Predicting food specialization from a food vendor's name

A comprehensive classifier analysis on the association of food vendor names with their descriptions

Zaki Aslam, Hector Palafox Prieto, Jennifer Tsang, & Samrawit Mezgebo Tsegay

2025-12-07

Table of contents

Summary

In this project, we used decision trees, logistic regression, and a Naive Bayes classifier to identify whether or not a food vendor sells hot dogs via their name. We trained each model individually using a cross-validation setup, and we compared the scores of the accuracy in order to determine a model to train and to compare to the test data. The model we chose, finally, was the Naive Bayes, as it provided a slightly better score and less underfit and overfit than the other models present. Finally, we validated it with our test data and came to the conclusion that even though it is good enough for classifying most of the cases, it still struggles to discern from the minority class, which, in our case, is our target.

Introduction

Food trucks and mobile food vendors are a common sight in Downtown Vancouver, offering a wide range of cuisine types from hot dogs and burgers to shawarma and tacos. With so many different vendors and food options, it can be useful to automatically identify what kind of food a vendor specializes in based only on select information. In this project, we study whether we can predict if a food vendor is a hot dog vendor or not using the vendor's business name. We used a publicly available dataset of mobile food vendors in Vancouver from the City of Vancouver Open Data Portal (Vancouver 2025a), where each row represents relevant information for a single food vendor and includes columns such as business name, location, description, local area, and geographic coordinates. For our analysis, we constructed a binary target variable labelled as True when the description is "Hot Dogs" and False otherwise. This allows us to investigate how much information about the type of food a vendor sells can be extracted from the business name, as well as putting to the test the predictive power of some of the most common classification algorithms: Decision Trees, Logistic Regression, and Naive-Bayes.

Methods

Data

The data set used in this project is Street food vending created by the City of Vancouver(Vancouver 2025b). It was sourced from the City of Vancouver Open Data Portal (Vancouver 2025a) and can be found here. Each row in the dataset represents a food vendor and includes information such as the business name, description, and location. In this project, we derive the binary target from the description column ("Hot Dogs" vs other descriptions) and use the business name as the main predictor in our models.

Before splitting the data into training and test sets and fitting models, we perform basic data validation on the raw tabular data to check that it is well-formed and consistent with our expectations. We expect the food vendors' data to come from a CSV file that can be read into a non-empty pandas DataFrame. If the download fails or the file is not in the expected tabular format, we want the analysis to stop early instead of producing confusing errors later.

Analysis

The dataset was partitioned into 70% for training set and 30% for testing set. Following an exploratory data analysis (EDA), a Countvectorizer was implemented to generate a Bag-of-Words (BoW) representation. This method will split each individual word in the names of the businesses into its own individual columns, and will assess whether or not the word is present in the data set. Four distinct models were compared using 5-fold cross-validation: 1) Dummy Classifier: Used as a baseline to establish the minimum acceptable performance for predicting the 'hot dog' category, 2) Decision Tree Classifier: Selected to determine if simple, hierarchical decision rules could effectively map out the relationship of the names to the category, 3) Logistic Regression: Included to identify linear relationships and determine which specific tokens (words) were most relevant for classification, and 4) Bernoulli (Naive-Bayes): Chosen for its efficiency in training and its probabilistic approach to classification. The models were ranked based on accuracy ($\frac{\text{correct predictions}}{\text{total predictions}}$). The top-performing model chosen was Naive-Bayes, and hyperparameter optimization was performed. The hyperparameters optimized were the alpha hyperparameter (which controls the tradeoff between variance and bias of our model), as well as the max features variable (the actual size of our vocabulary, considering the top max features words) of our Countvectorizer, as this can also play a role in overfitting. A randomized approach was used to test in a wide space, with 500 iterations and a random integer ranging from 5 to the size of the vocabulary for max features, and a loguniform distribution ranging from 0.001 to 1000 for alpha. Lastly, the model was retrained using the best hyperparameter selections and evaluated with the test set to formulate the final conclusion.

Results

EDA

When visualizing our EDA, we can notice several key points. From Figure 1, we can see that of all the cuisine types from Downtown Vancouver food vendors, hot dog stands seem to be the most common of them all. It is also very important to analyze our classes before starting our work. When you have a large class imbalance, a lot of the time, your model will give you a score that is not representative of whether or not your model works well. For example, if you observe the second plot for a data set where one class is represented in a much higher

proportion than the other, a model like Dummy Classifier will give you an extremely high score. This isn't because the model works perfectly, it's because it'll always predict the higher represented class! Yet we can see from Figure 2 that we do not have that issue as much here, as the class imbalance isn't too severe. Lastly, we noticed that there were some initial blanks in the business name, which we addressed by changing it to an empty space (and thus not screwing up the Countvectorizer instance we will need for this analysis). We can observe from Figure 3 that all the cases of a blank business name belong to Hot Dog vendors, which would be something we would like our models to capture. Finally, we would expect our classifier to be able to identify the "easy" base case of having no name, since this is a relevant discriminator for both our classes.

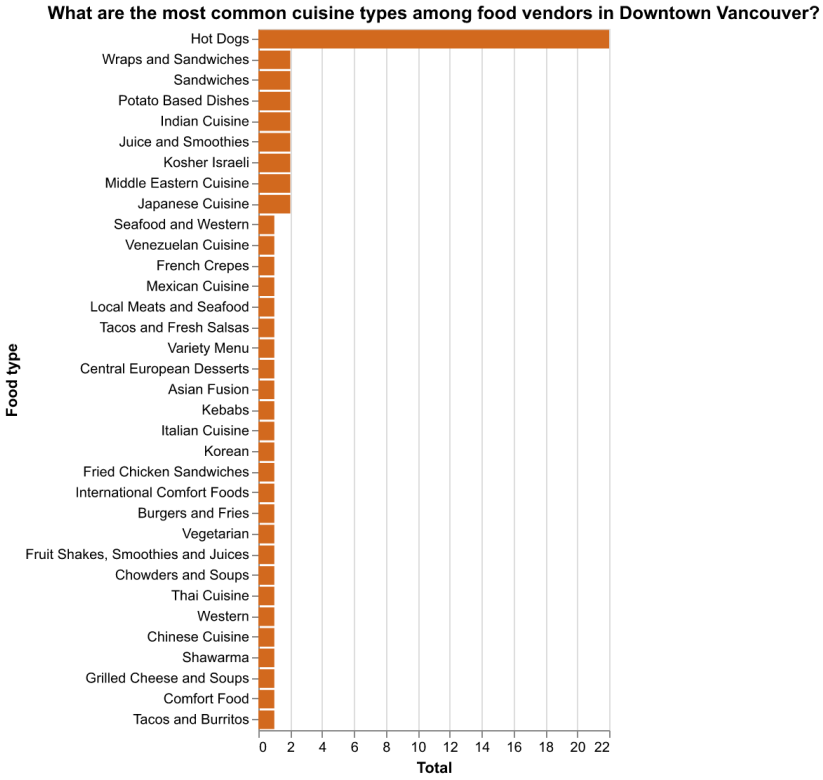

Figure 1: Most common cuisine types among food vendors in Downtown Vancouver.

Baseline

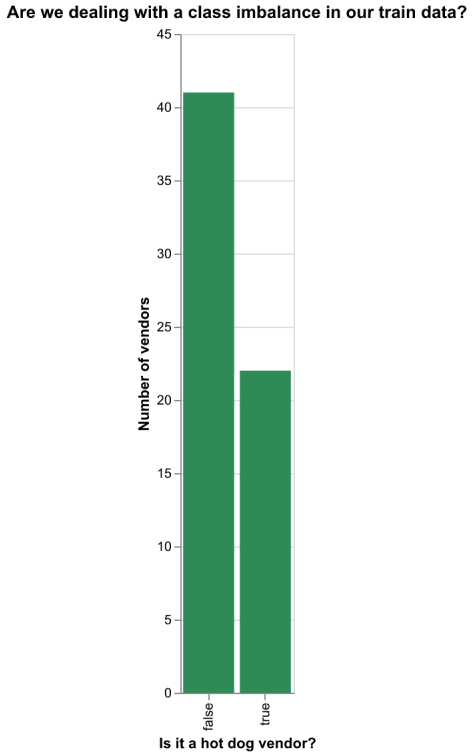

Figure 2: Class imbalance in training data.

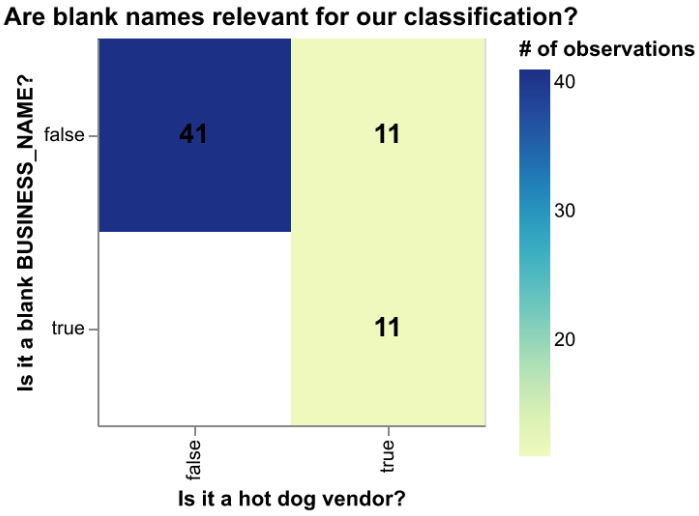

Figure 3: Blank names relevance in classification.

Table 1: Dummy Classifier cross validation scores and times.

fit_time	score_time	test_score	train_score
0.0013	0.0004	0.6154	0.66
0.0005	0.0003	0.6154	0.66
0.0004	0.0003	0.6923	0.64
0.0004	0.0003	0.6667	0.6471
0.0004	0.0003	0.6667	0.6471

As expected, we can see from Table 1 that the dummy consistently predicts the majority class, with an accuracy of around np.float64(0.6513), being consistent with the representation of our split.

Decision Tree

Here we are training a simple decision tree to identify whether the vendor sells hot dogs or not. This is a simple model with easy to interpret coefficients, and it would be interesting to check whether or not it correctly identified some of the most relevant clues (something like "Joe's Hot Dogs" being correctly classified, for instance).

Table 2: Decision Tree cross validation scores and times.

fit_time	score_time	test_score	train_score
0.0019	0.0004	0.6154	0.98
0.0008	0.0003	0.6923	0.98
0.0007	0.0003	0.6923	0.98
0.0007	0.0003	0.4167	0.9804
0.0007	0.0003	0.4167	1

From Table 2, we can see that the decision tree performed worse than the dummy classifier, as it is overfitting the prediction, which is evident in the substantial gap between the validation and training scores. By taking a look at the depths and tree structure, we can see the most discriminating factors and better understand these discrepancies. From Figure 4, we can observe some sensible initial discriminations, such as "dogs", "japadog" and "dog", which would quickly identify the vendor as a Hot Dog place. As we can see, the model contains 34 levels of decisions, yet, the level of specificity (given that we are using a bag of words) makes it perform poorly.

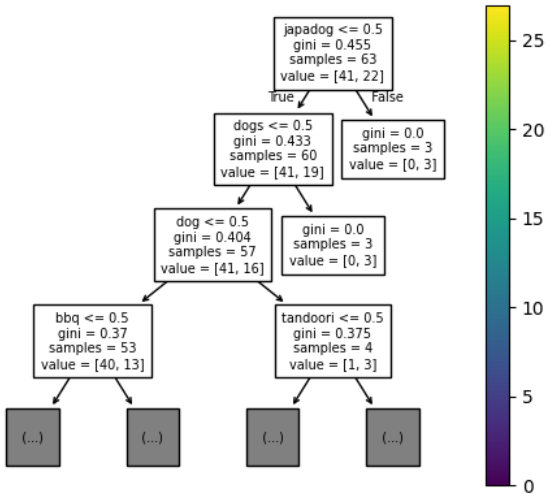

Figure 4: Decision tree structure (limited to the first 5 levels of the depth).

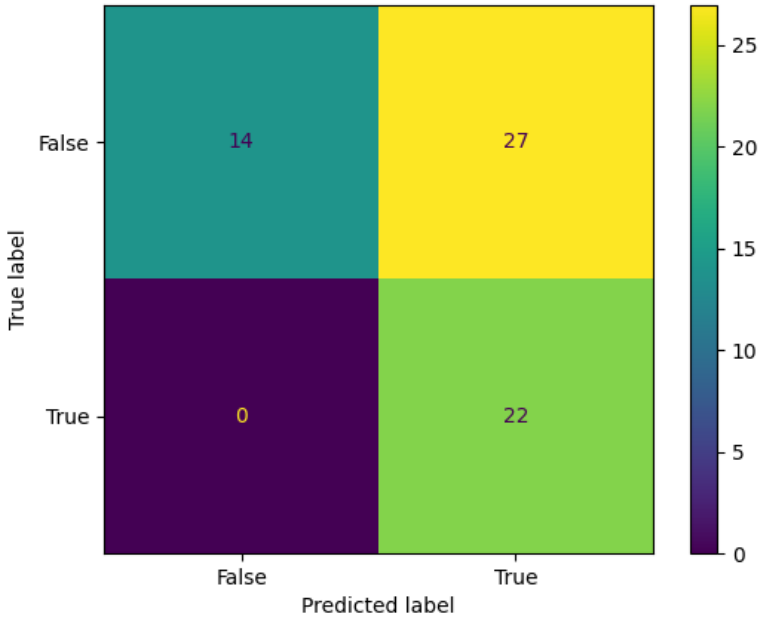

Figure 5: Confusion matrix for the Decision Tree.

Table 3: Mismatches for Decision Tree.

fit_time	score_time	test_score	train_score
0.0019	0.0004	0.6154	0.98
0.0008	0.0003	0.6923	0.98
0.0007	0.0003	0.6923	0.98
0.0007	0.0003	0.4167	0.9804
0.0007	0.0003	0.4167	1

From Figure 5, we can observe the confusion matrix and the misses in the cross validation of the model trained. We can observe that the model is very good at identifying when something seems "Hot-Doggy", yet it produces a high degree of false positives. We can observe some of the mistakes shown below in Table 3, and see that the model tends to predict most of the time that the vendor is a hot dog stand, with probably the only reasonable exception being Van Dog.

Logistic Regression

Here we will train a logistic regression in order to see whether or not we can improve our accuracy and reduce the possible overfitting. This model also has the advantage of having interpretable parameters, which in our case relate to how often each of our features is associated with the target variable.

Table 4: Logistic Regression cross validation scores and times.

fit_time	score_time	test_score	train_score
0.0471	0	0.6923	0.8
0.0159	0	0.7692	0.82
0.006	0	0.7692	0.82
0.0059	0.0014	0.9167	0.7843
0.0055	0.002	0.75	0.8235

From Table 4, we can see a slight improvement in generalization from the decision tree, but it performs not much better than the Dummy Classifer with an average validation score of np.float64(0.779). This could indicate that there may not be a single independent linear relationship between the token features to the target. Also, it is worth noting that the model has a lot of variability between folds.

Table 5: Coefficients of the logistic regression.

token	coefficients
actual	-0.287395
ali	0.258929
arancino	-0.209049
arepa	-0.287395
arturo	-0.195682
truck	-0.463946
van	0.208222
vij	-0.251701
wraps	-0.330149
yokabai	-0.337719

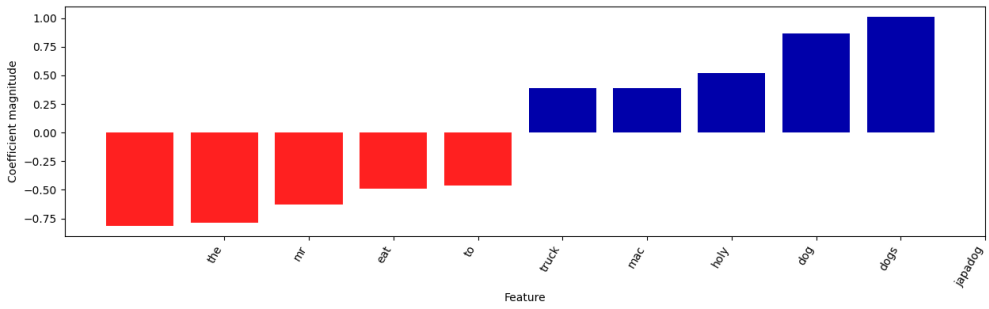

Figure 6: Top 5 most discriminant features (upper and lower).

From Table 5 and Figure 6, we can see that the coefficients associated roughly match with the choices determined by the decision tree, with “japadog”, “dog”, and “dogs” being relevant. Yet if we observe the intercept, we see the model is heavily biased toward making a negative prediction. Thus, we would not expect the model to be good at identifying hot dog places in particular. The number of coefficients produced is 95 and the intercept is np.float64(-0.335).

Table 6: Mismatches for Logistic Regression.

y	y_hat	probabilities	x
False	True	[0.48486372091146257, 0.5151362790885374]	Van Dog
True	False	[0.5575291620674792, 0.4424708379325208]	nan
True	False	[0.5665436668638206, 0.43345633313617943]	nan
True	False	[0.5665436668638206, 0.43345633313617943]	nan

Table 6: Mismatches for Logistic Regression.

y	y_hat	probabilities	x
True	False	[0.5951209415798449, 0.40487905842015515]	nan
True	False	[0.5951209415798449, 0.40487905842015515]	nan
True	False	[0.5951209415798449, 0.40487905842015515]	nan
True	False	[0.6011852449918378, 0.39881475500816216]	nan
True	False	[0.6011852449918378, 0.39881475500816216]	nan
True	False	[0.6011852449918378, 0.39881475500816216]	Holy Smokes
True	False	[0.6011852449918378, 0.39881475500816216]	Mac BBQ
True	False	[0.6048256520648894, 0.39517434793511064]	nan
True	False	[0.6048256520648894, 0.39517434793511064]	nan
True	False	[0.6048256520648894, 0.39517434793511064]	nan

In Figure 7, we can observe the confusion metrics and misses in the cross validation of the model. We can observe that the Logistic Regression is not particularly good at discriminating, since it is evidently favouring the "not hot dog" class, as we expected from the coefficients calculated. We can also see the patterns for the mismatches in Table 6. We can see the model failed to identify some of the "easy" catches we found previously, such as identifying blanks or keywords like "dog", which do not skew the balance enough in favour of the target class.

Naive-Bayes

Table 7: Naive-Bayes cross validation scores and times.

fit_time	score_time	test_score	train_score
0.0011	0.0004	0.6923	0.8
0.0007	0.0003	0.7692	0.82
0.0007	0.0003	0.7692	0.82
0.0007	0.0003	0.9167	0.7843
0.0006	0.0003	0.75	0.8235

Table 8: Mismatches for Naive-Bayes.

y	y_hat	probabilities	x
False	True	[0.3642228229895249, 0.6357771770104746]	Van Dog
True	False	[0.6544133650223363, 0.34558663497766406]	nan
True	False	[0.6839649670539567, 0.3160350329460436]	nan
True	False	[0.6839649670539567, 0.3160350329460436]	nan

Table 8: Mismatches for Naive-Bayes.

y	y_hat	probabilities	x
True	False	[0.7066548087250116, 0.29334519127498854]	nan
True	False	[0.7066548087250116, 0.29334519127498854]	nan
True	False	[0.7066548087250116, 0.29334519127498854]	nan
True	False	[0.7247833230416214, 0.27521667695837865]	nan
True	False	[0.7247833230416214, 0.27521667695837865]	nan
True	False	[0.7247833230416214, 0.27521667695837865]	Holy Smokes
True	False	[0.7247833230416214, 0.27521667695837865]	Mac BBQ
True	False	[0.7834187709904163, 0.21658122900958396]	nan
True	False	[0.7834187709904163, 0.21658122900958396]	nan
True	False	[0.7834187709904163, 0.21658122900958396]	nan

Finally, we will be testing the Naive-Bayes model, which is also a relatively simple model that does not tend to over-fit as much, just to see which model is best. The cross validation scores and time are shown in Table 7. We can also see that it performs better than Decision Tree and Dummy Classifer with less overfit, however, the the test scores are not consistent, similar to Logistic Regression. As shown from Figure 8, the confusion metrics are quite similar to the Logistic Regression, as we can see is not particularly good at identifying hot dog features. From the mismatches in Table 8, we can also see a similar pattern as the Logistic Regression model, failing to identify the "obvious" patterns we stated in the earlier.

Model Comparisons

As shown in Table 9, we can see a negligible difference between Naive-Bayes and Logistic Regression. But we will choose to move forward with the Naive-Bayes classifier, as it is also a relatively simple model with slightly faster fitting time.

Table 9: Performance comparison across the different models.

	Unnamed: 0	Dummy	DecisionTree	LogisticRegression	NaiveBayes
0	fit_time	0.001	0.001	0.002	0.001
1	score_time	0	0	0	0
2	test_score	0.651	0.567	0.779	0.779
3	train_score	0.651	0.984	0.81	0.81

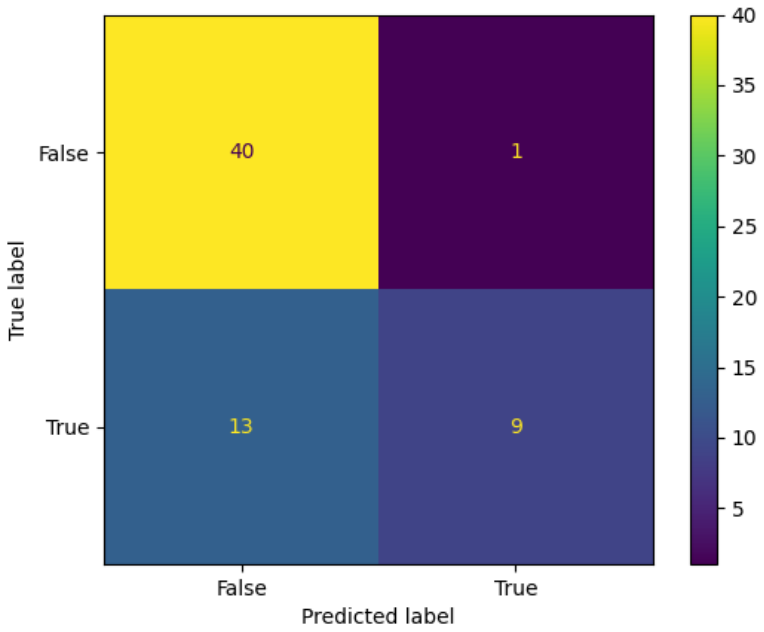

Figure 7: Confusion matrix for the Logistic Regression.

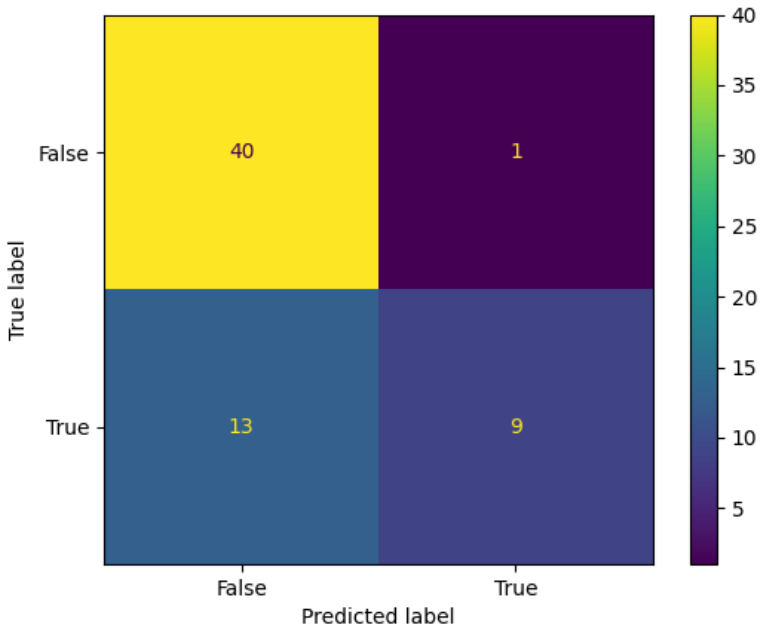

Figure 8: Confusion matrix for the Naive-Bayes classifier.

Best Model Hyperparameter Optimization

From Table 10, we can observe a slight increase in the validation score compared to before. After evaluating it on the test set, the model ended up with a test score of 0.714, which is not too far off from the validation score of 0.778. However, in Figure 9, we can see that, although the model makes good predictions, it still fails to raise the "hot dog alert". In Table 11, we can further see the failed classifications. Save for the first one, which we would even classify as a hot dog stand, it still missed some of the cues we identified at the beginning, meaning this model will probably encounter these limitations in future predictions.

Table 10: Randomized search CV results for best Naive-Bayes model.

rank_test_score	mean_test_score	mean_train_score	mean_fit_time	mean_score_time	param_countvectorizer__max_features	param_multinomialnb__alpha
1	0.779487	0.809569	0.00490127	0.00118113	52	0.737466
1	0.779487	0.793804	0.0043901	0.00207233	42	0.790496
1	0.779487	0.809569	0.00425262	0.00211668	60	0.623982
1	0.779487	0.809569	0.00402508	0.00163946	82	0.810249
1	0.779487	0.809569	0.00519519	0.00224056	61	0.833756

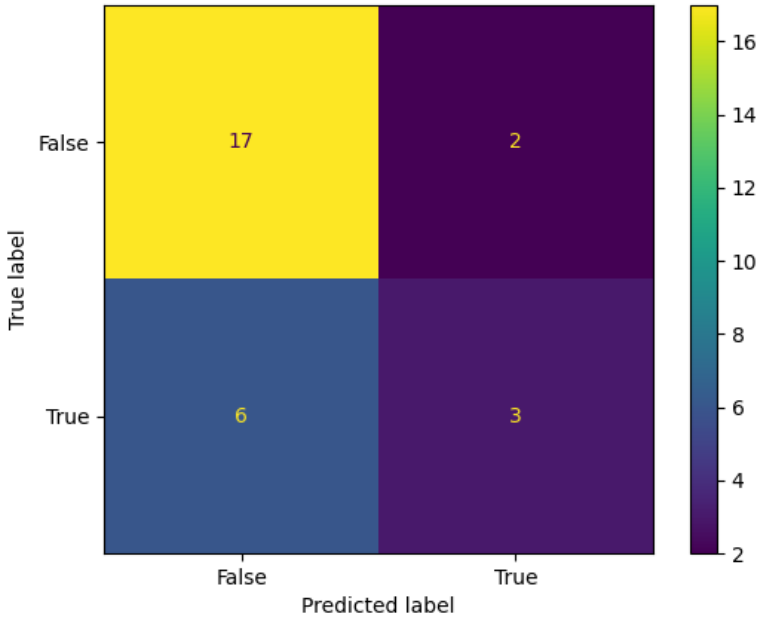

Figure 9: Confusion matrix for the best Naive-Bayes classifier.

Table 11: Confusion matrix for the best Naive-Bayes model.

y	y_hat	probabilities	x
True	False	0.120257	Mr. Tube Steak
True	False	0.265	nan
True	False	0.265	nan
True	False	0.265	nan
True	False	0.265	nan
True	False	0.265	nan
False	True	0.557181	Lemon Sea
False	True	0.830611	Mo's Hot Dog Plus

Discussion

As we saw, there are several limitations to what a Countvectorizer and a binary classification can perform. The relationships that we found within our variables may not linear, as there are some cases where the biases of our more intelligent classifiers, such as the Naive-Bayes and the Logistic Regression, would favor into classifying something as not hotdog, when we noticed from our EDA that it was those specific cases of no name where the model should have predicted that that was hotdog stand. This makes for a model that will be particularly good at identifying the majority class, which pretty much makes it comparable to a dummy. And thus, from the limitations of our current estimators, our model is not ready to be used yet for prediction.

Now, depending on the context of our problem, we may lean in favour of having a model that is really good at predicting when something is not a hotdog, versus wanting a model that is really good at predicting when something is a hotdog. Let's say we have someone who doesn't really like hot dogs that much. We would prefer a model that probably outputs more consistently or classifies non-hot dog places as non-hot dog places, where this default probably like opens up possibilities for someone looking for options that are likely not hot dog related. And it is not too terrible if a hot dog place slides in, given that it is the fewer of the bunch. For that particular case, our model is probably the one fitting better into that narrative, as it is consistent enough to determine or correctly classify the null class, even though it's not as good as classifying the positive class. Now, in the context of someone really craving a hot dog and wanting to be very sure that that is a hot dog place, then probably the best model that we trained will not fit into that description as much, since it's not particularly good at predicting a class. In that case, it would have been better to train a decision tree, which we saw had much higher bias in identifying the hot dog class.

A test accuracy of 0.71 shows that our model is still a work in progress, but it shows promising results from the confusion matrix, where it only has 1 false positve. There are also 6 false nega-

tive, as that is a byproduct of how the model learned the patterns. Some challenges in the data set include the size, which is a small dataset with only around 90 entries. Another challenge is the imbalance of classes. It would be ideal if each class would represent roughly 50–50% of the samples. We believe that the imbalance was not severe, so we didn't make any adjustments. In the future, we can consider balancing the weight of classes during hyperparameter tuning for further optimization.

Finally, we could also add, for future iterations, or as a different research question, whether a Support Vector Machine (SVM) would perform better given the conditions we have mentioned. It is likely that a nonlinear model will fare better since we have found some instances where the natural bias of Logistic Regression and Naive-Bayes have pushed the model to incorrectly classify some of the examples.

References

Vancouver, City of. 2025a. "City of Vancouver Open Data Portal." https://opendata.vancouver.ca/pages/home/.

———. 2025b. "Street Food Stationary Vending Permits." https://vancouver.ca/doing-business/selling-food-on-vancouver-s-streets.aspx.