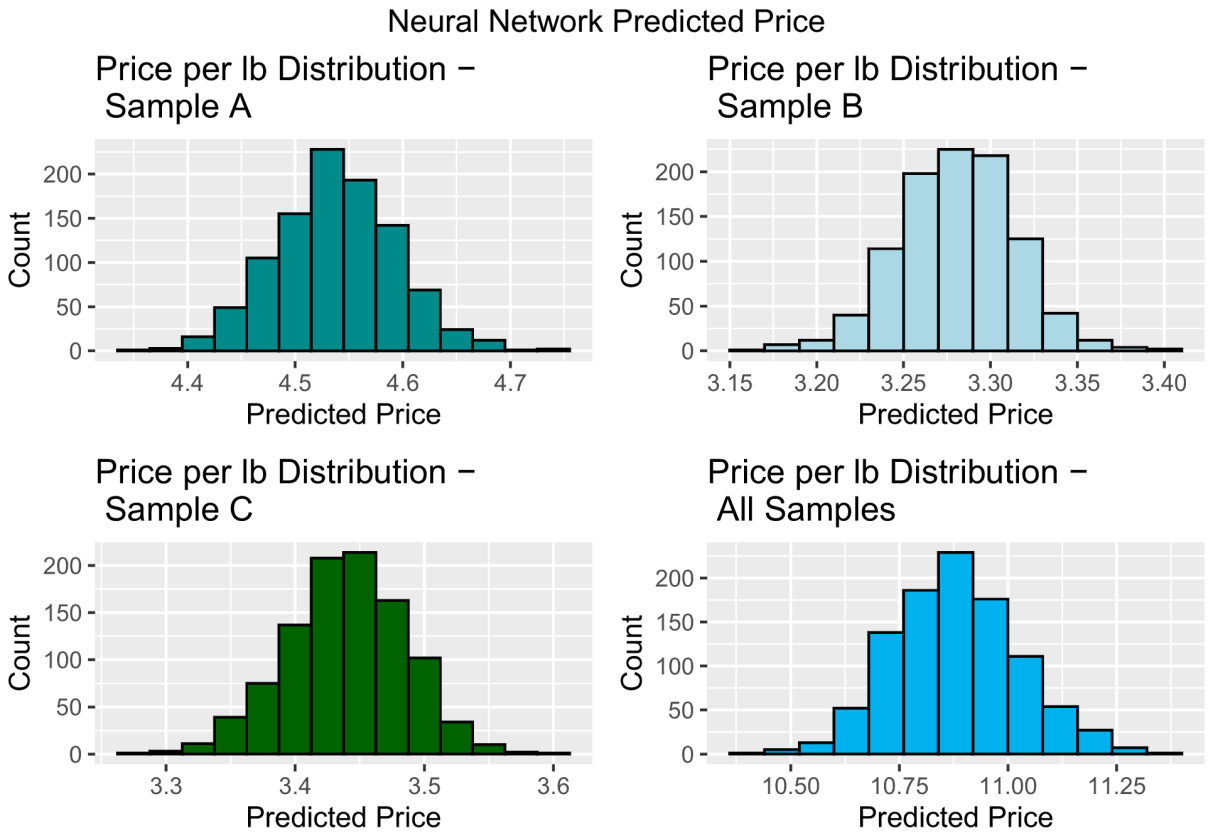

Figure 3: NNet - Predicted Price

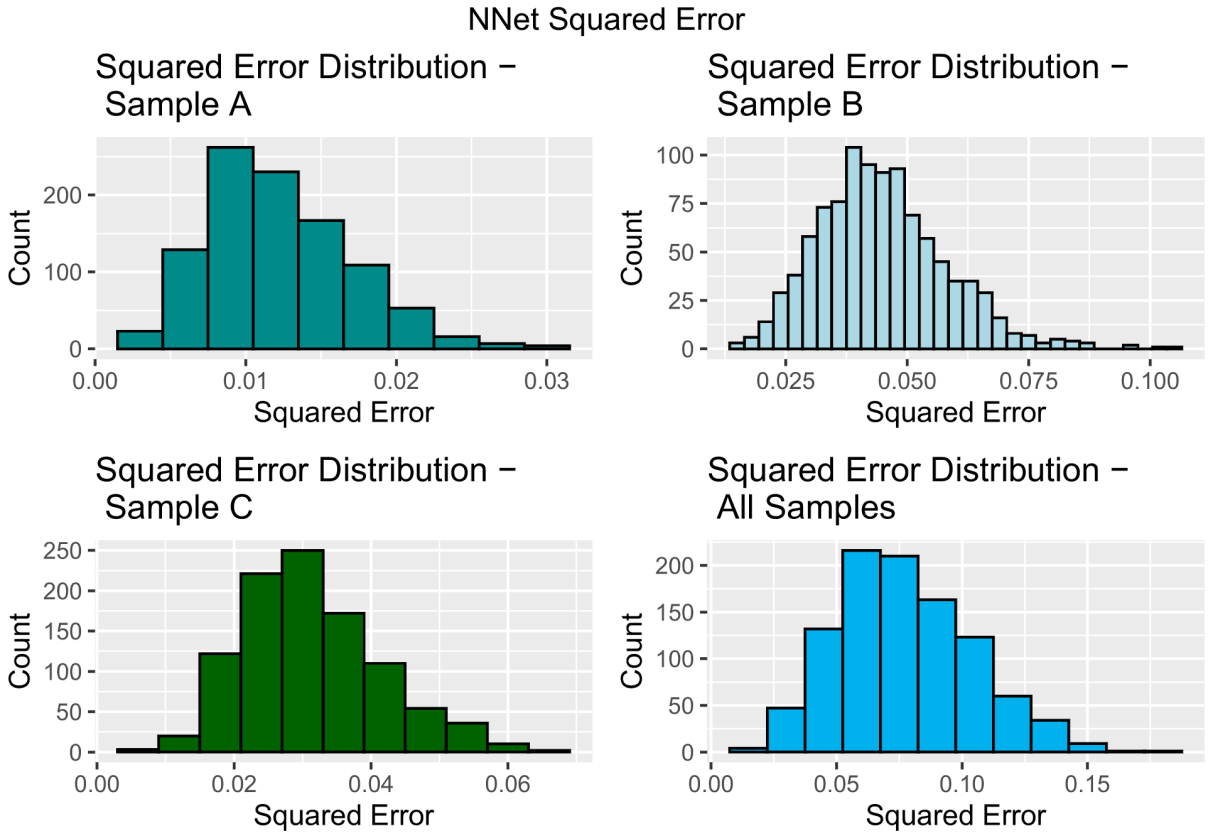

Figure 4: NNet Squared Error

Table 9: Result Metrics of Multinomial - All Variables

	Metrics
decay	0.0001
logLoss	0.2775
Accuracy	0.9003
Mean_F1	0.9005
Mean_Sensitivity	0.9003
Mean_Specificity	0.9801

Table 10: Accuracy by Class - All Variables

Class	Accuracy
BOMBAY	0.9980
CALI	0.9463
DERMASON	0.8734
HOROZ	0.9035
SEKER	0.9158
SIRA	0.7708

Table 11: Result Metrics of Multinomial - Noncorrelated Features

	Metrics
decay	0.0001
logLoss	0.2741
Accuracy	0.9027
Mean_F1	0.9027
Mean_Sensitivity	0.9027
Mean_Specificity	0.9805

Table 12: Accuracy by Class - Non-Correlated Features

Class	Accuracy
BOMBAY	1.0000
CALI	0.9463
DERMASON	0.8755
HOROZ	0.9035
SEKER	0.9198
SIRA	0.7765

Table 13: Sample Prediction Bean Distribution

Class	A	B	C	All
BOMBAY	0.0284	0.0000	0.0000	0.0070
CALI	0.4652	0.0000	0.1039	0.1479
DERMASON	0.0193	0.5790	0.1660	0.3108
HOROZ	0.0142	0.0087	0.5550	0.1814

Class	A	B	C	All
SEKER	0.4433	0.1704	0.0092	0.1875
SIRA	0.0296	0.2418	0.1660	0.1654

Table 14: Bootstrap Samples Results - Multinomial Logistic Regression

Metric	A	B	C	All
Actual Mean Price	4.64906	3.08579	3.27824	10.65218
Predicted Mean Price	4.52999	3.29789	3.45888	10.92324
Lower Conf	4.42382	3.23481	3.36798	10.64743
Median	4.53014	3.29772	3.45958	10.92038
Upper Conf	4.64276	3.36212	3.54722	11.23464

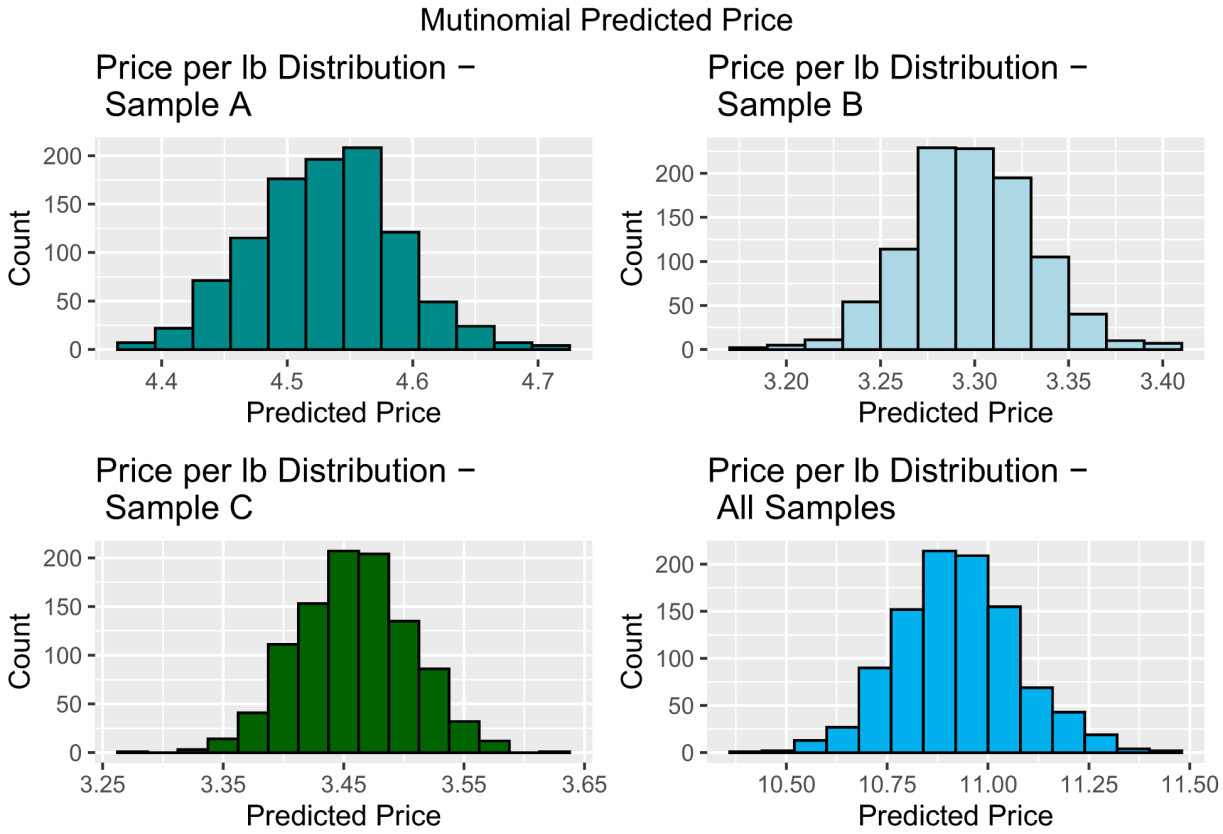

Figure 5: Multinomial Predicted Price

Table 15: Result Metrics of SVM - All Variables

	Metrics
cost	0.500
Accuracy	0.905
Mean_F1	0.905

	Metrics
Mean_Sensitivity	0.905
Mean_Specificity	0.981

Table 16: Accuracy by Class - All Variables

Class	Accuracy
BOMBAY	1.0000
CALI	0.9577
DERMASON	0.8768
HOROZ	0.9006
SEKER	0.9215
SIRA	0.7776

Table 17: Result Metrics of SVM - Random Forest Variables

	Metrics
cost	0.500
Accuracy	0.905
Mean_F1	0.906
Mean_Sensitivity	0.905
Mean_Specificity	0.981

Table 18: Accuracy by Class - Random Forest Variables

Class	Accuracy
BOMBAY	1.0000
CALI	0.9537
DERMASON	0.8645
HOROZ	0.9087
SEKER	0.9235
SIRA	0.7871

Table 19: Sample Prediction Bean Distribution

Class	A	B	C	All
BOMBAY	0.0284	0.0000	0.0000	0.0070
CALI	0.4613	0.0007	0.1039	0.1472
DERMASON	0.0193	0.5849	0.1670	0.3136
HOROZ	0.0142	0.0066	0.5519	0.1795
SEKER	0.4407	0.1726	0.0092	0.1878
SIRA	0.0361	0.2353	0.1680	0.1648

Table 20: Bootstrap Samples Results - SVM

Metric	A	B	C	All
Actual Mean Price	4.65042	3.06881	3.28372	10.62345
Predicted Mean Price	4.53187	3.29670	3.47563	10.92201
Lower Conf	4.42765	3.22686	3.39042	10.66061
Median	4.53220	3.29732	3.47763	10.92266
Upper Conf	4.64320	3.36409	3.57120	11.21054

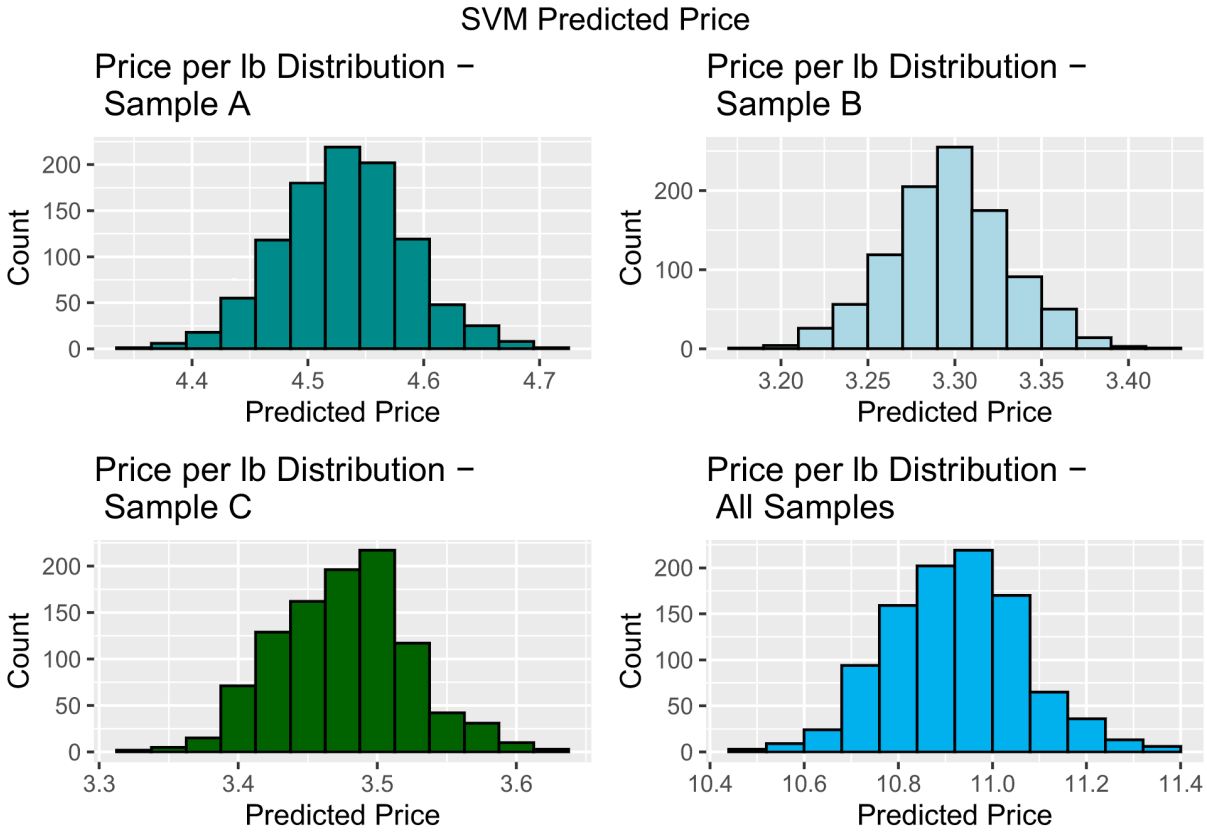

Figure 6: SVM Predicted Price

Table 21: Predicted Samples Squared Error - SVM

Metric	A	B	C	All
Mean	0.0146	0.0529	0.0377	0.0914
Lower Conf	0.0060	0.0281	0.0191	0.0419
Median	0.0140	0.0516	0.0367	0.0886
Upper Conf	0.0270	0.0843	0.0635	0.1527

Table 22: Predicted Samples Squared Error - SVM

Metric	A	B	C	All
Mean	0.0146	0.0529	0.0377	0.0914
Lower Conf	0.0060	0.0281	0.0191	0.0419
Median	0.0140	0.0516	0.0367	0.0886
Upper Conf	0.0270	0.0843	0.0635	0.1527

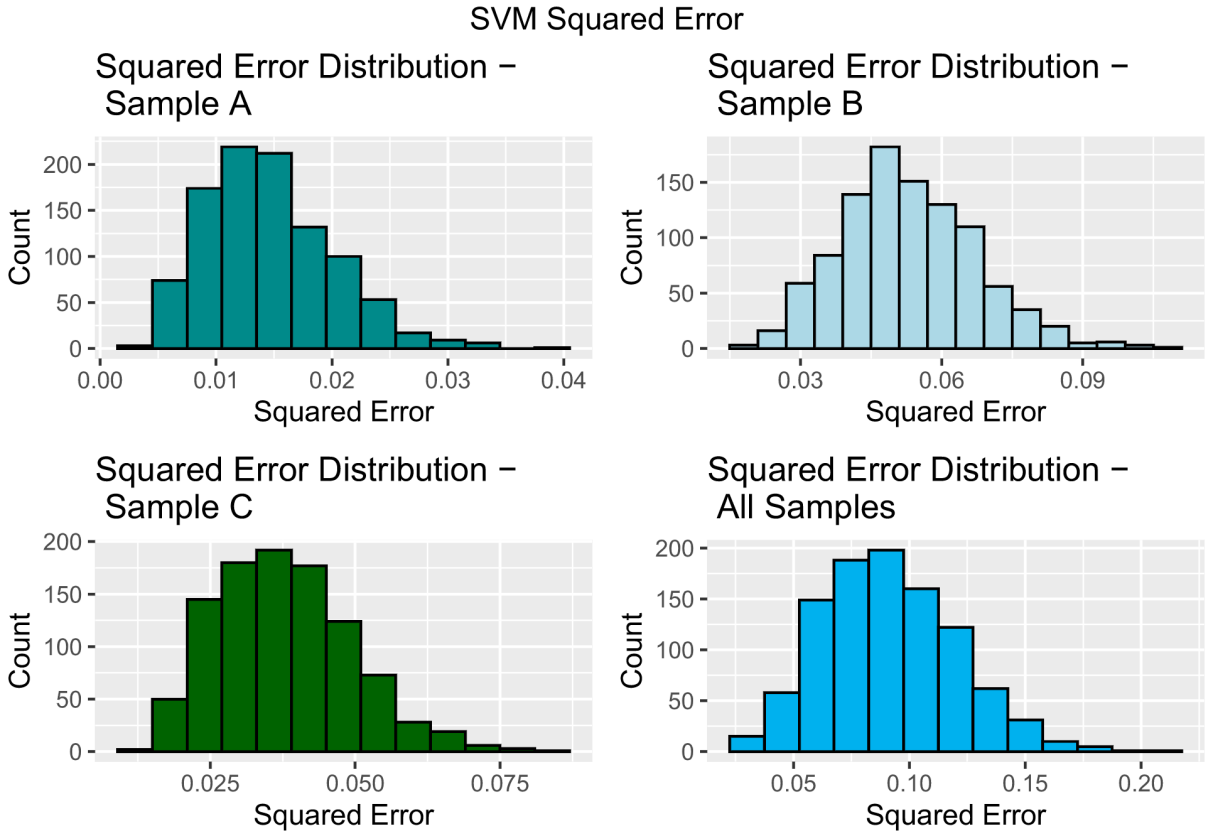

Figure 7: SVM Squared Error

Table 23: MSE Comparison Between Samples

Model	A	B	C	All
Neural Network	4.6463	3.0712	3.2649	10.6066
Multinomial	4.6491	3.0858	3.2782	10.6522
SVM	4.6504	3.0688	3.2837	10.6235

Table 24: Predicted Prices for 1lb Bean Samples

Sample	Price
Sample A	4.55
Sample B	3.25

Sample	Price
Sample C	3.37
All Samples	11.18

Table 25: Average accuracy across unequal distributions

nnet	mnom	svm
0.909	0.906	0.909

Appendix A:

Variables:

The training dataset contains the following variables:

- **Area(A):** The area of a bean zone and of pixels within its boundaries: $A = \sum_{r,c \epsilon R} 1$, where, r,c is size of bean region.
- **Perimeter(P):** Bean circumference is defined as the length of its border.
- **MajorAxisLength(L):** The distance between the ends of the longest line that can be drawn from a bean.
- **MinorAxisLength(l):** The longest line that can be drawn from the bean while standing perpendicular to the main axis.
- **Eccentricity (Ec):** Eccentricity of the ellipse having the same moments as the region.
- **ConvexArea (C):** Number of pixels in the smallest convex polygon that can contain the area of a bean seed.
- **Extent (Ex):** The ratio of the pixels in the bounding box to the bean area. $Ex = \frac{A}{A_h}$, where, $A_b$is the Area of a bounding rectangle.
- **Class:** One of the six bean types (BOMBAY, CALI, DERMASON, HOROZ, SEKER, SIRA)

Additional Calculated Variables

In addition to the variables given in the training dataset, we used an additional 9 features for classification prediction that were used in the Koklu el al. bean classification study to predict the classification of a bean.

- **Aspect Ratio(K)**: The relationship between the major axis length (L) and mine axis length (l) and is calculated by: $K = \frac{L}{l}$
- **Solidity(S)**: The ratio of the area of a bean to the area of the smallest convex polygon that fit tightest around the bean. $S = \frac{A}{C}$
- **Roundness(R)**: The measure of the sharpness of corners and edges for a given bean. $R = \frac{4\pi A}{P^2}$

- **Equivalent Diameter(Ed)**: Calculating an adjusted diameter in relation to the area of a bean. $Ed = \sqrt{\frac{4A}{\pi}}$
- **Compactness(CO)**: Measure the roundness of the bean using the equivalent diameter in relation to the major axis length. $CO = \frac{Ed}{L}$
- **Shape Factor Features**: Taking the relation of multiple measures to help further classify each bean:

$$ShapeFactor1(SF1) = \frac{L}{A}$$

$$ShapeFactor2(SF2) = \frac{l}{A}$$

$$ShapeFactor3(SF3) = \frac{A}{\frac{L}{2}\frac{L}{2}\pi}$$

$$ShapeFactor3(SF4) = \frac{A}{\frac{L}{2}\frac{l}{2}\pi}$$

Appendix B

Reference list:

- Murat Koklu, Ilker Ali Ozkan, 2020. *Multiclass classification of dry beans using computer vision and machine learning techniques*.
- SDSU, Spring 2021. *STAT 602 Dry Bean Final Project Instructions.*
- Lecture by Statquest,Jun 4,2018.*Logistic Regression Details Pt1:Coefficients*.https://www.youtube.com/watch?v=vN5cNN2-HWE
- Lecture by Statquest,Jul 31,2017.*Maximum Liklihood,cearly explained!!!*. https://www.youtube.com/watch?v=XepXtl9YKwc
- Lecture by Dr. Bharatendra Rai, Nov 12,2015.*Multinomial Logistic Regression with R: Categorical Response Variable at Three Levels*. https://www.youtube.com/watch?v=fDjKa7yWk1U
- UCLA Institute for Digital Research & Education, Statistical Counseling.*Multinomial Logistic Regression R data Analysis Examples*. https://stats.idre.ucla.edu/r/dae/multinomial-logistic-regression/
- Data Science Beginners by Mohit Sharma, Dec 20,2018. *Multinomial Logistic Regression Using R*. https://datasciencebeginners.com/2018/12/20/multinomial-logistic-regression-using-r/
- R for Statistical Learning. *Chapter 21, The caret Package*. https://daviddalpiaz.github.io/r4sl/the-caret-package.html
- R Documentation, randomForest(version 4.6-14). *Importance: Extract variable importance measure*. https://www.rdocumentation.org/packages/randomForest/versions/4.6-14/topics/importance
- R Documentation,caret(version 6.0-86). *preProcess: Pre-Processing of Predictors*. https://www.rdocumentation.org/packages/caret/versions/6.0-86/topics/preProcess
- R Documentation,caret(version 6.0-86). *trainControl: Control parameters for train*. https://www.rdocumentation.org/packages/caret/versions/6.0-86/topics/trainControl - RPubs, Ubiqum Code Academy. *TuneGrid and TuneLength in Caret*. https://rpubs.com/Mentors_Ubiqum/tunegrid_tunelength
- Data Analytics Workouts, May 23,2017. *Neural Networks in R (Part 2 of 4) - caret and nnet on State 'Personality' and Political Outcomes*. http://dataanalyticsworkouts.blogspot.com/2017/05/neural-networks-part-2-of-4-caret-and.html
- stackoverflow, *Looking up values from another dataframe in r*. https://stackoverflow.com/questions/51328147/looking-up-values-from-another-dataframe-in-r
- StackExchange, CrossValidated. *How is R calculating prediction intervals for the mean response at x0? Manual calculation yields different result*. https://stats.stackexchange.com/questions/183460/how-is-r-calculating-prediction-intervals-for-the-mean-response-at-x0-manual-ca

- StackExchange, CrossValidated. *Do we have to scale new unseen feature data for prediction*. https://stats.stackexchange.com/questions/205061/do-we-have-to-scale-new-unseen-feature-data-for-prediction
- Stackoverflow. *In R, how do I compute factors' percentage given on different variable?[duplicate]*. https://stackoverflow.com/questions/24901061/in-r-how-do-i-compute-factors-percentage-given-on-different-variable
- *Model Building with caret (Nominal Outcomes)*. https://remiller1450.github.io/s230f19/caret3.html
- Carlos Santillan, Aug 8,2017. *Parameter tuning caret*. https://csantill.github.io/RTuningModelParameters/
- rdrr.io. *models: A List of Available Models in train*. https://rdrr.io/cran/caret/man/models.html
- stackoverflow. *run a for loop in parallel in R*. https://stackoverflow.com/questions/38318139/run-a-for-loop-in-parallel-in-r
- Package 'foreach', Oct 15,2020. *Provides Foreach Looping Constrct*. https://cran.r-project.org/web/packages/foreach/foreach.pdf
- Matthew E. Clapham, Feb 15, 2016. *26: Resampling methods (vootstapping)*. https://www.youtube.com/watch?v=gcPIyeqymOU
- dplyr.*Recode values*.https://dplyr.tidyverse.org/reference/recode.html
- stackoverflow. *How do I replace NA values with zeros in an R dataframe?*. https://stackoverflow.com/questions/8161836/how-do-i-replace-na-values-with-zeros-in-an-r-dataframe
- dplyr. *Join two tbls together*. https://dplyr.tidyverse.org/reference/join.html
- The caret Package. *7 train Modles By tag*. https://topepo.github.io/caret/train-models-by-tag.html#support-vector-machines
- stackoverflow. *For each row return the column name of the largest value*. https://stackoverflow.com/questions/17735859/for-each-row-return-the-column-name-of-the-largest-value
- stackoverflow. *Add a variable to a data frame containing max value of each row*. https://stackoverflow.com/questions/3071271/add-a-variable-to-a-data-frame-containing-max-value-of-each-row
- Gabor Csardi, Jun 24, 2008. *[R] reset row numbers when extracting a subset of a table*. https://stat.ethz.ch/pipermail/r-help/2008-June/165787.html
- The caret Package. *5.7 Extracting Predictions and Class Probabilities*. https://topepo.github.io/caret/model-training-and-tuning.html#extracting-predictions-and-class-probabilities
- Tweet by Hilary Parker. *Does anyone know a hack to use a vector of sizes rather than a single size when using sample_n()? #rstats #dplyr*. https://twitter.com/kilroy2718/status/862101172030554113

Appendix C

Document Libraries

- **knitr** package used for *kable* function used to format tables
- **ggplot2** package for graphs
- **gridExtra** package for formatting multiple **ggplot2** graphs
- **GGally** package for pair plots in **ggplot2** formatting
- **dplyr** package for restructuring data frames
- **tidyverse** package for restructuring data frames
- **mlbench** package
- **gridExtra** package for formatting multiple **ggplot2** graphs
- **randomForest** package used for feature selection modeling
- **NeuralNetTools** package for Neural Network plotting
- **doParallel** package for parallel processing on all cores
- **caret** package for neural network model, cross validation, and bootstrapping
- **NeuralNetTools** package for plotting neural network structures