Mark Kipper

Final Report:
How to Identify the Theme of a Lego set
(Every theme is Awesome!)

Problem Statement

Lego is a plastic, self-locking brick system patented by the Lego Group in 1958. Beginning in 1978, the Lego Group began marketing a series of sets based on the present, past, and future. These were termed a “system within the system” and led to the birth of the idea of Lego themes. The first themes were Town, Castle, and Space (representing the present, past, and future, respectively.) All sets released prior to this were labeled as the “System” theme. Since then, new themes have been introduced as older, less popular themes have been discontinued. 1999 saw the introduction of licensed themes such as Star Wars. There are currently 444 themes in the Lego database. This data includes all sets made since 1949.

The question I intend to answer is: Can a Lego theme be identified given the year of production, color id, part name, part number, etc.?

By using the Lego Rebrickable database, I have created a tool to identify the theme associated with a set of Lego bricks based on the number of parts, the color of the parts, the year it was produced, etc. Paired with sales data, this can be used by the engineers to design newer models based on the best-selling, or most popular, themes. Likewise, there is an app called Brickit that scans your loose, scattered Lego parts and generates building instructions for new models. Using my tool, a Theme element can be incorporated into this app, and it can generate theme-specific building ideas spanning the history of the Lego catalog, bringing new life to old bricks.

Data Wrangling

The Lego Rebrickable database contains 12 csv files:

- themes.csv

- colors.csv
- part_categories.csv
- parts.csv
-
-
-
-
-
-
-

part_relationships.csv
elements.csv sets.csv
minifigs.csv
inventories.csv
inventory_parts.csv
inventory_sets.csv
inventory_minifigs.csv

Luckily, none of these tables have NaNs or missing values.

The task is to have one table that lists all the sets and references the theme id. So, I start with the sets table. This table has 18507 rows. From there I merge with the themes table on the theme id in order to identify a name for each theme id. I drop the parent id column from the themes table because it is irrelevant. My next merge is with the inventories table. This gives me an id for each set. This is different than the set_num column and allows me to merge this main table with any remaining tables. I still need part numbers, color, etc. Merging with the inventory_parts table adds a part_num and color_id for each row. Since this table details all parts for each set, my table has increased to 919,288 rows! I then merge the main table with the color table and the parts table to complete it.

Now there are some columns that have useful information in them but are not yet usable for any machine learning models as they contain non-numeric data. For example, the set_name, part_name, and color_name columns. I created columns for each of these that includes the length of its value: set_name_len is created from the set_name column, etc.

The goal is to identify a theme. I then take my dataset and choose a theme. I chose a theme that is among the most common entries in the main table. The theme I chose was the Ninjago theme, however this can be applied to any theme. I identified the theme id and created a new binary column, is_ninjago, that compares each row to the

theme id for Ninjago and fills in a 0 if it is not a Ninjago set and a 1 if it is. I use get_dummies on this column, drop the is_ninjago_False column and rename the is_ninjago_True column to is_ninjago.

The final step is clean-up. I drop a few irrelevant or redundant columns like part_material, is_spare, rgb, and part_cat_id. I renamed a few columns to clarify their contents. The name column was renamed to set_name and name_len to set_name_len.

The final shape of my dataset was 919288 rows with 17 columns. The part_categories, part_relationships, elements, minifigs, inventory_sets, and inventory_minifigs tables were not needed.

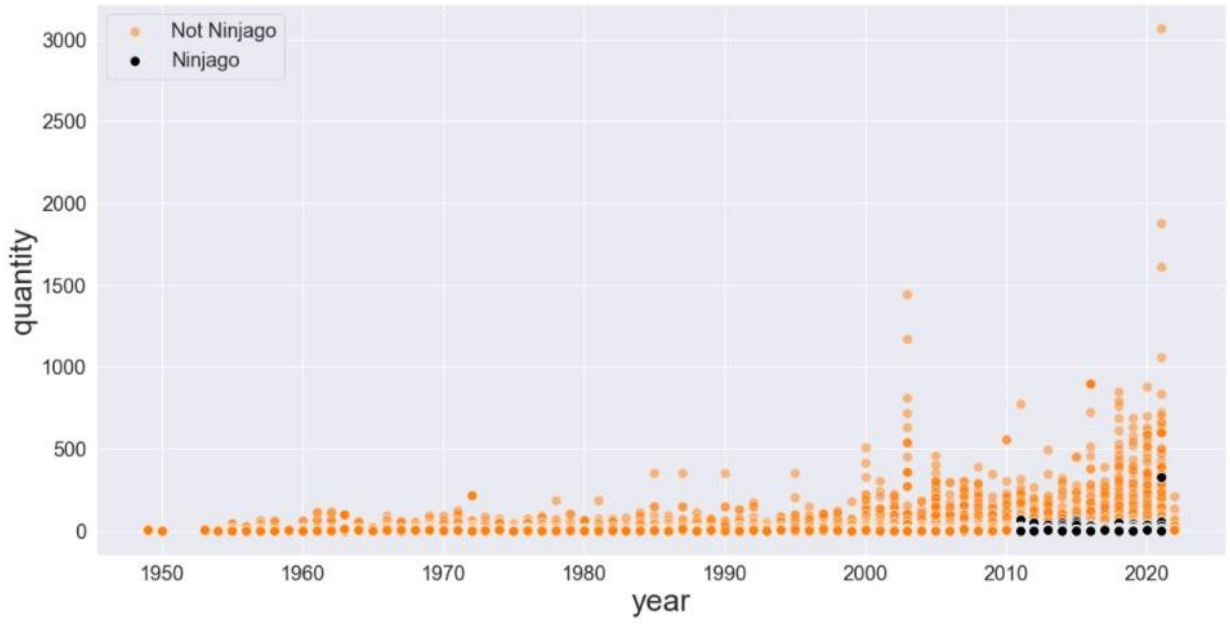
Exploratory Data Analysis

In order to maximize the utility of the main dataset, I created a list of features that contains the usable columns:

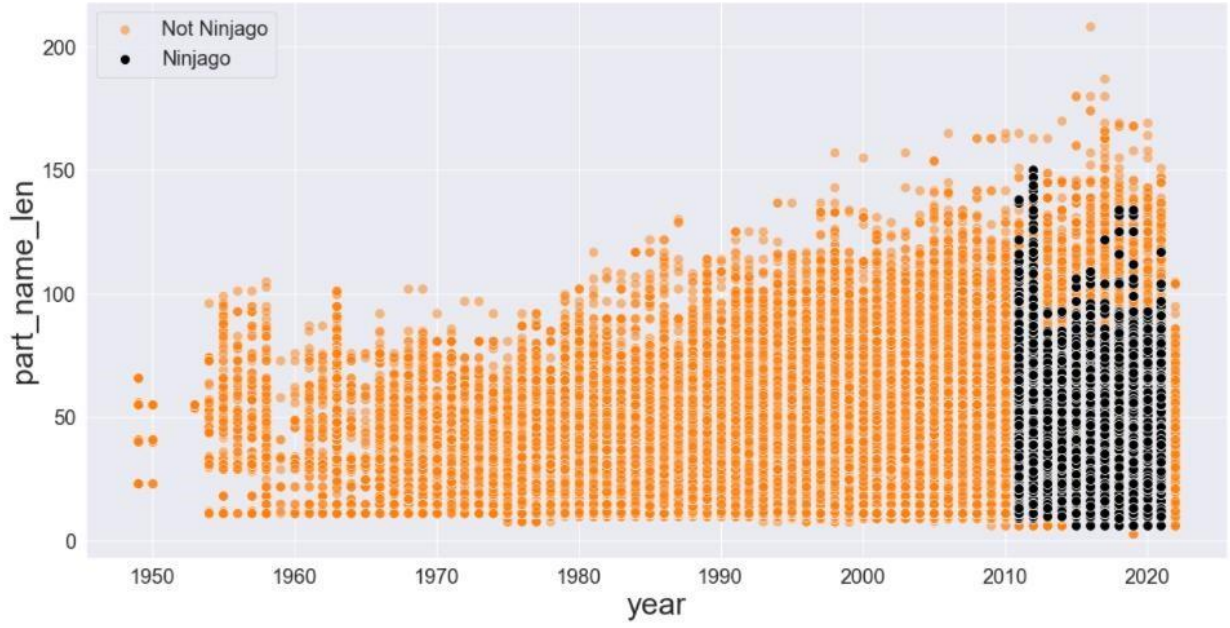
- year
- num_parts
- set_num_len
- set_name_len
- color_id
- quantity
- is_trans
- part_num_len
-
-
-

color_name_len
part_name_len
is_ninjago

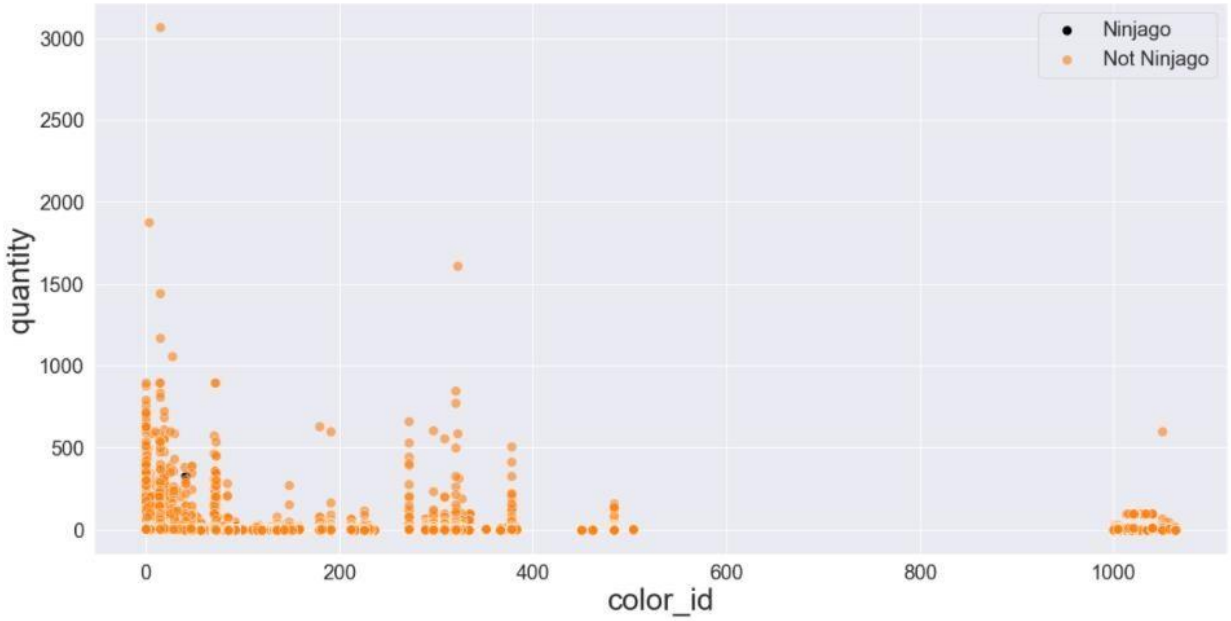
I used Seaborn to plot a pairplot on this subset to see how the features relate to each other. There are a few relationships to take note of. There is a positive correlation between the quantity and the year:



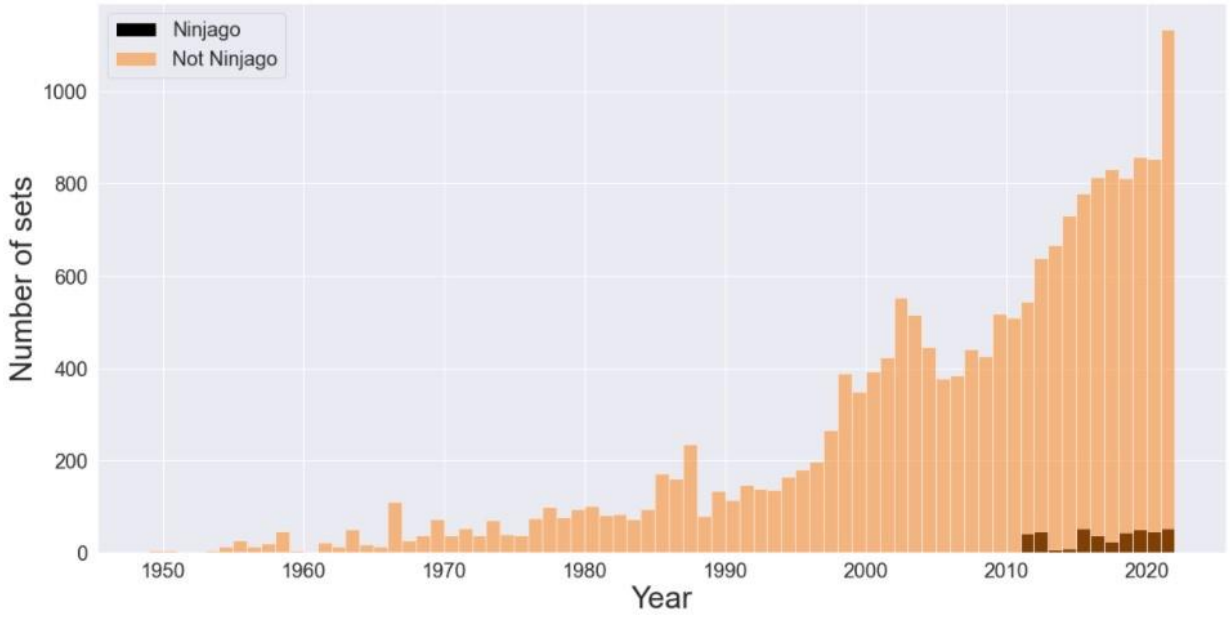
And between the part_name_len and the year:



And a slightly negative correlation between quantity and color_id:



Ninjago as a theme debuted in 2011 and accounts for roughly 3% of all sets ever made by the Lego Group.



This fact makes the data very imbalanced and any theme we try to determine would also be imbalanced.

Because of the imbalance, any model I select would get 97% accuracy just by classifying all sets as non-Ninjago. There are methods by which we can balance this dataset. I created a dataset where I over sampled the minority class. I also created a dataset where I under sampled the majority class. I ran the models on all datasets separately to compare the scores and determine the best method and model for this question.

Model Selection

Because this is a classification question, I tested using Linear Regression, Random Forest, and XGBoost models. Each of these models were used on the cleaned dataset, an oversampled dataset, and an under sampled dataset. All models used a

90% training set and a 10% test set. All models were scored for accuracy and all results for all models were stored in a hyper table for review.

The first test of the Linear Regression model was done on the imbalanced dataset. I set the weighting to “balanced”, and I ran a test for the best “C” value. I ran the test iterating through the feature list adding a feature for each iteration. I did this to see if a certain set of features would yield a higher score. Using the best “C” value I ran the Linear Regression again. Again, iterating through the feature list. Results were stored in the hyper table.

Next, I ran a Random Forest model with the imbalanced data, again setting the weighting to “balanced”, and again iterating through the feature list. Results were stored in the hyper table. I plotted the feature importance attribute of the Random Forest model to determine the most important features.

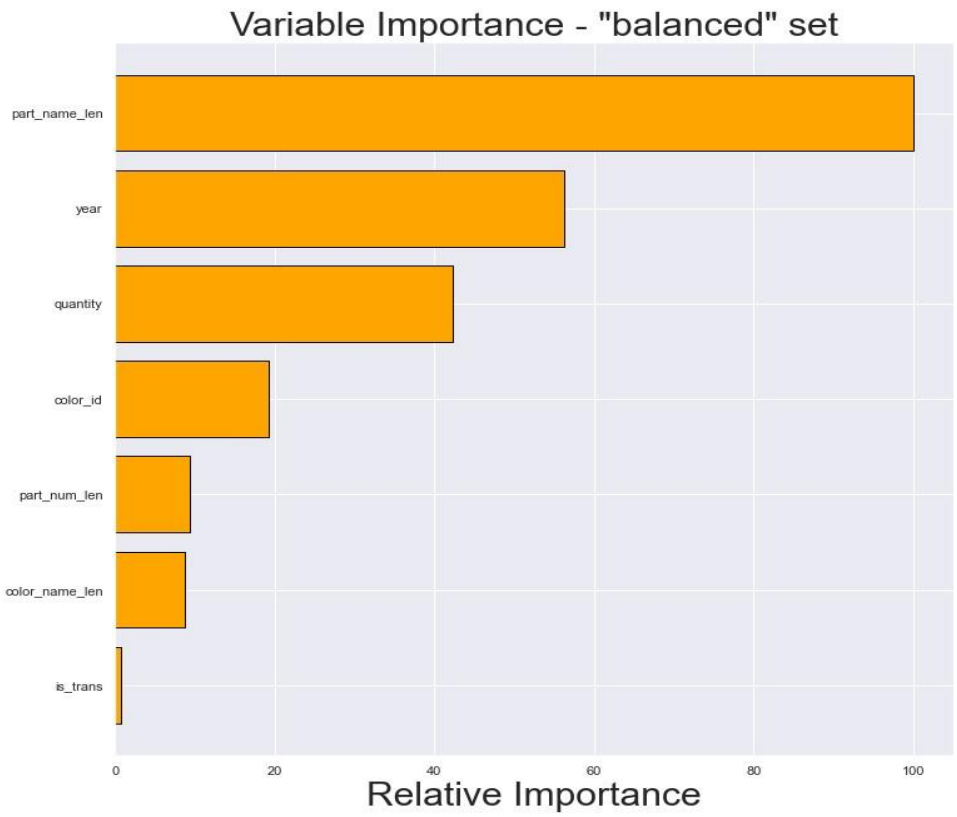
Finally, I used the XGBoost model. I set the weight for the minority class. Results were stored in the hyper table.

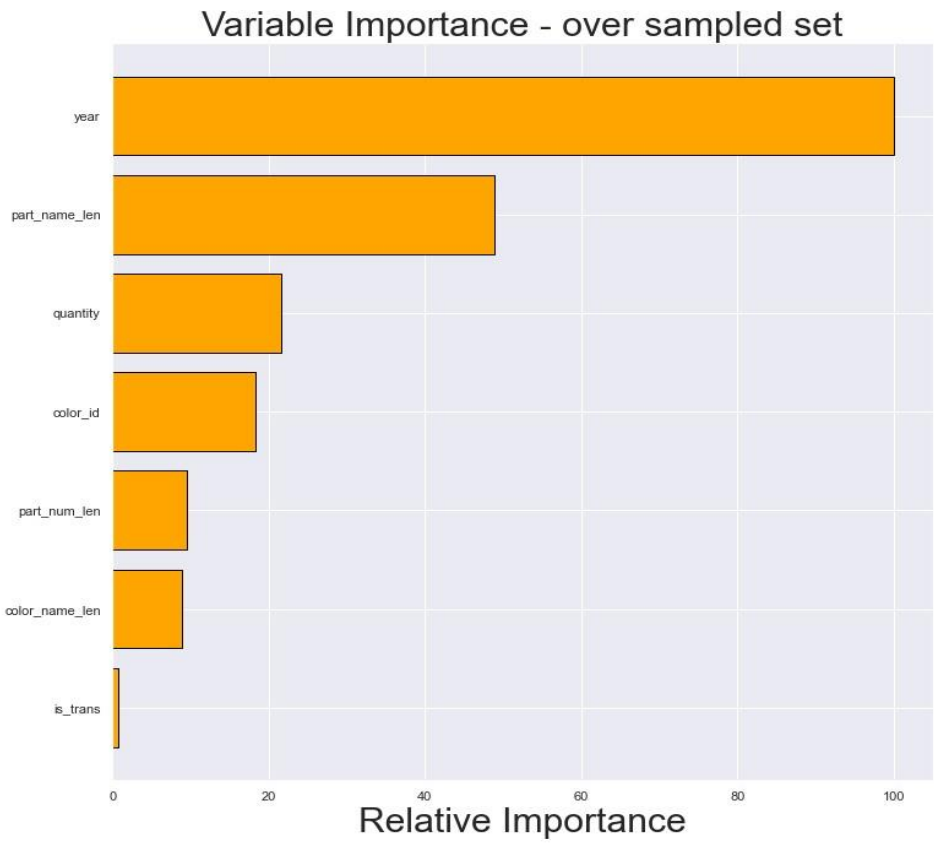
Moving on to the over sampled dataset, I split the data into training and test sets first and then I balanced the training set by oversampling the minority class, with replacement. This gave me a training set with 1,591,130 rows! I trained a Logistic Regression model on the oversampled set, iterating through the feature list, and scored it on the unaltered test set. I then trained a Random Forest model on the oversampled training set, iterating through the feature list, and scored it on the unaltered test set. All results were recorded in the hyper table. I plotted the feature importance attribute of the Random Forest model to determine the most important features. Finally, I trained the XGBoost model. Since the training set was already balanced, I only had to specify the number of estimators, max depth, and learning rate.

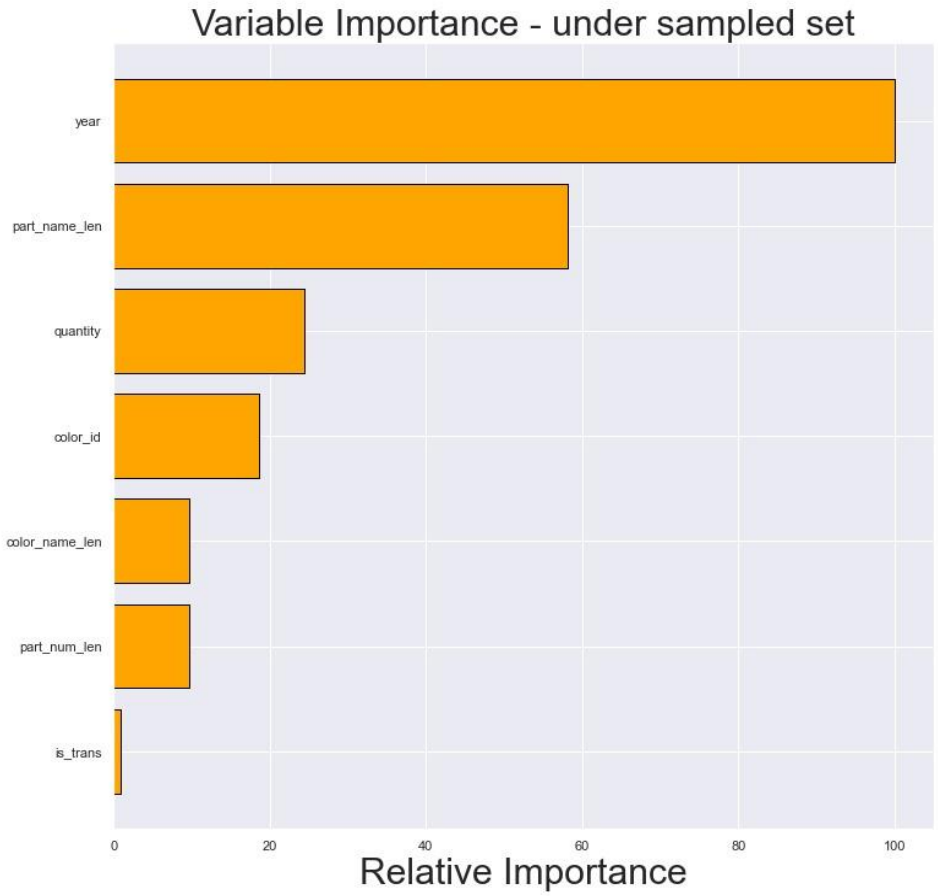
Finally, the under sampled dataset. Again, I split the data into training and test sets. I then under sampled the majority class of the training set giving me a training set with 63,588 rows. I trained a Logistic Regression model on this training set, iterating through the feature list, and scored it on the unaltered test set. I then trained a Random Forest model on the under sampled set, iterating through the feature list, and scored it on the unaltered test set. All results were recorded in the hyper table. I plotted the feature importance attribute of the Random Forest model to determine the most important features. Finally, I trained the XGBoost model. Since the training set was already balanced, I only had to specify the number of estimators, max depth, and learning rate.

Results

Upon review of the hyper tables, the Random Forest models performed the best, as measured by the test set accuracy. The model that performed the best was the model where I set the weight to “balanced.” It had an accuracy of 82.7247%. This is decent accuracy. What is interesting about the Random Forest models is that they were able to achieve such accuracy with only 4 features: part_name_len, year, quantity, and color_id. The ‘year’ feature is intuitive as the Ninjago theme did not arrive until 2011 so any set before 2011 is certainly not a Ninjago set. Similarly, themes tend to have a certain colors or parts so the color_id and part_name_len features are significant. Looking at the plot of the feature importance for the Random Forest models shows the most important features.


Variable Importance - "balanced" set
part_name_len
year
quantity
color_id
part_num_len
color_name_len
is_trans
0
20
40
60
80
100
Relative Importance


Variable Importance - over sampled set
year
part_name_len
quantity
color_id
part_num_len
color_name_len
is_trans
0
20
40
60
80
100
Relative Importance



Future Work

Looking ahead, I believe a function could be written that takes a theme as input and automatically generates the necessary features for prediction. This could yield a more robust model capable of predicting any theme in the entire Lego catalog. Similarly, a multiclass classifier can be trained to identify any theme, not just Ninjago.

This problem successfully identifies if a single piece may belong to a certain theme but ideally the model should identify several pieces to determine what theme they may belong to. For that a different approach may be required. A CountVectorizer could be used to record a vector of part identifiers such as color name length and id, part name length and name, and part number length, and then this vector would be ‘fed’ into the model to predict the theme.