Assignment 3 – Home Depot Product Search Relevance

Bachelor of Astronomy Data Science
Date of submission 8th of June 2025
Student: Benjamin David Ben Harosh s1977679

Task definition

The objective of this project is to develop a predictive model that estimates the relevance of product search results on HomeDepot.com[1]. In the context of home improvement retail, users expect accurate and timely responses to their search queries, whether they are searching for a specific brand of ceiling fan or browsing materials for a kitchen renovation. Ensuring a seamless and efficient search experience is critical to customer satisfaction and business performance.

To enhance this experience, Home Depot launched a competition challenging participants to design models capable of predicting a *relevance score* that reflects how well a returned product matches a given user query. This score, typically assigned by manual assessment, serves as an internal metric to evaluate the effectiveness of search algorithms. However, manual relevance assessments are time-consuming and inherently subjective. By automating this evaluation process, Home Depot aims to enable faster experimentation and iteration on its search infrastructure, thereby improving its product discovery capabilities.

The task is evaluated using Root Mean Squared Error (RMSE) between predicted relevance scores and the ground truth ratings provided by Home Depot annotators.

Data description

The training dataset consists of 74,067 product-query pairs, each annotated with a relevance score between 1 and 3, indicating how well a returned product matches the user's search query[2]. These pairs correspond to 54,667 unique products. The two most frequently occurring products each appear 21 times: product ID `102893` ("Lithonia Lighting All Season 4 ft. 2-Light Grey T8 Strip Fluorescent Shop Light") and product ID `101959` ("Pressure-Treated Timber #2 Southern Yellow Pine").

The average relevance score is **2.38**, with a **median** of **2.33** and a **standard deviation** of **0.534**. These values suggest that most product-query pairs are considered fairly relevant, with a concentration of scores near 3.0. This distribution is illustrated in Figure (1), which shows a histogram of the annotated relevance scores. However, the variability in the distribution shows the importance of a model capable of capturing subtle differences in semantic match quality. The most frequent brand in the product attributes dataset is `Unbranded` (2954 occurrences), followed by `Hampton Bay` (1723) and `KOHLER` (1389). This pattern reflects a long tail distribution, where a few brands dominate the dataset but many others appear infrequently, which highlights the overall diversity of the brand and supports the use of brand matching features in the model.

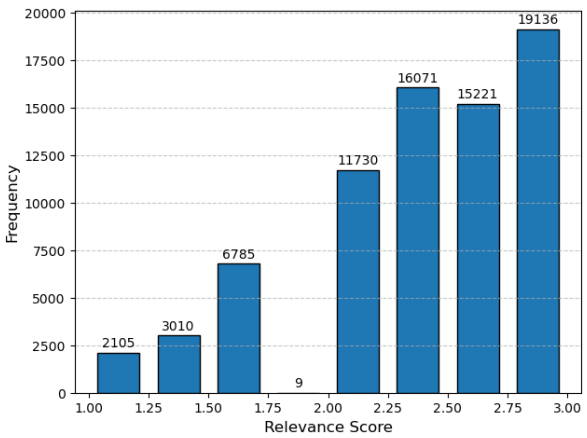

Figure 1: Distribution of annotated relevance scores in the training set, highlighting the skew toward moderately and highly relevant results.

Baseline method

To establish a baseline, we implemented a model based on the original approach by Yao-Jen Chang[3]. The dataset was split into an 80% training set and 20% test set using `train_test_split` with a fixed random seed. A `RandomForestRegressor` with 15 estimators and a maximum depth of 6 was trained on both stemmed and unstemmed versions of the dataset. Following the original code, the regressor was wrapped in a `BaggingRegressor` with 45 base estimators and a subsample ratio of 0.1, which helped reduce variance in the predictions.

The following features were considered for the baseline: **Query length**: Number of tokens in the search query. **Common words**: Overlap between the query and product title/description. Initial evaluation on the test split yielded an RMSE of **0.4851** for the stemmed data, and **0.5183** for the unstemmed data. To obtain a more robust estimate of model performance, we additionally applied 5-fold cross-validation. This produced a mean RMSE of **0.4867** $\pm$ **0.0026** for the stemmed dataset, and **0.5200** $\pm$ **0.0019** for the unstemmed version. These results confirm that stemming improves predictive accuracy, and establish a reliable baseline performance against which other feature combinations and models were evaluated.

[1] `https://www.kaggle.com/c/home-depot-product-search-relevance/`
[2] `https://www.kaggle.com/c/home-depot-product-search-relevance/data`
[3] `https://www.kaggle.com/wenxuanchen/sklearn-random-forest`

Regression models

Following the baseline evaluation, we compared several regression models to determine which performs best using the same feature set and preprocessing pipeline. All models were trained and evaluated using the same 80/20 dataset split and stemmed data to ensure consistency in comparison. We first re-evaluated the `RandomForestRegressor` with 15 estimators and a maximum depth of 6, wrapped in a `BaggingRegressor` with 45 base estimators and a subsample ratio of 0.1, replicating the baseline setup. Next, we tested a `GradientBoostingRegressor`, configured with the same number of estimators and depth. We then evaluated a `Support Vector Regressor (SVR)` using an RBF kernel, with regularization parameter $C = 1.0$ and $\epsilon = 0.2$, preceded by feature scaling using `StandardScaler`. Finally, we included a `KNeighborsRegressor` with $k = 5$ neighbors as a non-parametric baseline to gauge the performance of instance based methods in this context. Each model was also assessed using 5-fold cross-validation, with performance measured by the mean RMSE and an informal comparison of training times. The results of these evaluations are shown in the following table.

Model	RMSE	CV RMSE	Train Time (s)
Random Forest	0.4804	0.4830 ± 0.0026	0.33
Gradient Boosting	0.4820	0.4851 ± 0.0021	0.55
Support Vector Regressor	0.4850	0.4866 ± 0.0028	35.14
KNN Regressor	0.5200	0.5272 ± 0.0020	0.01

Table 1: Comparison of regression models based on RMSE, cross-validation performance, and training time.

Additional experiments were conducted with the KNN model using higher values for the number of neighbors (e.g., $k = 10$, $k = 15$), but these variations did not lead to any notable performance improvements. The SVR model yielded moderate predictive accuracy but incurred significantly higher training times compared to the tree based models. Based on this comparison, only the `RandomForestRegressor` and `GradientBoostingRegressor` were selected for further tuning through hyperparameter optimization.

Hyperparameter Optimization

Based on the initial model comparison, both the `RandomForestRegressor` and `GradientBoostingRegressor` were selected for further tuning via hyperparameter optimization. The goal was to identify configurations that could improve performance beyond the default settings used in the baseline and comparative evaluations. Hyperparameter tuning was performed using `RandomizedSearchCV` with 5-fold cross-validation on the training set. For the `RandomForestRegressor`, the randomized search explored between 50 and 200 estimators, maximum depths from 4 to 15, and values for `min_samples_split` and `min_samples_leaf` ranging from 2 to 10 and 1 to 10 respectively. For the `GradientBoostingRegressor`, a similar search range was applied for the number of estimators and tree depth, with additional tuning of the learning rate over a continuous uniform distribution between 0.01 and 0.3, along with the same ranges for split and leaf constraints.

For the random forest, the best configuration achieved a cross-validated RMSE of $\mathbf{0.4825 \pm 0.0021}$ with a training time of approximately **45.63 seconds**. The optimal parameters were: `n_estimators = 183`, `max_depth = 7`, `min_samples_split = 7`, and `min_samples_leaf = 2`.

The gradient boosting regressor achieved a slightly better result, with a cross-validated RMSE of $\mathbf{0.4820 \pm 0.0035}$ and a training time of **48.85 seconds**. Its best configuration included `n_estimators = 51`, `max_depth = 5`, `learning_rate = 0.1101`, `min_samples_split = 6`, and `min_samples_leaf = 6`.

These results indicate that both tree based models perform competitively, with gradient boosting achieving the lowest error overall while remaining computationally feasible.

To assess whether the performance difference between the Random Forest and Gradient Boosting models is statistically significant, we conducted a paired t-test on their fold wise RMSE values. The resulting p-value was **0.14**, indicating that the difference is not statistically significant at the conventional 0.05 level. This suggests that both models perform comparably in terms of error. However, the choice to proceed with the `GradientBoostingRegressor` remains valid: it achieved the lowest mean RMSE, showed strong performance on key semantic features, and is theoretically well suited for capturing fine-grained relevance patterns due to its sequential error-correcting approach. The selection thus reflects a principled preference based on both empirical results and model characteristics, even if the performance gains are modest.

Feature Engineering

With hyperparameter tuning complete, the next step focused on improving model performance through manual feature engineering. The tuned regressor served as a stable evaluation baseline, allowing us to assess the isolated impact of each engineered feature on model accuracy.
The objective of this phase was to introduce new features that could capture deeper semantic relationships between queries and product data beyond simple token overlaps. This included structured attributes such as brand, material, and color, as well as domain specific patterns like measurement units and bigrams. Additionally, multiple forms of textual similarity were incorporated, including Jaccard and fuzzy string matching. The full set of features considered is listed in the tabular below:

Feature	Description
`query_length`	Number of words in the query
`common_words`	Query token overlap with title and description
`brand_match`	Overlap with product brand from attributes
`tfidf_similarity`	Cosine similarity of query and product text
`query_has_number`	Binary indicator for numeric terms in query
`unit_match`	Match on measurement units (e.g., ft, inch)
`initial_term_match`	First query terms appear early in title
`material_match`	Match between query and product material
`color_match`	Match between query and product color
`jaccard`	Jaccard similarity with product title and description
`bigram_overlap`	Count of overlapping bigrams in title and description
`fuzzy`	Fuzzy matching score with product title and description

Table 2: Engineered features and their descriptions.

To evaluate the effectiveness of each individual feature, we trained a new model using only that single feature and measured its performance using the tuned `GradientBoostingRegressor`. This allowed us to isolate each feature's predictive contribution. The results are presented below, reporting RMSE on the test set, mean cross-validation RMSE with standard deviation, and training time.

Feature	RMSE	CV RMSE (mean)	Std. Dev.	Train Time (s)
TF-IDF similarity	0.5027	0.5101	0.0103	195.82
Initial term match	0.5140	0.5187	0.0152	120.50
Query length	0.5267	0.5310	0.0083	120.44
Jaccard	0.5082	0.5157	0.0144	128.78
Fuzzy	0.5197	0.5273	0.0117	148.22
Query has number	0.5311	0.5361	0.0093	119.74
Bigram overlap	0.5236	0.5294	0.0105	127.27
Color match	0.5216	0.5218	0.0029	125.53
Brand match	0.5320	0.5372	0.0080	120.93
Common words	0.5160	0.5236	0.0149	123.73
Unit match	0.5324	0.5373	0.0082	119.82
Material match	0.5289	0.5336	0.0090	120.32

Table 3: Evaluation of single feature models using a tuned `GradientBoostingRegressor` trained on **stemmed** data. Standard deviation refers to the cross-validation RMSE variability across folds.

This analysis highlights the relative value of each feature in isolation and provides guidance for selecting effective combinations in the full feature set. After evaluating the predictive power of individual features, we turned our attention to analyzing how much each contributed within the full model.

Inspect Feature Weights

To better understand which features contribute most significantly to the model's predictions, we analyzed the learned feature importances from our final model, a tuned `GradientBoostingRegressor`. Since this model belongs to the family of tree based estimators, we made use of the `feature_importances_` attribute to extract each feature's contribution to the overall prediction accuracy. This required disabling the `BaggingRegressor` wrapper in order to access the estimator directly.

Each feature name was stored in an ordered list and aligned with its corresponding importance score. We then sorted the features in descending order of importance to identify the most influential variables.
Figure (2) visualizes the sorted feature importances. The five most influential features in the final model were: `tfidf_similarity`, `query_length`, `initial_term_match`, `title_overlap` (part of the `common_words` feature), and `fuzzy_title` (part of the `fuzzy` feature). Their high importance values indicate that measures of token overlap and semantic similarity played a central role in determining search relevance. This aligns with our expectations, as such features capture both exact and approximate matches between search queries and product information, which are key drivers of relevance in this domain. These features outperform others due to their ability to model semantic and structural similarity at various linguistic levels, from token to vector space. This analysis confirms that fuzzy matching and distributed similarity metrics are especially effective in this domain.

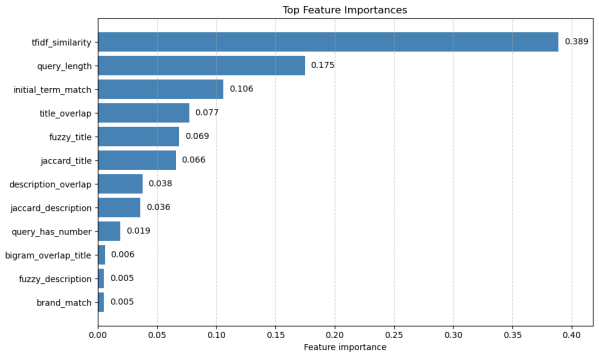

Figure 2: Bar plot showing the relative importance of features in the final model.

Matching Improvements

After evaluating each feature individually, we incrementally constructed feature sets by combining the highest performing features step by step. This allowed us to assess how different combinations interacted and whether they offered complementary value. Starting with the most predictive feature in isolation, we added one feature at a time, re-evaluating the model performance at each step. This bottom-up approach provided clear evidence of which features contributed the most to reducing RMSE when used in combination.

We observed that some combinations of features resulted in diminishing returns due to overlapping information content. For example, several features measuring textual similarity (such as fuzzy matching, bigram overlap, and Jaccard similarity) tended to capture similar aspects of the query-product relationship. As a result, adding all of them together did not always yield substantial improvements beyond what the strongest individual features already offered.

Figure (3) shows how the model's cross-validated RMSE changes with the number of features used. The plot confirms that relevance prediction performance generally improves as more features are included, especially when stemming is applied, though the gains plateau as overlapping or redundant features are added.

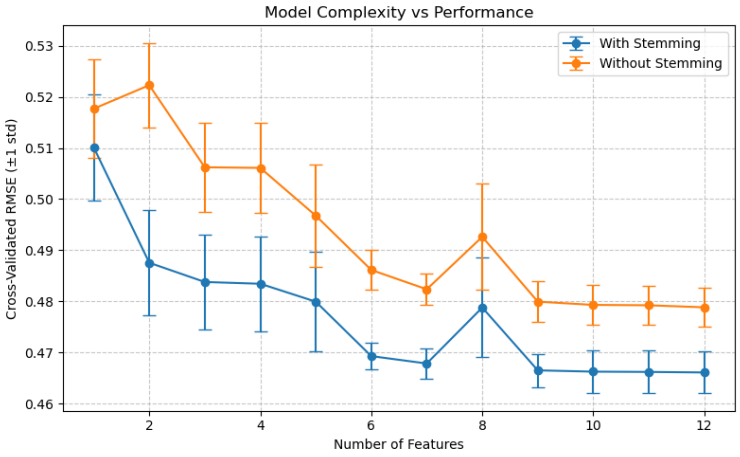

Figure 3: Relationship between the number of features and model performance (RMSE) with and without stemming.

Conclusion

The best performing feature set was selected manually and consisted of the following seven features: `query_length`, `initial_term_match`, `jaccard`, `common_words`, `color_match`, `fuzzy`, and `bigram_overlap`.

This combination provided a strong balance between capturing surface level and semantic similarities without introducing excessive redundancy. It included both simple syntactic cues (e.g., `query_length`, `initial_term_match`) and more advanced similarity metrics (e.g., `fuzzy`, `bigram_overlap`), contributing to more robust performance.

This configuration achieved a cross-validated RMSE of approximately **0.4679** with a standard deviation of **0.00298**, and a training time of roughly **196.5 seconds**. Although the RMSE continued to decrease slightly as more features were added, as shown in Figure (3). This trend likely reflects overfitting to the training data. The test set did not include the true relevance scores, so we were unable to directly evaluate generalization performance. Nevertheless, this behavior is expected for ensemble models like gradient boosting, where the inclusion of highly similar or overlapping features can lead to diminishing returns. The chosen seven-feature configuration therefore represents a practical trade off between model complexity and overfitting risk.

Interestingly, domain specific features such as `unit_match`, `brand_match`, and `material_match` contributed relatively little when evaluated in isolation. This may be because such attributes are inconsistently represented in the product metadata or may not appear frequently in the search queries themselves. Additionally, these attributes often lack standardization (e.g., unit formats or brand names), making them less reliable as matching signals compared to text based similarity metrics. Their low importance suggests that more robust preprocessing or structured feature extraction may be necessary to make better use of this domain knowledge.

Future work could explore the impact of automatic feature selection and dimensionality reduction techniques, which may further reduce overfitting risk. Additionally, testing on datasets with known test labels would help verify the generalization capability of the chosen feature set.