Price Prediction

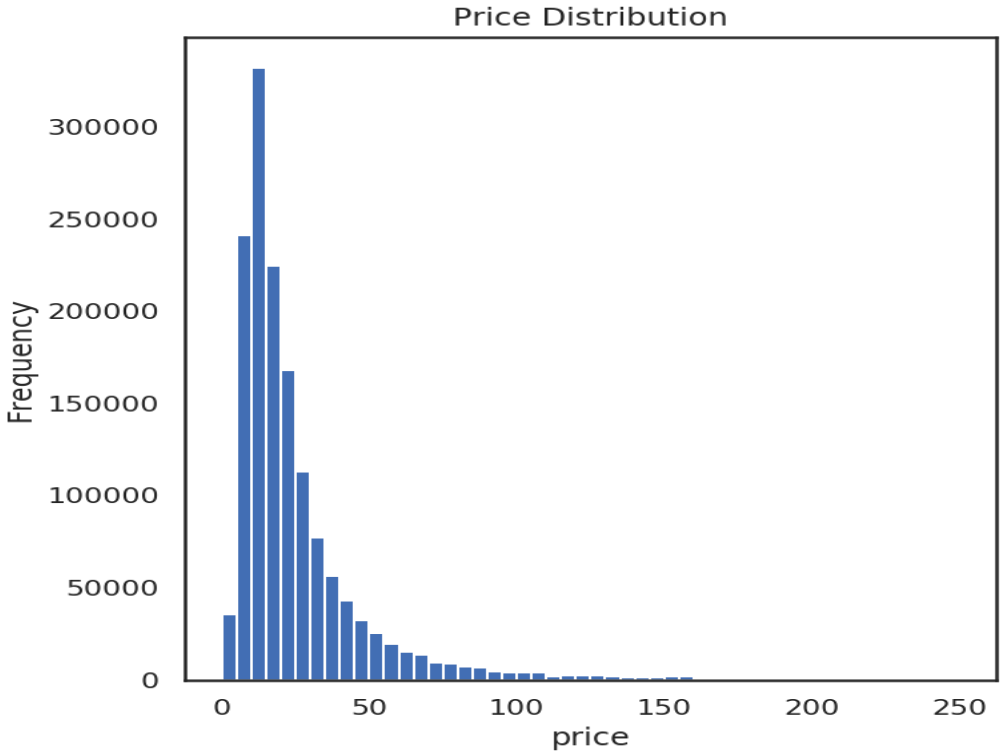
Mahshid Marbouti

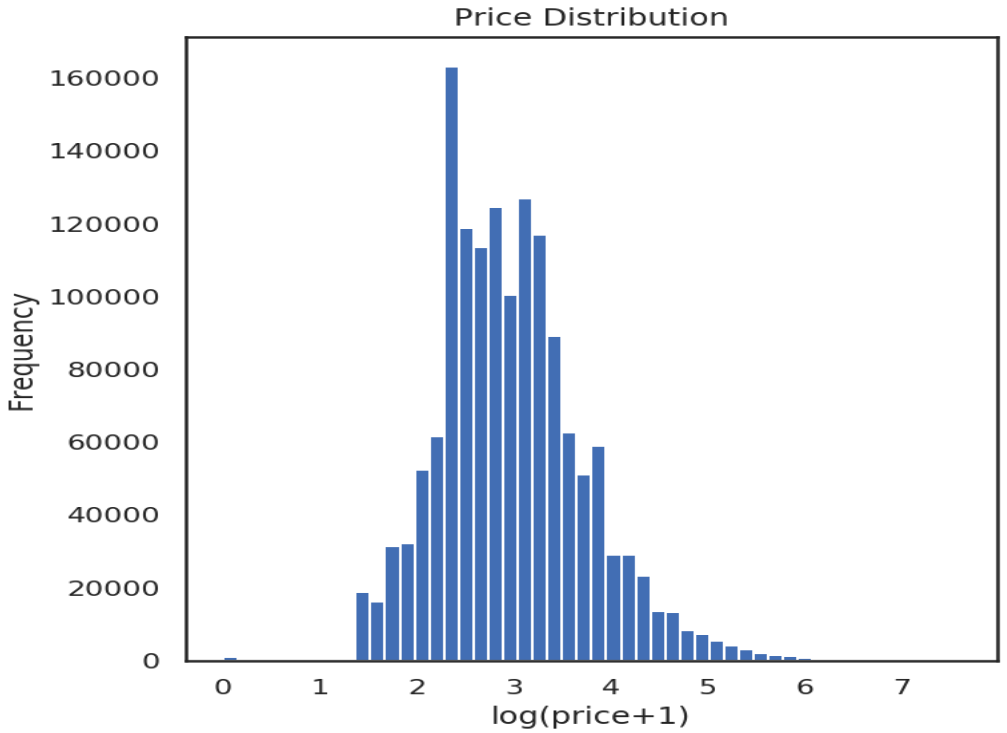
Contents:

- Analyzing Target Variable (Price)
- Exploring the dataset
- Feature Preparation and building the model
- Model Evaluation

Analyzing Target Variable (Price)

The price distribution is right skewed. Most of the values seems to be between 10-20 but there are some high prices in the dataset as well. **Transforming the price to a logarithmic scale** will probably help us achieve better results in this case.





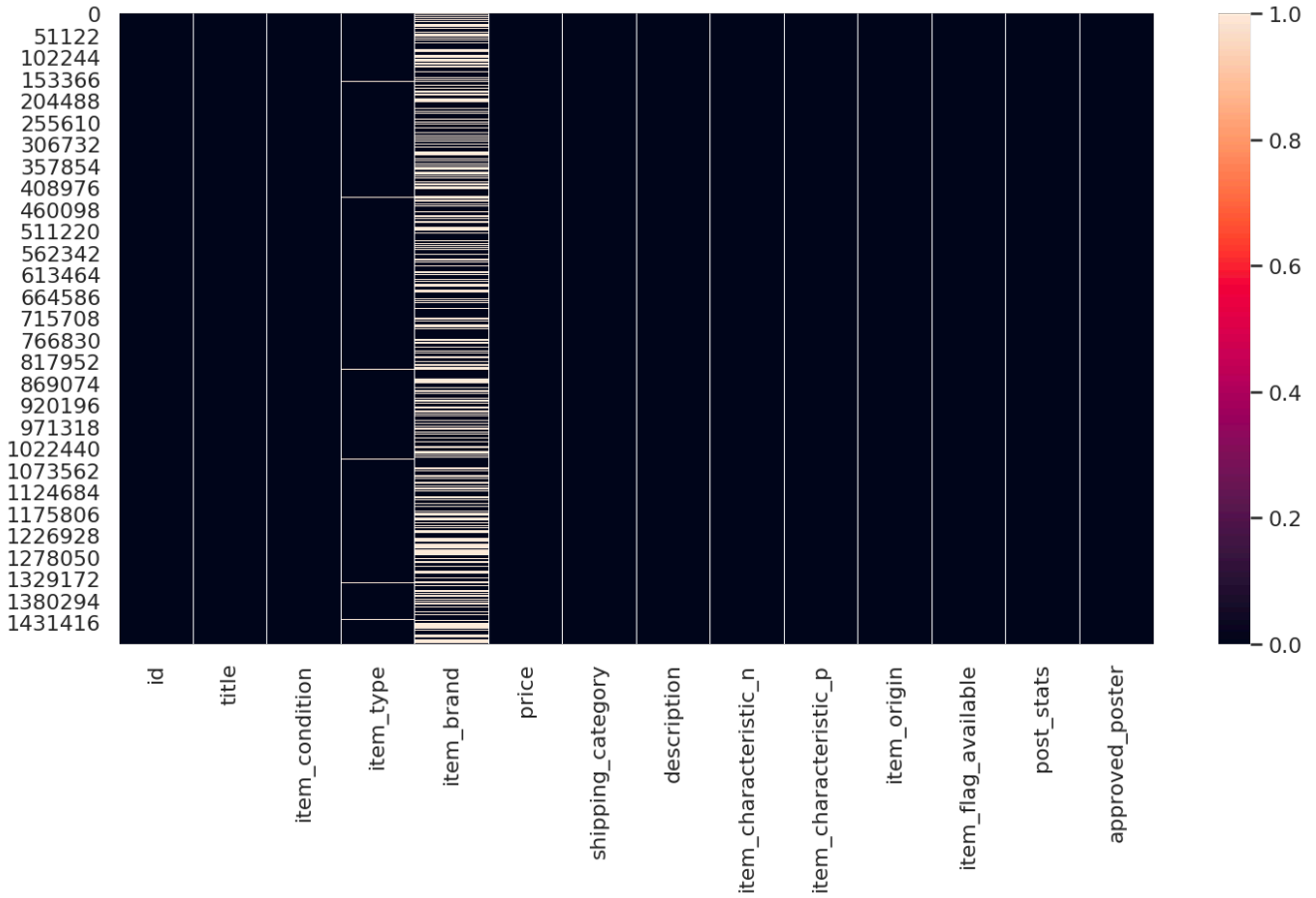
Exploring the dataset

Are there any missing values

According to the heatmap, **item_brand** has the most amount of missing values.

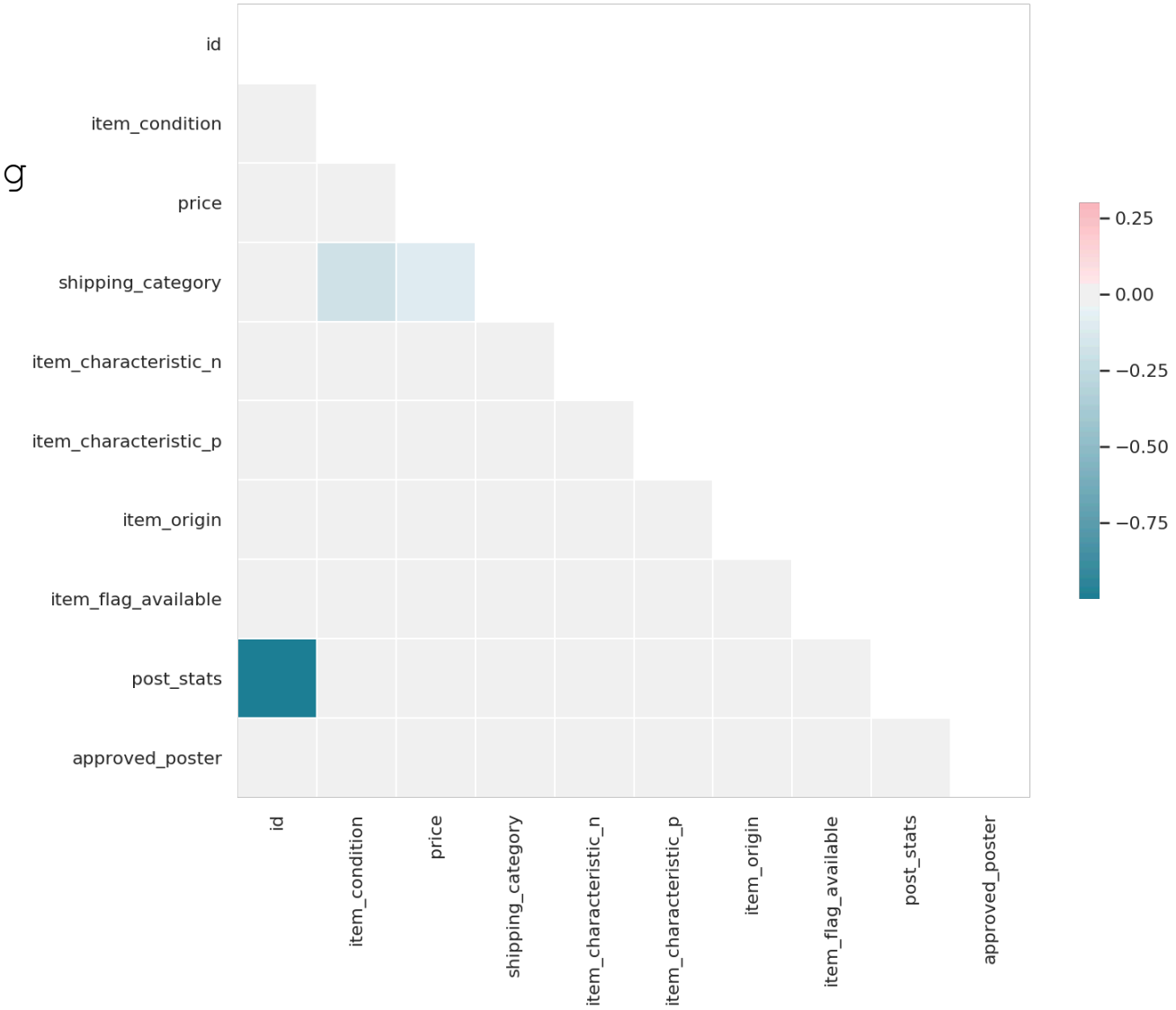
The next attribute with most missing values is **item_type**.

Items with no brand name: 632682
Items with no type: 6327
Items that do not have a description: 4



Correlation matrix

- As you can see, correlation matrix does not show any strong relation between numeric features and the price.

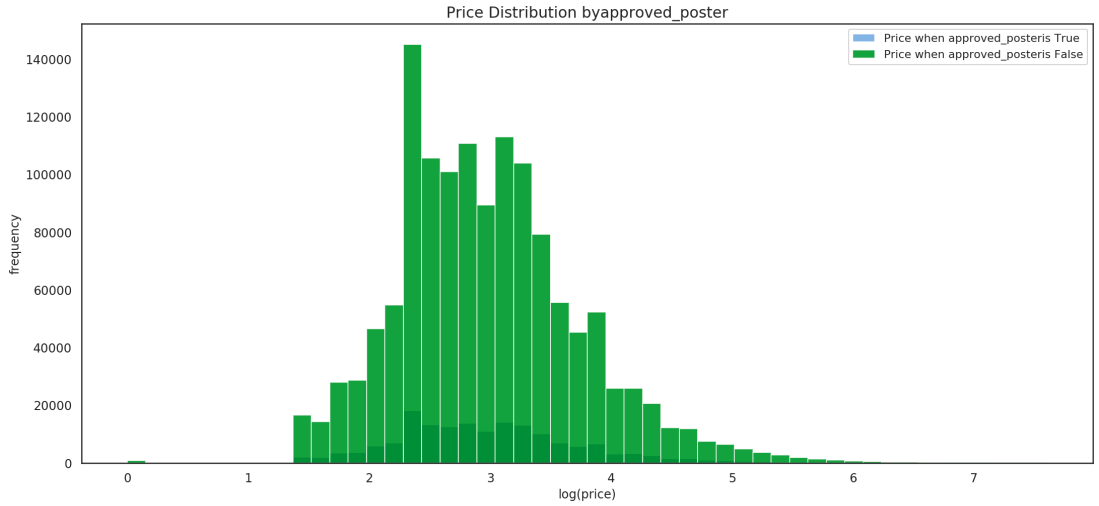


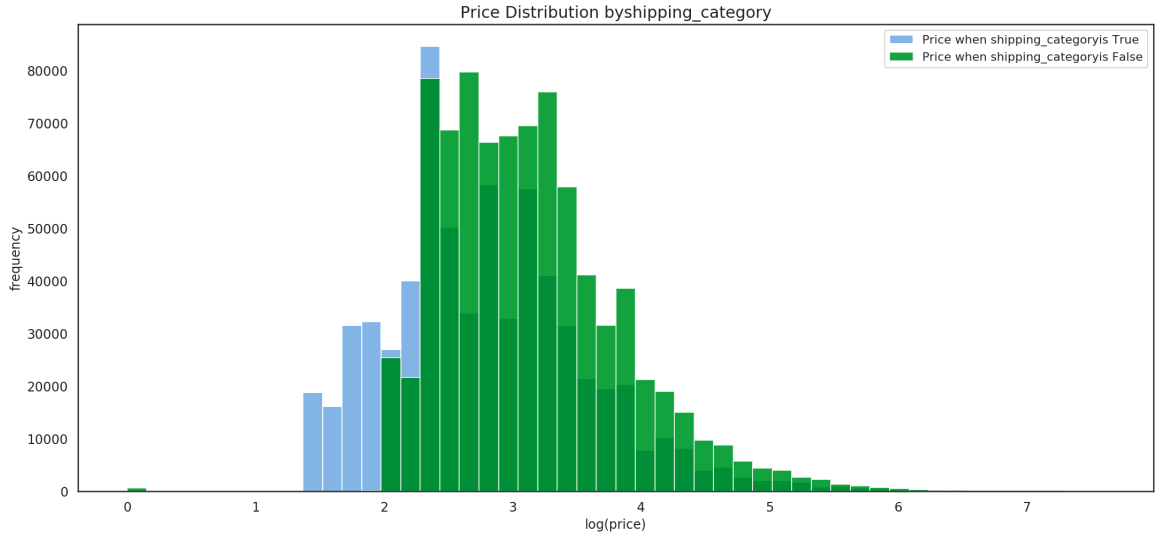
Categorical Features vs Price

Approved Poster: over 88% of the items does not have a verified poster. There seems to be no difference between the average price (on log scale) for approved or non-approved poster.

Shipping Category: In 55% of the items buyer pays for the shipping.

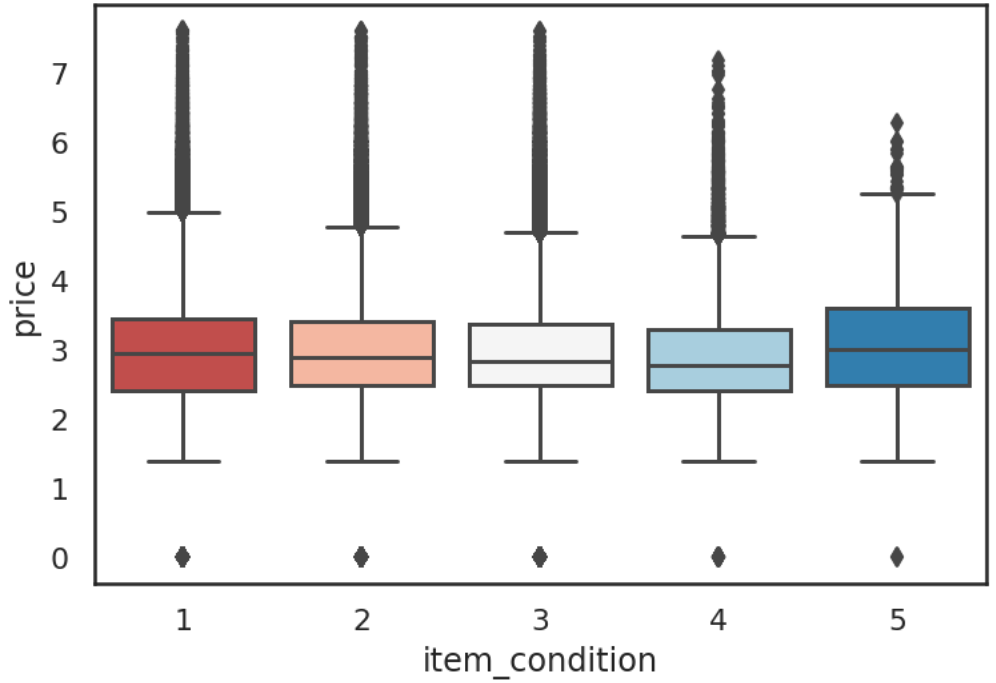
From the price distribution between *if buyer pays for the shipping* or not, Overall the price seems higher if buys pays for the shipping.





Item condition by price

- Price seems to be various between different item condition.



General information

- Data set seems to be a mixture of text, categorical and numeric features. There seems to be no strong correlation between numeric features and the price.

- Textual features are title, item_type, item_brand, and description. Initial exploration of the data set shows item_type has the most meaningful values with 1288 distinct items. While brand has the most number of missing values

- For further info on the dataset look at General information about data section in the notebook.

Data Cleanup

- Replace Missing Values
- Clean up textual feature (Transform them into list of tokens)
  - Tokenize data
  - Remove stop words
  - Keep only alphanumeric characters

Feature Extraction

Feature Selection

- To construct the feature set we need to combine text features with other categorical and numeric features. I followed two approaches:
-

  Traditional NLP Features: convert textual feature (title, description, item_brand, item_type) to traditional NLP features (BoW, tf-idf) . This will create a sparse matrix. We then convert numerical and categorical features as another sparse matrix and this will form our entire feature set.

  Using Doc2Vec: In order to represent each document (in this case product type) by numbers, I used a pretrained word2Vec model provided by google. This model was trained on 100 billion words of Google News and contains 300-dimensional vectors for 3 million words and phrases. For each document, the mean of the embeddings of each word was calculated, so that each document is represented by a 300-dimensional vector.

Building and evaluation of the model

Building the model

Selected a range of regression models:

- LightGBM
- XGBRegressor
- GradientBoostingRegressor
- Ridge
- Lasso

Evaluating the model

- First step is to split the data into train and test sets.
- Next we choose metrics to compare different models. For regression I chose, **Mean Squared Error.**
- Mean Squared Error is a measure that shows how much the predicted values differs from the actual value. Basically it’s a measure of the error around the regression line.

Results for LightGBM

Features	Model	Root Mean Squared Error
Traditional NLP (only type and brand) + other features	LightGBM	0.603767209076339
Traditional NLP (all text features) + other features	**LightGBM**	**0.552617292797008 7**
Doc2VecModel(only type) + other features	LightGBM	0.678349093130193 2

"Other features" are :

- item_condition
- approved_poster
- shipping_category
- item_origin
- approved_poster
- item_characteristic_p
- item_flag_available

Overall we can see traditional NLP features lead to less error comparing to inferred word2vec feature. This might be because we used a pre-trained word2vec model, and also because we only used one text based feature in the word2vec model (rather than all textual features) so it's hard to reach to any conclusion about the most informative features without further exploration.

Other models

I also tried other regression models. Within these models Ridge appear to have less error. Again, to make a concise conclusion, more feature exploration and parameter tuning is required.

Selected Features	Model	Negative mean squared error
Traditional NLP (only type and brand) + other features	XGBRegressor	-0.438756
Traditional NLP (only type and brand) + other features	GradientBoostingRegressor	-0.443028
Traditional NLP (only type and brand) + other features	**Ridge**	**-0.368129**
Traditional NLP (only type and brand) + other features	Lasso	-0.622433

Future Enhancements

- **Train downloaded word2vec model:** For word2vec model I used a pretrained google model. It's better to build a corpus of all available tokens within the dataset and train the model using the new tokens. This might help us achieve better results.Also, I only used item_type column, for word2vec model.

- **Feature extraction:** I'd like to explore feature extraction approaches like PCA to see if it helps in better results. Specially since we have many input features.

- This was just an initial exploration of the dataset, the code needs lots of cleanup, with proper preprocessing, model creation, and evaluation classes.