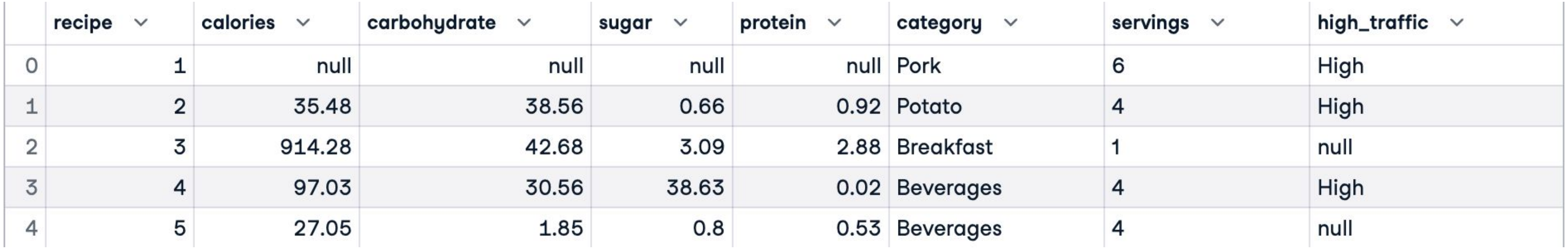
Alexey Arkhangelskiy

Predicting High Traffic Recipes

Data validation: missing values



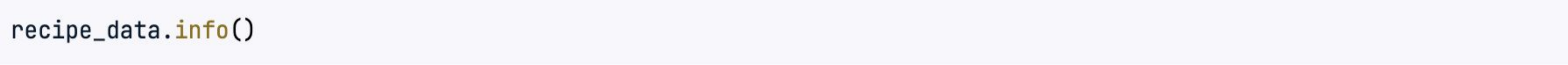
	recipe	calories	carbohydrate	sugar	protein	category	servings	high_traffic
0	1	null	null	null	null	Pork	6	High
1	2	35.48	38.56	0.66	0.92	Potato	4	High
2	3	914.28	42.68	3.09	2.88	Breakfast	1	null
3	4	97.03	30.56	38.63	0.02	Beverages	4	High
4	5	27.05	1.85	0.8	0.53	Beverages	4	null

- We observe some null values in the data.

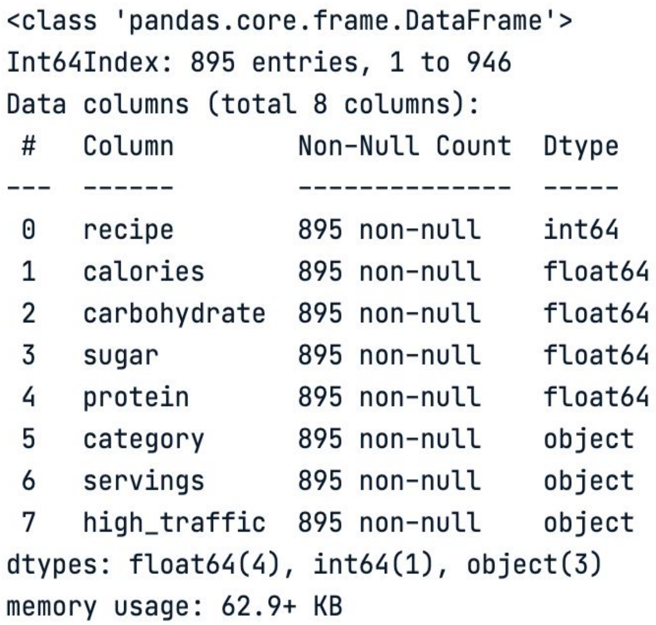
- We noticed a high proportion of missing values in the 'high_traffic' variable. According to the task description, 'high_traffic' is marked as “High” if site traffic was high when the recipe was displayed. So, we'll assume that null values in 'high_traffic' actually mean 'low' traffic. We'll replace the null values with 'low'.

- We will later find that the 'category' variable plays a significant role in predicting traffic response. Therefore, we will not remove rows with missing numeric data, as they still contain a category. Instead, we will impute the missing values using the mean of each category and the number of servings.

Data validation: data types



```
recipe_data.info()
```

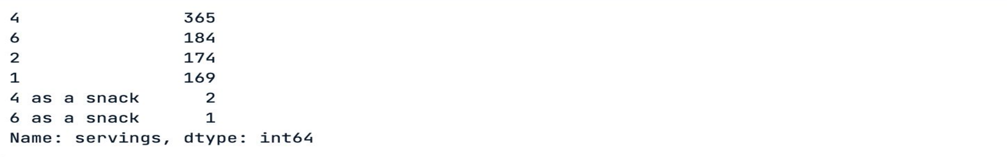


```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 895 entries, 1 to 946
Data columns (total 8 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   recipe        895 non-null    int64
 1   calories      895 non-null    float64
 2   carbohydrate  895 non-null    float64
 3   sugar         895 non-null    float64
 4   protein       895 non-null    float64
 5   category      895 non-null    object
 6   servings      895 non-null    object
 7   high_traffic  895 non-null    object
dtypes: float64(4), int64(1), object(3)
memory usage: 62.9+ KB
```

- In the task description the 'serving' feature variable is presented as 'numeric, number of servings for the recipe.'

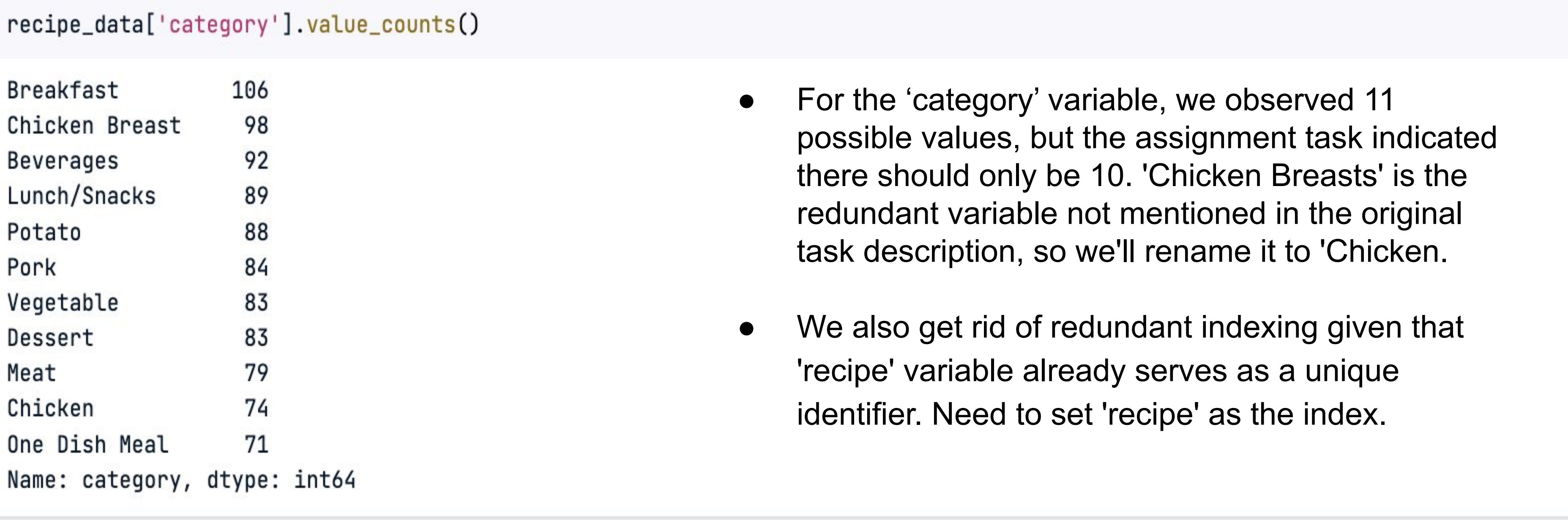
- Here, however, the 'servings' variable has an ‘object’ datatype.

- We rename the ambiguous string values that ‘servings’ can take to numbers and then convert the datatype of the variable to integer



```
4               365
6               184
2               174
1               169
4 as a snack      2
6 as a snack      1
Name: servings, dtype: int64
```

Data validation: data types



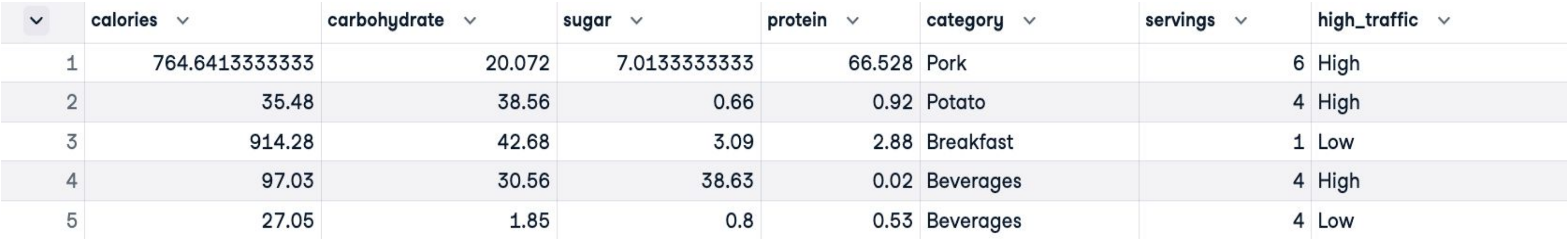
```
recipe_data['category'].value_counts()
```

```
Breakfast         106
Chicken Breast     98
Beverages          92
Lunch/Snacks       89
Potato             88
Pork               84
Vegetable          83
Dessert            83
Meat               79
Chicken            74
One Dish Meal      71
Name: category, dtype: int64
```

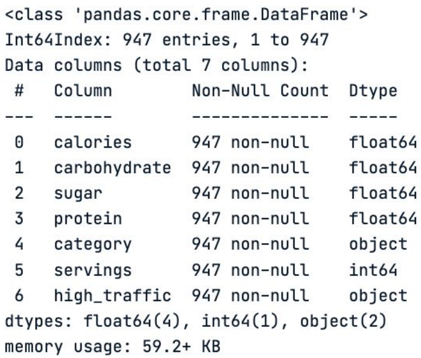
- For the ‘category’ variable, we observed 11 possible values, but the assignment task indicated there should only be 10. 'Chicken Breasts' is the redundant variable not mentioned in the original task description, so we'll rename it to 'Chicken.

- We also get rid of redundant indexing given that 'recipe' variable already serves as a unique identifier. Need to set 'recipe' as the index.

Data validation: cleaned data

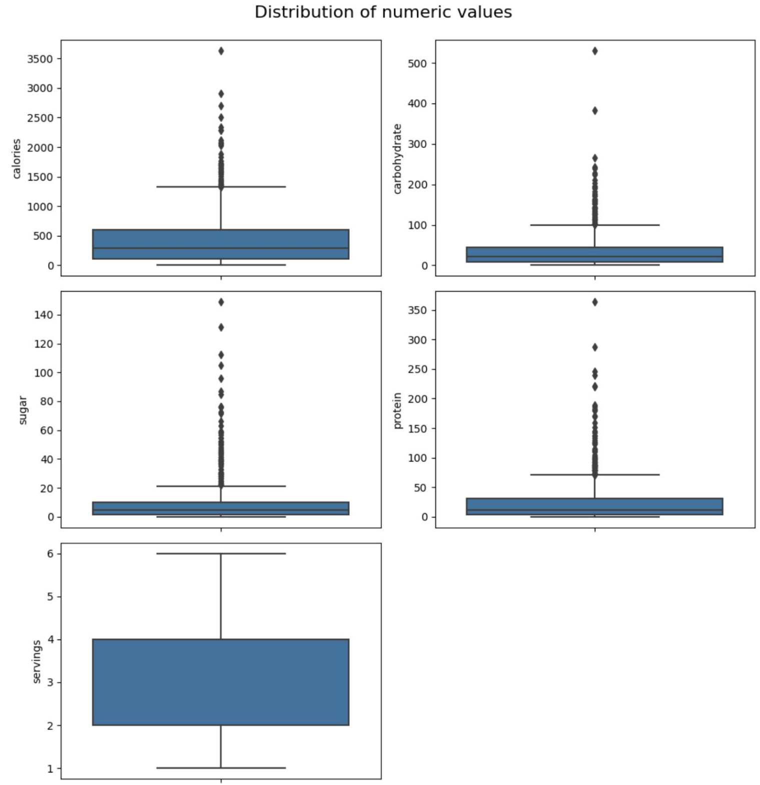


	calories	carbohydrate	sugar	protein	category	servings	high_traffic
1	764.6413333333	20.072	7.0133333333	66.528	Pork	6	High
2	35.48	38.56	0.66	0.92	Potato	4	High
3	914.28	42.68	3.09	2.88	Breakfast	1	Low
4	97.03	30.56	38.63	0.02	Beverages	4	High
5	27.05	1.85	0.8	0.53	Beverages	4	Low



```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 947 entries, 1 to 947
Data columns (total 7 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   calories      947 non-null    float64
 1   carbohydrate  947 non-null    float64
 2   sugar         947 non-null    float64
 3   protein       947 non-null    float64
 4   category      947 non-null    object 
 5   servings      947 non-null    int64  
 6   high_traffic  947 non-null    object 
dtypes: float64(4), int64(1), object(2)
memory usage: 59.2+ KB
```

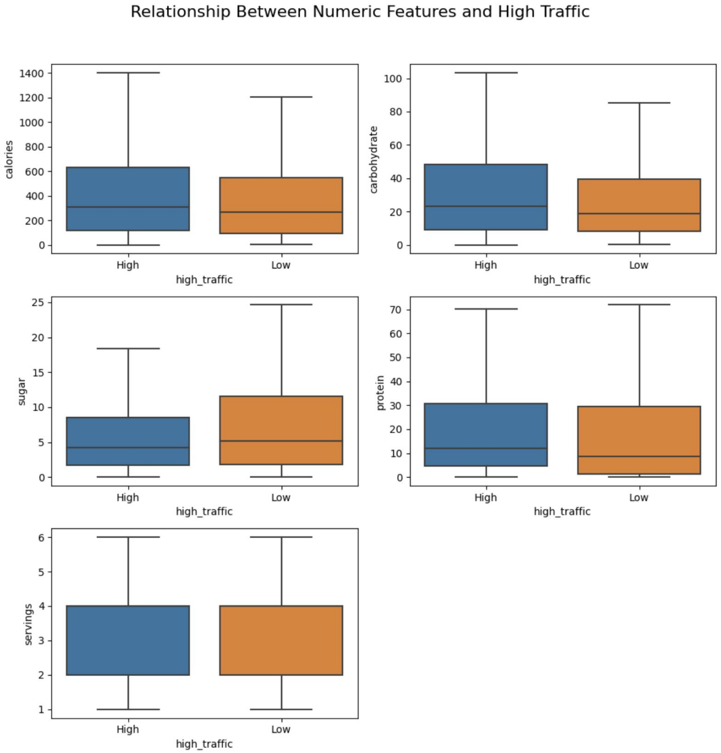
Exploratory Data Analysis: numeric feature variables



- We aim to examine the distribution of values for each numeric feature to identify any potential outliers that might influence subsequent model building.

- The sample-by-sample analysis of each extreme outlier showed that there are signs of erroneous data input. We remove outliers with values that appear nutritionally nonsensical. Additionally, we filter out rows where the gram counts for protein, carbohydrates, or sugar exceed the calorie count, as such proportions are implausible.

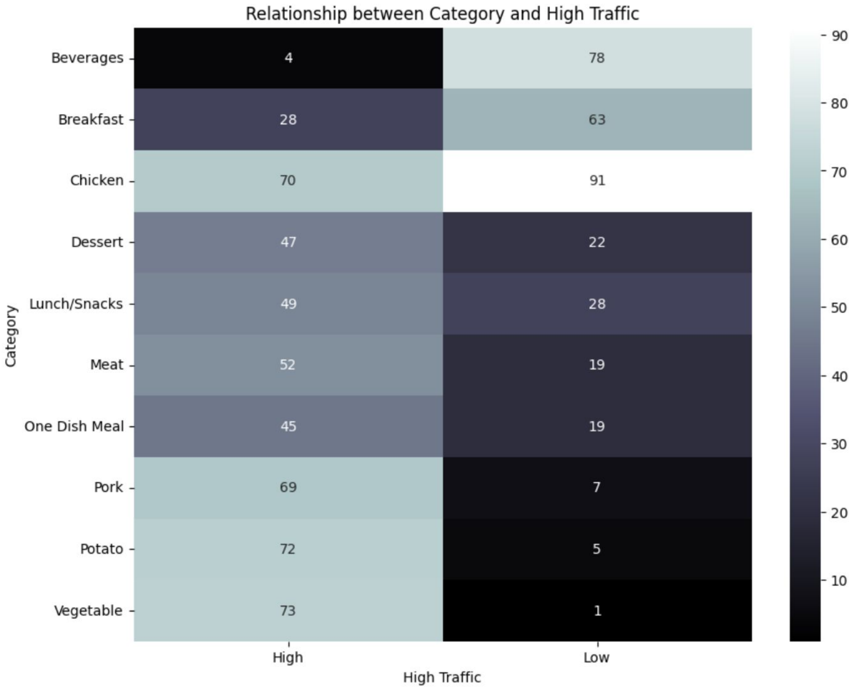
Exploratory Data Analysis: numeric variables vs. traffic



- This chart aims to clarify any relationship between these numerical variables and user traffic.

- The average sugar and calorie counts were comparable for both high and low traffic recipes, suggesting these variables don't significantly influence traffic. This observation held true for other numeric variables as well.

Exploratory Data Analysis: recipe’s category vs. traffic



- Recipe categories appear to influence traffic patterns. For instance, 'Vegetable', 'Potato', and 'Pork' recipes are notably popular, while categories like 'Beverages' attract fewer visits.

- The subsequent chi-square contingency test further confirms the observed dependency.

Machine Learning Preprocessing

- We observed that the initial distribution of the numeric variables is strongly right-skewed. To achieve a more Gaussian-like distribution, we'll apply a log transformation.

- One-Hot Encoding for 'Category': This transforms the 'category' variable into multiple binary columns, ensuring each category can independently affect the model's predictions.

- Label Encoding for Response Variable: This changes the 'high' traffic label to 1 and the 'low' traffic label to 0, converting textual classifications into numerical values for model compatibility.

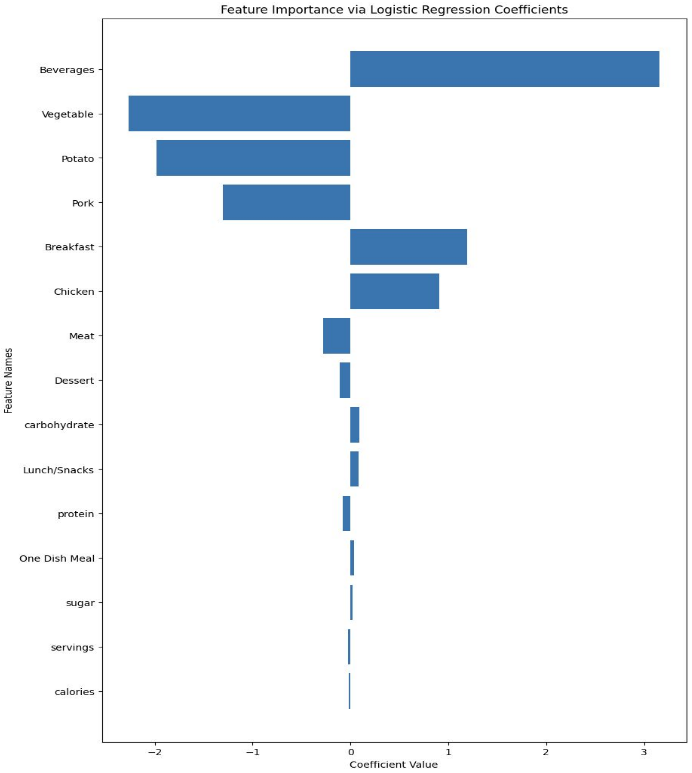
- Feature Scaling: This adjusts the numeric variables to a consistent scale, ensuring algorithms that rely on numeric computations, like distance measurements or gradient descent, work optimally.

Model Construction

- Response Variable: 'high_traffic' is binary variable that takes either 'High' or 'Low’ values. As such, we will test models applicable to binary response variable, such as Logistic Regression, K nearest neighbors, and Gradient Boosting Tree.

- Evaluation Metric: Our key metric is precision. Given the vast recipe pool and the few highlighted on the main page, our priority is choosing correctly, not necessarily catching all popular ones. This approach favors precision over recall or accuracy. The aim is to confidently showcase, say, 5 top recipes to guarantee a positive user experience.

Logistic Regression & K-nearest neighbors



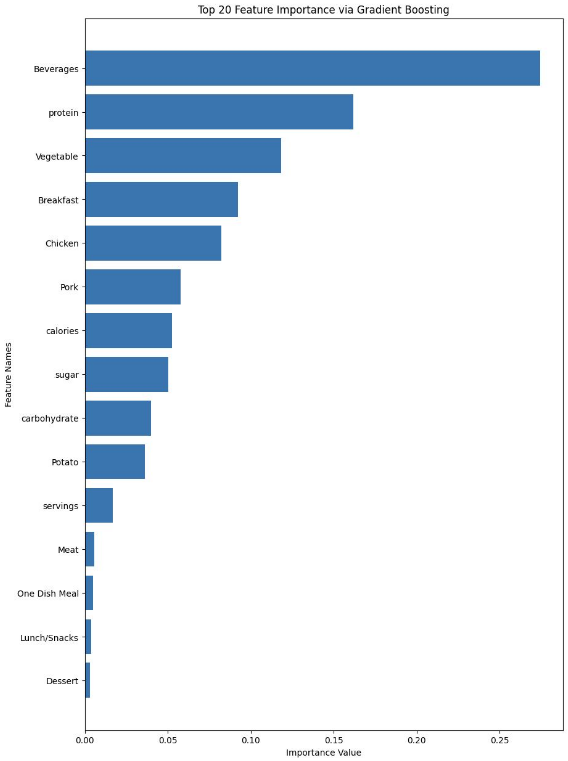
- Logistic Regression Result: Achieved an 75.8% precision and best cross-validation score is 68% precision.

- K Nearest Neighbours: With optimized settings, it hit around 64% precision score with best cross-validation being 75%.

- Feature Importance Chart: Showcases variables most correlated with high traffic.

- Numeric Feature Insight: Confirming the EDA insight, numeric features don't significantly influence traffic predictions.

Gradient Boosting Classifier



- The Gradient Boosting Classifier is the only model that consistently hits a precision score above 80%. Using 1000 bootstrap samples, we determine that 91.6% of iterations hits precision score above 80%.

- We made an importance chart using the gradient boosting classifier and compared it to the logistic regression's. While the category hierarchy is similar in both, the gradient boosting emphasizes numeric variables more. We should refine the nutritional data further to ensure this trend isn't just noise from faulty data.

Baseline prediction

- During our Exploratory Data Analysis, we found that about 60% of recipes achieve high user traction. This gives a baseline precision of 60%; if we predicted high traffic for every recipe randomly, we'd be correct 60% of the time.

- Our objective was to build a model that predicts a recipe's popularity with 80% precision, significantly reducing the likelihood of showcasing unpopular recipes.

Business Metric Choice

- We prioritized precision as our main metric. While missing out on some potentially popular recipes is acceptable, mistakenly labeling a 'low' traffic recipe as 'high' is not. The goal isn't necessarily to identify all 'high' traffic recipes, but to ensure those we do label and showcase as popular are genuine hits. High precision ensures this accuracy and confidence.

- We need to minimize the false positives in our prediction, which is the essence of prioritizing precision

- Out of the three models, the Gradient Boosting Classifier stands out, achieving an 80% precision score over 90% of the time. We've met our goal of 80% accuracy in predicting popular recipes.

Business Focus / Going Further

- Our new method is more systematic in predicting recipe popularity compared to the older subjective preference-based approach.

- From the feature importance chart, categories like 'Vegetable', 'Potato', and 'Pork' often drive high traffic. Conversely, 'Beverage', 'Chicken', and 'Breakfast' recipes tend to underperform.

- Over 10% of our data had questionable nutritional values, so further cleaning is essential. Therefore, we shouldn't base decisions on the protein or calorie counts, despite their high importance in the gradient boosting tree's results. Need to consult with the data engineer or nutritionist to further validate the nutritional data.

- If we need to spotlight, for example, 5 recipes weekly, picking one from each top-performing category would be a strategic move.

Thank you