Data Bootcamp Final Project

Predictive Analysis on Chocolate Bar Ratings

Yulin Xia

I. Introduction

Chocolate is one of the most widely enjoyed snacks in the world, with countless brands competing to deliver the "Best Chocolate". However, the quality of a chocolate bar is influenced by a variety of factors—origin of cocoa beans, cocoa percentage, ingredient composition, and sensory characteristics such as flavor profiles. For my final project, I aim to analyze which characteristics most strongly influence rating outcomes, and to build a predictive model capable of estimating a chocolate bar's rating given its attributes. This project will provide chocolate makers with insight to cite selection of cocoa origins and refine product formulations. For retailers, the project predicts potential high-quality chocolate bars and allows them to stock high-quality products.

By combining exploratory data analysis with predictive models, this project not only predicts ratings but also uncovers patterns in how experts evaluate chocolate—insights that can inform product development, marketing, and sourcing decisions.(change to conclusion?)

II. Data Description

The dataset used in this project is sourced from Kaggle and it documents expert reviews of over 1,700 plain dark chocolate bars. The data is presented in CSV format and includes both categorical and numerical variables that capture the composition and characteristics of each chocolate sample. Each entry represents a single chocolate bar evaluated by an expert reviewer, with scores ranging from 1.0 to 5.0 in increments of 0.25, 0.50, or 0.75. For each rating of chocolate bar, corresponding features including the year of review, cocoa percentage, ingredient count, and presence or absence of components, etc. are also documented along the data. However, data cleaning and preprocessing are necessary in order to remove null values, minimize noise, and exclude unrelated details.

2.1 Data Preprocessing

- *Taste*

For the first_taste and second_taste, since the taste documented is too detailed (tastes almost vary between every chocolate bar), I decided to recreate the column by keeping the 5 most frequently existing tastes while assigning the rest tastes as "Other".

For the third_taste and forth_taste, since the majority values are missing, I decided to drop these two columns.

- *Ingredient*

Since salt/sugar/sweetener have consistent values in almost all samples, I decided to narrow my prediction to those without salt, with sugar, without sweetener chocolate bars, resulting in 1976 samples.

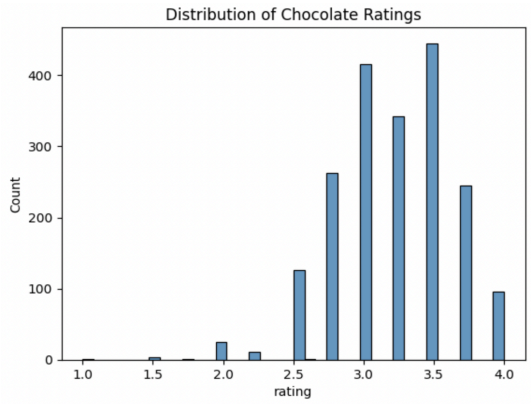
- *Company*

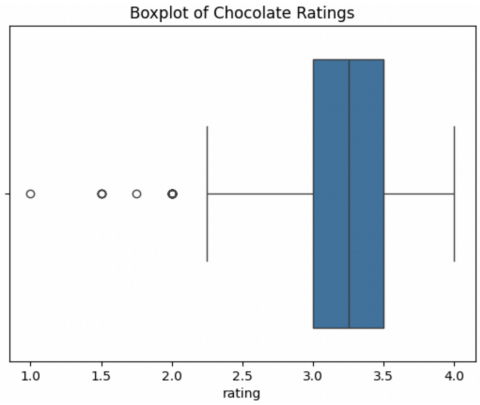
After observing the data, I realized that companies are also too dispersed, which means it is hard to generalize and predict since new companies keep existing. Intuitively, company and company location also should not be a factor of chocolate bar taste rating, so I chose to drop company related columns.

- *Bean Region*

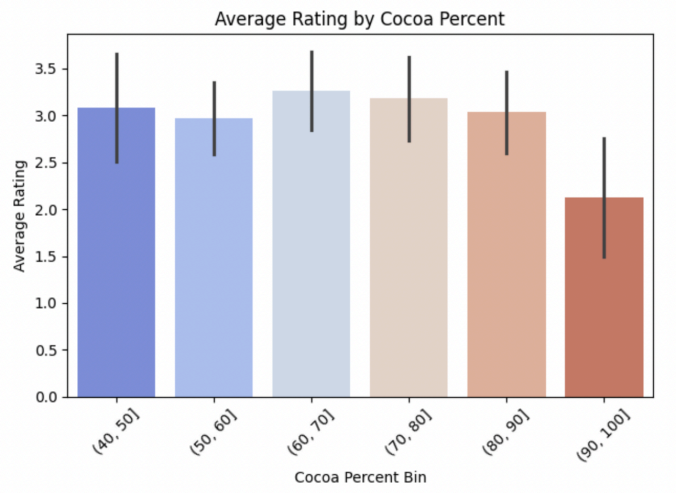
While bean region being too detailed, it is important to the taste and thus the rating of chocolate bars. Therefore, I choose to map detailed country information to larger region categories including `South America (SA)`, `Central America & Caribbean (CA&C)`, `Africa (AF)`, `Asia (AS)`, `Oceania (OC)`, `North America`, and `Mixed/Other`

2.2 Exploratory Data Analysis



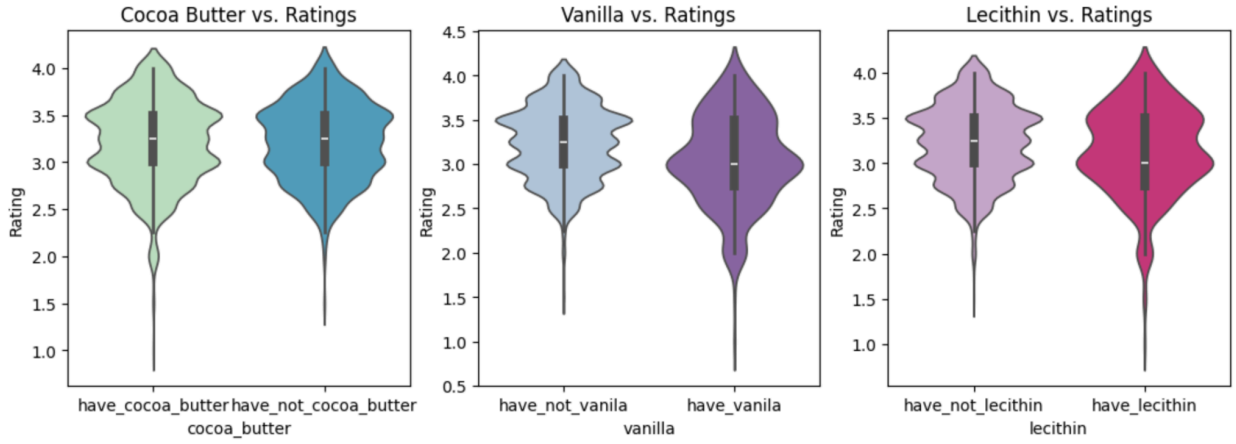


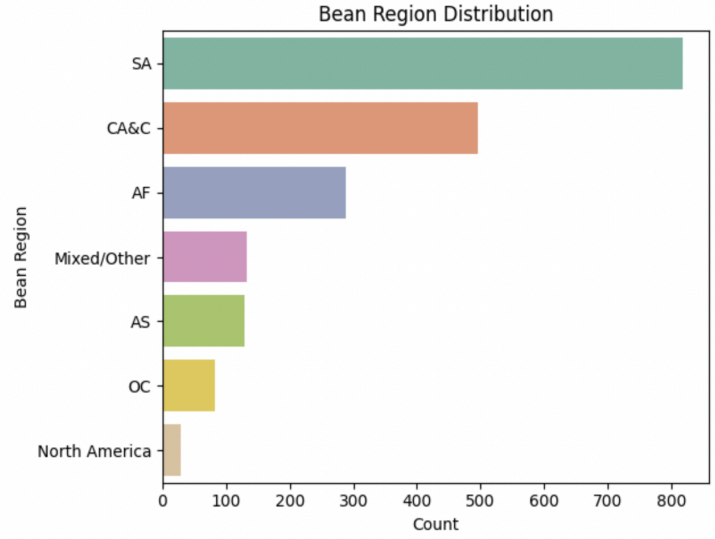
Based on the histogram and boxplot of chocolate ratings, we can observe distribution of chocolate ratings is left-skewed, with most bars rated between 3.0 and 3.5. A few outliers on the lower end (around 1.0 to 2.0) indicate occasional poor reviews, but overall, ratings tend to be favorable. The average rating is 3.21.



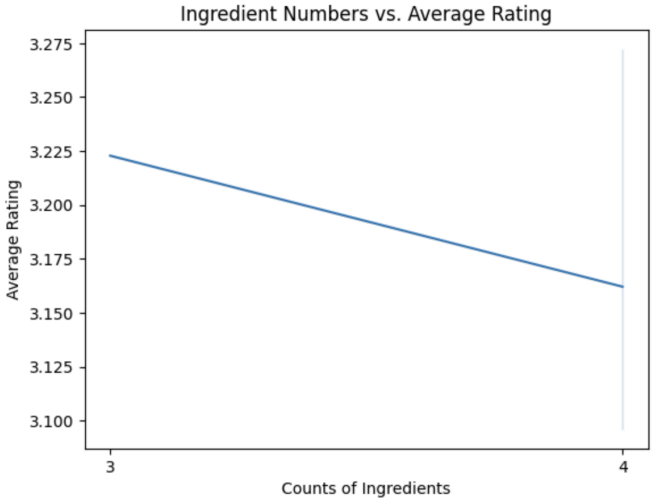
The Average ratings remain relatively stable across cocoa percentages below 70%, mostly clustering around 3.0–3.3. However, ratings started to drop gradually for chocolates with cocoa content above 70%, suggesting bitterness may reduce consumer appeal.

Chocolates without vanilla or lecithin tend to have slightly higher ratings, suggesting simpler ingredient profiles may be preferred. The presence or absence of cocoa butter appears to have no significant impact on ratings.

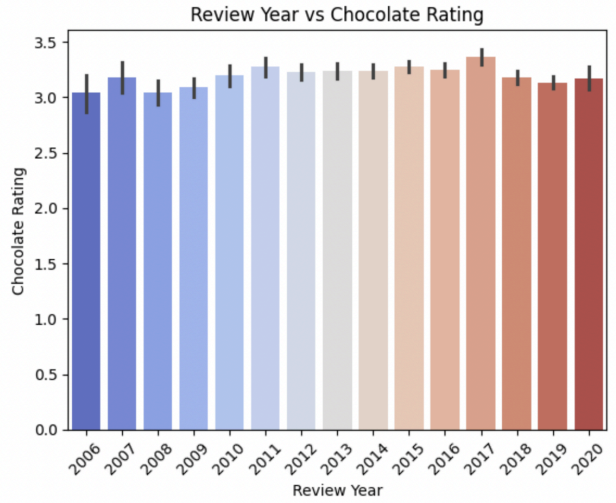




Based on the Bean Distribution, South America is the most common bean source by a wide margin, followed by Central America & the Caribbean and Africa. Regions like Oceania and North America are much less represented in the dataset.



The graph suggests a slight negative trend: chocolate bars with 4 ingredients tend to receive lower average ratings than bars with 3 ingredients. This may indicate that customers would prefer simpler ingredients, but the evidence is not strong. It is worth mentioning that the number of ingredients is highly correlated with the column indicating the presence or absence of specific ingredients, so when building the predictive model, we should avoid double counting this column while keeping specific ingredients. chocolate bars.



Which year is the review made present a stable pattern on ratings across the years, which indicates review year might not be related to chocolate ratings.

III. Models & Methods

```
X = df[['cocoa_percent','cocoa_butter','vanilla','lecithin','first_taste','second_taste','bean_region']]
y = df['rating']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_state = 24)
```

Considering the potential correlation between features and ratings, I chose `'first_taste'`, `'second_taste'`,`'bean_region'`,`'cocoa_butter'`,`'vanilla'`,`'lecithin'` as categorical features, `'cocoa_percent'` as numerical features to predict the ratings.

For each of the following models, I decided to utilize an 70-30 train-test split, training my model on 70% of the data and then testing it on the remaining 30%.

The performance of the models are evaluated by minimizing the mean standard error on test data.

3.0 Baseline Model

```
y = df['rating']
baseline_preds = np.ones(len(y))*y.mean()
mean_squared_error(y, baseline_preds)
```

```
0.18859511250389288
```

For the baseline model, I simply use the mean of the rating column as a prediction for chocolate ratings. Then, I computed the mean square error of 0.189 as my baseline. It indicates that the further models should at least have a mean square error of 0.189 to outperform the baseline model. Considering the tight range of rating between 1 to 4, 0.189 is a reasonable baseline but improvement is possible to be made.

3.1 Multiple Linear Regression

I chose multiple linear regression because it enables simultaneous consideration of multiple independent variables and how they each contribute to the dependent variable (ratings). It is able to explore linear relationships we have seen in exploratory data analysis.

1. I first used OneHotEncoder to transform categorical columns.

```
transformer = make_column_transformer((OneHotEncoder(drop = 'first', sparse_output = False), cat_col), remainder = 'passthrough')
```

2. I created a linear regression pipeline: lr_pipe, and fit my train data on the lr_pipe.

```
# create pipeline for multiple regression model
lr_pipe = Pipeline([('encode', transformer), ('model', LinearRegression())])
```

```
# fit pipeline
lr_pipe.fit(X_train, y_train)
```

3. I used “lr.intercept” and “lr.coef_” to find the intercept and coefficients of the regression line. The intercept of 3.84 means the predicted chocolate rating when all encoded features are at their baseline level is 3.84.
4. I calculated the mean squared error for both train and test data based on the model prediction. Both of them show a smaller mse than baseline model.

```
#calculate mse for training data
y_train_preds = lr_pipe.predict(X_train)
mean_squared_error(y_train, y_train_preds)
```

```
0.17297778352124954
```

```
#calculate mse for testing data
y_test_preds = lr_pipe.predict(X_test)
mean_squared_error(y_test, y_test_preds)
```

```
0.17061069883708987
```

5. Additionally, I also computed the feature importance. We can see that the most important features are the first taste of the chocolate, and the presence of vanilla in the ingredients. It is surprising that cocoa percent did not play a significant role in chocolate rating.

cocoa_percent	0.012487
cocoa_butter	0.006736
vanilla	0.061957
lecithin	0.010463
first_taste	0.071357
second_taste	0.001739
bean_region	-0.002879

3.2 K-Nearest Neighbors Regression Model

I used KNN regression because it is a simple yet effective non-parametric method. It makes predictions based on the local structure of the data. Unlike linear models, KNN does not assume a functional form between features and the target variable. Since the MSE improvement is fairly small in the regression, changing the linear relationship assumption may be helpful. Since distance calculations are sensitive to the scale of features, I applied a standard scaler in the pipeline to normalize all input variables to ensure fair comparisons across dimensions.

1. I first used OneHotEncoder to transform categorical columns. Also, use StandardScaler() to scale the numerical columns.

```
transformer = make_column_transformer((OneHotEncoder(drop = 'first', sparse_output = False), cat_col), remainder = StandardScaler())
```

2. Create pipeline for knn model: knn_pipe

```
#create pipeline for knn regression model
knn_pipe = Pipeline([('encode', transformer), ('model', KNeighborsRegressor())])
```

3. To optimize the model's performance, I used GridSearchCV to search for the best number of neighbors, which directly controls the bias-variance tradeoff in knn. A small n can lead to overfitting by capturing noise, while a large n may ignore important local patterns.

   After trying n from 1 to 30 with 2 as step, we got the best numbers of neighbors as 21. The plug the best estimator into knn model

```
# Define grid of hyperparameters for number of neighbors
param_grid = {'model__n_neighbors': np.arange(1, 30, 2)}
```

```
# Perform grid-search with cross validation
grid_search = GridSearchCV(knn_pipe, param_grid, cv=5, scoring='neg_mean_squared_error')
grid_search.fit(X_train, y_train)
```

```
# Determine best parameter
grid_search.best_params_
```

```
{'model__n_neighbors': np.int64(21)}
```

```
# Use 21 neighbors in model
knn = grid_search.best_estimator_
```

4. Calculate the mse for both train and test data based on the model prediction.

```
# Calculate mse for training data
y_train_preds = knn.predict(X_train)
mean_squared_error(y_train, y_train_preds)
```

```
0.16833181259970845
```

```
# Calculate mse for testing data
y_test_preds = knn.predict(X_test)
mean_squared_error(y_test, y_test_preds)
```

```
0.17443178732988418
```

5. Compute importance. It shows that in the knn model, the importance of cocoa_percent increased to be the most important feature. “First taste” and “vanilla” are still at the top 3 importance.

cocoa_percent	0.048466
cocoa_butter	0.025435
vanilla	0.028776
lecithin	0.001036
first_taste	0.038279
second_taste	-0.001013
bean_region	-0.006107

3.3 Decision Tree Model

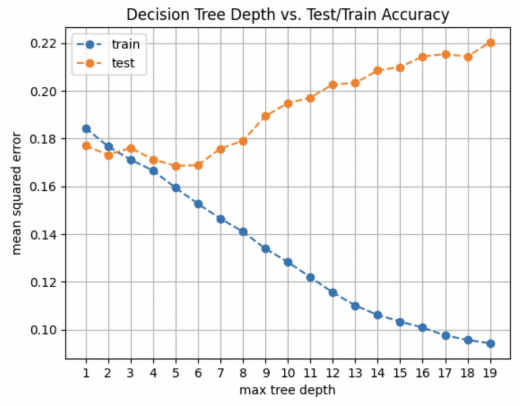
I chose to use a decision tree regression model because it can effectively capture non-linear patterns in the chocolate rating data. Since the knn results are not very strong, I decided to look at models that also capture non-linear relationships like the knn model, but include more. Decision tree is a good candidate because it not only analyzes non-linear features, it also shows specific decision making processes. Additionally, decision trees offer strong interpretability, allowing me to visualize how each feature contributes to the prediction and better understand the relationships within the dataset.

1. OneHotEncode categorical columns and encode x train and x test

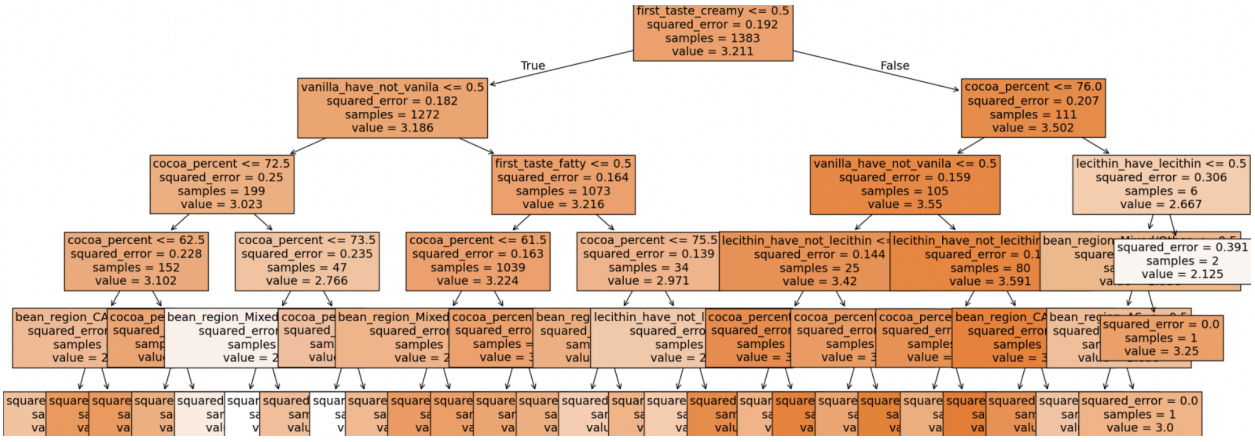
```
encoder = make_column_transformer((OneHotEncoder(sparse_output = False),cat_col), verbose_feature_names_out = False, remainder = 'passthrough')
```

```
X_train_encoded = encoder.fit_transform(X_train)
X_test_encoded = encoder.transform(X_test)
```

2. Find optimal max depth by plotting test accuracy and use the minimum mse caressponded depth. Make sure not to overfit.



3. Find the optimal max depth is 5, and then plot the decision tree. The root splits first on **first_taste_creamy**, indicating it is the most influential feature. Chocolate bars with a creamy first taste tend to receive lower ratings (mean ~3.21), while non-creamy ones skew higher (~3.50). The tree then frequently splits on **vanilla_have_not_vanila** and **cocoa_percent**, showing these are also key predictors. For example, bars without vanilla and with higher cocoa percentages (e.g., >76%) tend to get the highest predicted ratings (~3.59). Overall, **first taste, vanilla presence, and cocoa percentage** are the most decisive factors driving predicted chocolate ratings, which is also consistent with previous importance computation.



4. Calculate the mse for both train and test data based on the model prediction.

```
#calculate mse for training data
y_train_preds = dtree.predict(X_train_encoded)
mean_squared_error(y_train, y_train_preds)
```

```
0.15935208726758573
```

```
#calculate mse for testing data
y_test_preds = dtree.predict(X_test_encoded)
mean_squared_error(y_test, y_test_preds)
```

```
0.16848185585317949
```

3.4 Random Forest Model

I chose to use a Random Forest regression model because it mitigates overfitting by averaging the results of multiple decision trees. Unlike a single tree, Random Forests introduce randomness which enhances generalization and avoids being over sensitive. This model is well-suited for handling both categorical and numerical variables, and it automatically captures complex nonlinear interactions.

1. OneHotEncode categorical columns

```
transformer = make_column_transformer((OneHotEncoder(drop = 'first', sparse_output = False), cat_col), remainder = 'passthrough')
```

2. Create a random forest pipeline: forest_pipe and try different n estimators with different max depth, decide the best params. Apply the best parameter with 150 trees and 6 as max depth to the model.

```
# Create pipeline for multiple regression model
forest_pipe = Pipeline([('encode', transformer), ('model', RandomForestRegressor())])
```

```
# Define grid of hyperparameters for number of estimators and max depth
param_grid = {'model__n_estimators': [50, 100, 150, 200, 250],'model__max_depth': [2,3,4,5,6,8,10]}
```

```
# Perform grid-search w/ cross validation
grid_search = GridSearchCV(forest_pipe, param_grid, cv=5, scoring='neg_mean_squared_error')
grid_search.fit(X_train, y_train)
```

```
# Determine best parameters
grid_search.best_params_
```

```
{'model__max_depth': 6, 'model__n_estimators': 150}
```

```
# Apply the best estimator
forest = grid_search.best_estimator_
```

3. Calculate mse for both train and test data based on the model prediction.

```
] # Calculate mse for training data
  y_train_preds = forest.predict(X_train)
  mean_squared_error(y_train, y_train_preds)
```

```
0.14856856114804856
```

```
] # Calculate mse for testing data
  y_test_preds = forest.predict(X_test)
  mean_squared_error(y_test, y_test_preds)
```

```
0.1674196864108165
```

4. Compute importance. It shows consistent results that “cocoa_percent”, “vanilla” and “first_taste” are the most important features. Noticeably, their importance all increased compared to previous models.

cocoa_percent	0.096145
cocoa_butter	0.006526
vanilla	0.063749
lecithin	0.013845
first_taste	0.062283
second_taste	0.002307
bean_region	-0.007884

IV. Results and Interpretation

In this project, I tested four predictive models: Multiple Linear Regression, K-Nearest Neighbors (KNN), Decision Tree, and Random Forest. I compared each model’s performance using mean squared error (MSE) on both training and testing datasets, benchmarking them against a simple baseline predictor that always outputs the mean rating. The table below summarizes our findings:

	Train MSE	**Test MSE**
Baseline	0.189	
Multiple Linear Regression	0.173	0.171
KNN	0.168	0.174
Decision Tree	0.159	0.168
Random Forest	0.149	0.167

Multiple Linear Regression

The multiple linear regression model was my first attempt at capturing relationships in the data. It reduced the test MSE to 0.171, suggesting that there is at least a linear relationship between the features and the chocolate ratings. The training MSE (0.173) was nearly identical to the test MSE, which indicates overfitting is not present in this model. However, its performance gain over the baseline was very small. The limitation on linear regression performance makes sense since there are many non-linear features in the dataset including flavor, ingredients, and bean origin etc. The non-linear features’ complex interaction cannot be fully captured by the multiple

linear model. This model helped establish a foundation for comparison, but its limited flexibility constrained its predictive power.

K-Nearest Neighbors (KNN)

Next, I applied a KNN regression model. Since KNN is a distance-based algorithm, I scaled all numerical features using a standard scaler to ensure fair weighting during distance calculations. In addition, I used grid search to identify the optimal value of the number of neighbors n, settling on n=21, which yielded a test MSE of 0.174 and a training MSE of 0.168.

While KNN slightly outperformed linear regression on training data, its test MSE was higher, suggesting slight overfitting. Or it might indicate the model's sensitivity to local noise. Still, KNN's ability to capture nonlinear structure is useful when non-linear features interact in a subtle way that does not show significant pattern. However, the worse test MSE performance indicates that such a large n may overgeneralize local patterns and ignore smaller, localized rating patterns in favor of broader trends. In short, it provides an alternative understanding of non linear features, but not optimally.

Decision Tree Regression

To capture non-linearities and interactions more explicitly, I tried the Decision Tree regression model. With a carefully tuned max depth, the model produced a test MSE of 0.168 and a training MSE of 0.159. The lowest test MSE compared to KNN and linear regression shows its advantage in interpretability and flexibility. Importantly, the train-test MSE gap remained relatively small, suggesting that the model did not severely overfit the data despite its higher complexity.

In addition, the decision tree helped us understand which features most directly influenced chocolate ratings. In consistent with previous importance computation, first taste, presence of vanilla and cocoa percent are key for chocolate ratings.

Random Forest Regression

Given the improvement of the Decision Tree model compared to the first two models, I decided to further apply Random Forest Model on my data. My Random Forest combined the predictions of 150 trees with a max depth of 6, as determined by grid search. This model yielded the best performance, with a training MSE of 0.149 and a test MSE of 0.167.

The lower test MSE of 0.167 suggests that the Random Forest was more effective at generalizing unseen data while maintaining a strong fit on training data. The model reduces the decision tree model's overfitting tendencies by averaging multiple models trained on random subsets of the data and features. The conclusion from multiple trees capture subtle interactions without being overly sensitive to noise. The slight gain in test performance over all other models—while

numerically small—carries practical significance, especially given the complex, subjective nature of taste ratings.

Summary of Findings

In my predictive analysis of chocolate ratings, all models outperformed the baseline, with larger improvements every time. The model ranked in terms of performance are as follows:

Random Forest > Decision Tree > Multiple Linear Regression > KNearest Neighbors

Key findings:

1) Effectiveness of Random Forest Model:
   The Random Forest model delivered the best predictive performance, achieving the lowest test MSE of 0.167 among all models. Its strength lies in its ability to handle complex, non-linear relationships and interactions between categorical and numerical features, especially suitable for chocolate rating where objectivity and complex factors may interact. By averaging multiple decision trees, the model mitigates overfitting and captures subtle patterns that single models or linear regressions might overlook, making it particularly well-suited for this task.
2) Consistent Importance of Key Features:
   Across all models, certain features consistently emerged as significant predictors of chocolate ratings. Specifically, first_taste, vanilla, and cocoa_percent were frequently used in splits or assigned high coefficients. The best performing model Random Forest, in particular, assigned the greatest importance to these features, reinforcing their predictive relevance. Their consistency across different algorithms suggests these attributes play a central role in how chocolate quality is evaluated by reviewers.

V. Conclusion and Next Steps