The effects of feature engineering on the accuracy of a regression model

Petridou Elena (s2029197)

May 2024

1 Introduction

1.1 Context

Kaggle is an online data science community which hosts learning material, datasets and competitions – all with the aim of helping students and practitioners hone their data science skill-set. The "Home Depot Product Search Relevance" competition hosted on Kaggle, tasks participants with creating a regression model predicting the search relevance of a product from the Home Depot catalogue, given a certain search term.

1.2 Problem Description

The relevance score between product and search-term pairs is given on a continuous scale from 1 (not relevant) to 3 (highly relevant). A portion of the search term - product pairs have been labelled with a relevance score by human reviewers, and the competition asks users to come up with a regression model which scores product - search term pairs as closely as possible to the labels assigned by the human reviewers. This is assessed by extracting the root mean squared error (RMSE) achieved by the model. Overall, since this is a regression task with available human-labelled data from which to form a benchmark comparison, it is a supervised regression task.

1.3 Task definition

The task involves the usual steps followed when producing a machine learning model, which will be outlined here with notes about the particularities of this problem.

Firstly, data exploration is tantamount to understanding the attributes present in the data, as well as their measurement scales. Subsequently, data pre-processing involving natural language processing techniques is essential for extracting numerical representations of the data that can be used by machine learning models.

Following this, we move onto feature extraction which is essentially turning attributes into a usable input measure for a machine learning model. This often means finding a meaningful number which describes the relationship between a search term and some aspect of a product. The aspect of a product does not have to be a single attribute – and part of feature engineering is choosing which attributes, construed together would make for the closest approximation of a product vis-a-vis a search term looking for that product.

Thereafter, model selection takes into account the type of task at hand, as well as the structure of the data (dimensionality, number of samples, presence of NaN values). In a limited-scope project such as this, time-efficiency, and adaptability are also important considerations. The most important heuristic determining which model to select, however, is which model minimises the error measurements of the final predictions – primarily the RMSE.

Once the model has been selected, hyperparameter optimisation ensures that the parameters on which the model learns minimise the error as much as possible.

Finally, assessment of the accuracy of the model via error statistics (in the case of regression, the RMSE), as well as inspecting the learned weights for each feature paves the way to an analysis of the model's performance and the soundness of the decisions made along the way.

1.4 Research Question

Two points of interest inform the research question for this paper.

Firstly, in Assignment 2, it was noted that features quantified using measures of semantic similarity between the search term and different aspects of a product were ranked overall higher on the list of feature importance than the word count features, but the shared words in the product title, and the search query were informative too.

Secondly, the Random Forest Regressor yielded the lowest RMSE, which dropped even lower when combining it with the ensemble technique of bagging. As such, in this paper I compare the performance of the standalone Random Forest regressor, and two ensemble techniques using Random Forest: bagging, and boosting.

Altogether, these findings motivated my choice to compare the effects of pre-processing and feature selection versus the effects of model selection on the error reduction in the model. The following research question thus arises: *How do data pre-processing and feature engineering impact the error in the regression model in the Home Depot Product Search Relevance task?*

2 Data

The training dataset is made up of 74067 unique product-search term pairs. The relevance of the product in terms of the search term has been rated on a continuous scale from 1 to 3 by human raters. The test set contains 92621 unique product-search term pairs, which are un-labelled.

There is a reasonable overlap of 12520 Product uid's between the train and test datasets, but the unique Product uid's in the test (55652) and train (42147) sets means that the risk of overfitting due to overlap is not high.

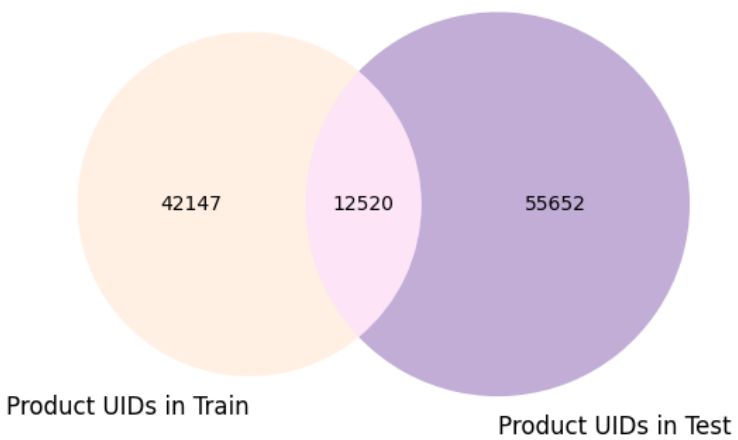

Figure 1: Venn Diagram of Products in the training VS Products in the test set

The two products occuring most often in the training set are the "Pressure-Treated Timber #2 Southern Yellow Pine (Common: 4 in. x 4 in. x 8 ft.; Actual: 3.56 in. x 3.56 in. x 96 in.)", (product uid: 101959), and the "Lithonia Lighting All Season 4 ft. 2-Light Grey T8 Strip Fluorescent Shop Light", (product uid: 102893), both of which appear in the training data 21 times.

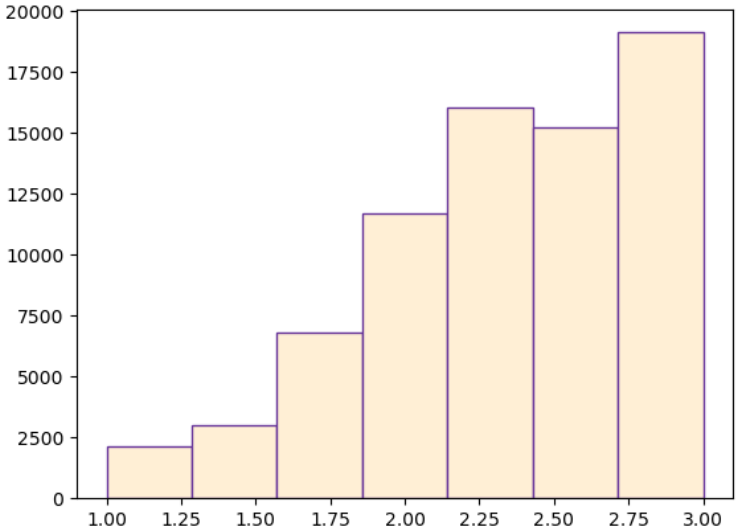

Figure 2: Histogram of Relevance Score distribution in Train dataset

The human-assigned relevance scores have a mean of 2.382 (to 3 decimals), a median value of 2.33, and a standard deviation of 0.534 (to 3 decimals). This shows that most product-search term pairs viewed by the annotators were quite relevant – which might mean that the trained model may over-rate certain items' relevance if they were not included in the test set – since the model will not be exposed to many non-relevant products, meaning the patterns that make items not relevant may be less likely to be 'learned'. Moreover, the small standard deviation shows that the product-search term pairs annotated were more or less similarly relevant – or at least more than expected from a standard normal distribution, which would have a standard deviation of 1.

In terms of attributes, the most listed product attributes, contained in the attributes.csv dataset, as well as the amount of products labelled with them are tabulated below. These informed the choice of matching features which were computed.

Table 1: Top attribute types in attributes.csv

Attribute Name	Count
MFG Brand Name	86250
Product Width (in.)	61137
Product Height (in.)	54698
Product Depth (in.)	53652
Product Weight (lb.)	45175
Color Family	40733
Material	30682
Color/Finish	28564
Assembled Height (in.)	18299
Assembled Width (in.)	18263
Assembled Depth (in.)	18198
Product Length (in.)	16705

Finally, the top-5 most occuring brand names (gathered under "MFG Brand Name") in the product attributes are: Unbranded, Hampton Bay, KOHLER, Everbilt, and Home Decorators Collection.

3 Methods

3.1 Data Pre-processing

Pre-processing choices were infomed by contributions by some of the top competitors to Simkievich's (2012) discussion, as well as Chen's (2012) flowchart visualising his successful team's process.

3.1.1 General Text Pre-processing

Firstly, I made every word in the dataset lowercase and removed the stop-words – words which carry no significant meaning in the English language. To do this, I used the NLTK stopwords dictionary. Next, I removed the terms in brackets from all fields. Following this, I also applied lemmatisation to the lower-cased words (without stopwords). Lemmatisation has a similar goal as stemming: namely, to ensure that words with the same meaning are grouped together regardless of the linguistic features (suffixes, for example) that make the strings nonidentical. According to several articles (Bhoi, 2022; Beri, 2021), lemmatisation usually yields higher accuracy since lemmas take into account the semantic meaning of words, whereas stems only group together words which look alike, hence my choice to extend pre-processing with lemmatisation.

Part of pre-processing was also preparing several attributes to be transformed into features. Informed by the top features discussed in the previous section, I wished to capture a relationship between the search term and the attributes of a product, to use as a feature, which was done in two ways:

3.1.2 Composite Strings

The first approach that I followed was combining the product title along with some of the top features in the same string, in order to then be able to compute a similarity measure between the search-term and this new composite string. For example, Assembled Measurement (DxWxH), Product Measurement (LxWxD), Product Measurement (DxWxH) were combined into single fields respectively – since these measurements occur individually per dimension in attributes.csv, but together in search terms. Then each measurement format was combined with the product title into a new variable.

3.1.3 Matching Isolated Attributes

The second approach was to isolate the product's attribute, and also recognise and extract any attributes in the search terms, and make them into a different variable. For example, I looked for colour lemmas in the lemmatised search terms, which if found, are then made into the 'color_in_search_term' attribute.

In all, the attributes addressed via both of these methods were: the product's colour (Color Family & Color/Finish), Assembled Measurement (DxWxH), Product Measurement (LxWxD), Product Measurement (DxWxH), brand name (MFG Brand Name), and the product's material (Material). This choice was informed by the results acquired at the end of the last assignment, whereby the product title, combined with the product's assembled measurements, as well as the product title combined with the product's measurements in the format (Depth x Width x Height), ranked third and fifth in the list of most predictive features, respectively.

3.1.4 Extracting "Clean" Product Names

Inspired by Harb, Khomenko, Gámez and Simkievich's (2015) unique method of isolating query-product name similarity, the last pre-processing approach I experimented with was trying to isolate a short and accurate product name. Essentially, what this achieved for a good portion of the products was shortening the majority of product title's from

something like: '*Safavieh Courtyard Natural/Brown 8 ft. 11 in. x 12 ft. Area Rug*' (product uid: 100078), to something like 'area rug', which is –presumably– a more realistic search term to expect from a user. For originality's sake, I did not follow the same steps as in the blog, but inspected the search terms in order to observe patterns on which to filter. First of all, all attributes (colour, material, measurements, etc.) were removed from the search terms – since those are already captured by the composite and matching features. Then, any digits or terms with brackets or hyphens were removed, as they are not likely to be part of the core description of an item. Finally, grammatical tagging using TextBlob was used to discard all parts of speech other than nouns, simple (non-comparative) adjectives, and base-verbs. The first two are the parts of speech most likely to be used in describing a product – for example: 'fluorescent light', while the last was kept to prevent false removal of words that are both noun and verb (e.g.: shower in the potential search term 'shower curtain').

3.2 Feature Extraction

Informed by the performance of features in Assignment 2, where it was observed that the semantic similarity quantified by spacy yielded features with (relatively) high predictive power, and significantly higher than the next best predictor, I have chosen to compute features based on further semantic similarity measures, in order to ascertain whether they could further reduce the RMSE of the model.

3.2.1 Pairwise vector similarity/distance

One method of computing semantic similarity is via vectorisation. First, every product attribute variable (both *matching* and *composite*), the bigrams and trigrams (two- and three-word groupings) in the *composite* attributes, as well as the search terms (*matching*, and whole) were transformed into vectors via tf-idf - which assigns a 'score' to each word based on its relevance in the corpus (the whole string). Thereafter, the pairwise distance between vectors in an inner-product space can be calculated, via different metrics. Two of the vector-distance measures calculated were the Euclidean distance and Cosine similarity.

For the *composite strings*, described above, the pairwise Cosine similarity and Euclidean distance were calculated between each composite string's vector representation (e.g.: product title + colour) and the lemmatised search term's vector representation. The resulting features are tabulated in **Table 2** (seen in page 5).

The same two similarity metrics were also computed pairwise between the (*matching isolated attributes*). In other words, between each individual attribute in the search term (e.g.: colour in the search term) and each individual attribute of a product (e.g.: product's colour). These features are presented in **Table 3**.

Table 3: Features, the variable pairs from which they were created, and the metric calculated for each – Pairwise vector distances for isolated attribute matching

cosine_color_matching	color_lemma	colour_in_search_term	Cosine similarity between the product's colour (as a word) and the colours in the search term via tf-idf vectorisation
cosine_material_matching	material_lemma	material_in_search_term	Cosine similarity between the product's material (as a word) and the material in the search term via tf-idf vectorisation
cosine_brand_matching	brand_lemma	brand_in_search_term	Cosine similarity between the product's brand (as words) and the brand in the search term via tf-idf vectorisation
cosine_assembled_measurement_matched	Assembled Measurement (DxWxH)	measurement_in_search_term	Cosine similarity between the product's assembled measurements and the measurements in the search term via tf-idf vectorisation
cosine_measurement_LWD_matched	Product Measurement (LxWxD)	measurement_in_search_term	Cosine similarity between the product's measurements in Length x Width x Depth and the measurements in the search term via tf-idf vectorisation
cosine_measurement_DWH_matched	Product Measurement (DxWxH)	measurement_in_search_term	Cosine similarity between the product's measurements in Depth x Width x Height and the measurements in the search term via tf-idf vectorisation
euclidean_color_matching	color_lemma	color_in_search_term	Euclidean distance between the product's colour (as a word) and the colours in the search term (as a word) via tf-idf vectorisation
euclidean_material_matching	material_lemma	material_in_search_term	Euclidean distance between the product's material (as a word) and the material in the search term (as a word) via tf-idf vectorisation
euclidean_brand_matching	brand_lemma	brand_in_search_term	Euclidean distance between the product's brand (as words) and the brand in the search term (as words) via tf-idf vectorisation
euclidean_assembled_measurement_matched	Assembled Measurement (DxWxH)	measurement_in_search_term	Euclidean distance between the product's assembled measurements and the measurements in the search term via tf-idf vectorisation
euclidean_measurement_LWD_matched	Product Measurement (LxWxD)	measurement_in_search_term	Euclidean distance between the product's measurements in Length x Width x Depth and the measurements in the search term via tf-idf vectorisation
euclidean_measurement_DWH_matched	Product Measurement (DxWxH)	measurement_in_search_term	Euclidean distance between the product's measurements Depth x Width x Height and the measurements in the search term via tf-idf vectorisation

Another method of quantifying similarity between strings is via count-based distance metrics. The Jaccard distance is such a metric – it is made up of the proportion of shared words or characters, divided by the total words or characters in a pair of strings. Both the proportion of words and characters were used to quantify the distance between the *composite string* product representations and the search terms, as well as the distance between the *matched isolated attributes* of the products and the *matched isolated attributes* of the search terms. This resulted in the features shown in **Table 4**.

Table 4: Features, the variable pairs from which they were created, and the metric calculated for each – Jaccard proportion of shared words or characters for all attributes

jaccard_product_title_lemma	product_title_lemma	search_term_lemma	Jaccard similarity (proportion of shared characters) between product title and search term
jaccard_product_description_lemma	product_description_lemma	search_term_lemma	Jaccard similarity (proportion of shared characters) between product description and search term
jaccard_title_and_assembled_measurement_lemma	title_and_assembled_measurement_lemma	search_term_lemma	Jaccard similarity (proportion of shared characters) between title and assembled measurement and search term
jaccard_title_and_measurement_LWD_lemma	title_and_measurement_LWD_lemma	search_term_lemma	Jaccard similarity (proportion of shared characters) between title and measurement in Length x Width x Depth and search term
jaccard_title_and_measurement_DWH_lemma	title_and_measurement_DWH_lemma	search_term_lemma	Jaccard similarity (proportion of shared characters) between title and measurement in Depth x Width x Height and search term
jaccard_title_and_color_fam_lemma	title_and_color_fam_lemma	search_term_lemma	Jaccard similarity (proportion of shared characters) between title and colour family and search term
jaccard_title_and_color_lemma	title_and_color_lemma	search_term_lemma	Jaccard similarity (proportion of shared characters) between title and colour and search term
jaccard_title_and_brand_lemma	title_and_brand_lemma	search_term_lemma	Jaccard similarity (proportion of shared characters) between title and brand and search term
jaccard_title_and_material_lemma	title_and_material_lemma	search_term_lemma	Jaccard similarity (proportion of shared characters) between title and material and search term
jaccard_cleaned_title	clean_product_title	search_term_lemma	Jaccard similarity (proportion of shared characters) between the product's "cleaned" title and the search term
jaccard_color_matching	color_lemma	color_in_search_term	Jaccard similarity (proportion of shared characters) between the product's colour and the colour in the search term
jaccard_material_matching	material_lemma	material_in_search_term	Jaccard similarity (proportion of shared characters) between the product's material and the material in the search term
jaccard_brand_matching	brand_lemma	material_in_search_term	Jaccard similarity (proportion of shared characters) between the product's brand and the brand in the search term
jaccard_assembled_measurement_matched	Assembled Measurement (DxWxH)	measurement_in_search_term	Jaccard similarity (proportion of shared digits) between the product's assembled measurements and the measurements in the search term
jaccard_measurement_LWD_matched	Product Measurement (LxWxD)	measurement_in_search_term	Jaccard similarity (proportion of shared digits) between the product's measurements in Length x Width x Depth and the measurements in the search term
jaccard_measurement_DWH_matched	Product Measurement (DxWxH)	measurement_in_search_term	Jaccard similarity (proportion of shared digits) between the product's measurements in Depth x Width x Height and the measurements in the search term
jaccard_word_product_title_lemma	product_title_lemma	search_term_lemma	Jaccard similarity (proportion of shared words) between the product's title and the search term
jaccard_word_product_description_lemma	product_description_lemma	search_term_lemma	Jaccard similarity (proportion of shared words) between the product's description and the search term
jaccard_word_title_and_assembled_measurement_lemma	title_and_assembled_measurement_lemma	search_term_lemma	Jaccard similarity (proportion of shared words) between the product's title and assembled measurement and the search term
jaccard_word_title_and_measurement_LWD_lemma	title_and_measurement_LWD_lemma	search_term_lemma	Jaccard similarity (proportion of shared words) between the product's title and measurements in Length x Width x Depth and the search term
jaccard_word_title_and_measurement_DWH_lemma	title_and_measurement_DWH_lemma	search_term_lemma	Jaccard similarity (proportion of shared words) between the product's title and measurements in Depth x Width x Height and the search term
jaccard_word_title_and_color_fam_lemma	title_and_color_fam_lemma	search_term_lemma	Jaccard similarity (proportion of shared words) between the product's title colour family and the search term
jaccard_word_title_and_color_lemma	title_and_color_lemma	search_term_lemma	Jaccard similarity (proportion of shared words) between the product's title and colour and the search term
jaccard_word_title_and_brand_lemma	title_and_brand_lemma	search_term_lemma	Jaccard similarity (proportion of shared words) between the product's title and brand and the search term
jaccard_word_title_and_material_lemma	title_and_material_lemma	search_term_lemma	Jaccard similarity (proportion of shared words) between the product's title and material and the search term
jaccard_word_cleaned_title	clean_product_title	search_term_lemma	Jaccard similarity (proportion of shared words) between the product's "cleaned" title and the search term
jaccard_word_color_matching	color_lemma	color_in_search_term	Jaccard similarity (proportion of shared words) between the product's colour and the colour in the search term
jaccard_word_material_matching	material_lemma	material_in_search_term	Jaccard similarity (proportion of shared words) between the product's material and the material in the search term
jaccard_word_brand_matching	brand_lemma	brand_in_search_term	Jaccard similarity (proportion of shared words) between the product's brand and the brand in the search term
jaccard_word_assembled_measurement_matched	Assembled Measurement (DxWxH)	measurement_in_search_term	Jaccard similarity (proportion of shared numbers) between the product's assembled measurements and the measurements in the search term
jaccard_word_measurement_LWD_matched	Product Measurement (LxWxD)	measurement_in_search_term	Jaccard similarity (proportion of shared numbers) between the product's measurements in Length x Width x Depth and the measurements in the search term
jaccard_word_measurement_DWH_matched	Product Measurement (DxWxH)	measurement_in_search_term	Jaccard similarity (proportion of shared numbers) between the product's measurements in Depth x Width x Height and the measurements in the search term

Table 2: Features, the variable pairs from which they were created, and the metric calculated for each – Pairwise vector distances for composite strings

Feature variable	Attribute variable 1	Attribute variable 2	Feature Metric and Explanation
cosine_product_title_lemma	product_title_lemma	search_term_lemma	Cosine similarity between the words in the product's title and the words in the search term via tf-idf vectorisation
cosine_product_description_lemma	product_description_lemma	search_term_lemma	Cosine similarity between the words in the product's description and the words in the search term via tf-idf vectorisation
cosine_title_and_assembled_measurement_lemma	title_and_assembled_measurement_lemma	search_term_lemma	Cosine similarity between the words in the product's title and assembled measurement and the search term via tf-idf vectorisation
cosine_title_and_measurement_LWD_lemma	title_and_measurement_LWD_lemma	search_term_lemma	Cosine similarity between the words in the product's title and measurement in Length x Width x Depth and the search term via tf-idf vectorisation
cosine_title_and_measurement_DWH_lemma	title_and_measurement_DWH_lemma	search_term_lemma	Cosine similarity between the words in the product's title and measurement in Depth x Width x Height and the search term via tf-idf vectorisation
cosine_title_and_color_fam_lemma	title_and_color_fam_lemma	search_term_lemma	Cosine similarity between the words in the product's title and colour family and the search term via tf-idf vectorisation
cosine_title_and_color_lemma	title_and_color_lemma	search_term_lemma	Cosine similarity between the words in the product's title and colour and the search term via tf-idf vectorisation
cosine_title_and_brand_lemma	title_and_brand_lemma	search_term_lemma	Cosine similarity between the words in the product's title and brand and the search term via tf-idf vectorisation
cosine_title_and_material_lemma	title_and_material_lemma	search_term_lemma	Cosine similarity between the words in the product's title and material and the search term via tf-idf vectorisation
cosine_cleaned_title	clean_product_title	search_term_lemma	Cosine similarity between the words in the "clean" product title and the words in the search term via tf-idf
euclidean_product_title_lemma	product_title_lemma	search_term_lemma	Euclidean distance between the words in the product's title and the words in the search term via tf-idf vectorisation
euclidean_product_description_lemma	product_description_lemma	search_term_lemma	Euclidean distance between the words in the product's description and the words in the search term via tf-idf vectorisation
euclidean_title_and_assembled_measurement_lemma	title_and_assembled_measurement_lemma	search_term_lemma	Euclidean distance between the words in the product's title and assembled measurement and the words in the search term via tf-idf vectorisation
euclidean_title_and_measurement_LWD_lemma	title_and_measurement_LWD_lemma	search_term_lemma	Euclidean distance between the words in the product's title and measurement in Length x Width x Depth and the words in the search term via tf-idf vectorisation
euclidean_title_and_measurement_DWH_lemma	title_and_measurement_DWH_lemma	search_term_lemma	Euclidean distance between the words in the product's title and measurement in Depth x Width x Height and the words in the search term via tf-idf vectorisation
euclidean_title_and_color_fam_lemma	title_and_color_fam_lemma	search_term_lemma	Euclidean distance between the words in the product's title and colour family and the words in the search term via tf-idf vectorisation
euclidean_title_and_color_lemma	title_and_color_lemma	search_term_lemma	Euclidean distance between the words in the product's title and colour and the words in the search term via tf-idf vectorisation
euclidean_title_and_brand_lemma	title_and_brand_lemma	search_term_lemma	Euclidean distance between the words in the product's title and brand and the words in the search term via tf-idf vectorisation
euclidean_title_and_material_lemma	title_and_material_lemma	search_term_lemma	Euclidean distance between the words in the product's title and material and the words in the search term via tf-idf vectorisation
euclidean_cleaned_title	clean_product_title	search_term_lemma	Euclidean distance between the words in the "clean" product title and the words in the search term via tf-idf vectorisation
cosine_product_title_lemma_bigrams	product_title_lemma	search_term_lemma	Cosine similarity between the bigrams in the product title and the words in the search term, vectorised via tf-idf
cosine_product_description_lemma_bigrams	product_description_lemma	search_term_lemma	Cosine similarity between the bigrams in the product description and the words in the search term, vectorised via tf-idf
cosine_title_and_assembled_measurement_lemma_bigrams	title_and_assembled_measurement_lemma	search_term_lemma	Cosine similarity between the bigrams in the product title and assembled measurements and the words in the search term, vectorised via tf-idf
cosine_title_and_measurement_LWD_lemma_bigrams	title_and_measurement_LWD_lemma	search_term_lemma	Cosine similarity between the bigrams in the product title and measurements in Length x Width x Depth and the words in the search term, vectorised via tf-idf
cosine_title_and_measurement_DWH_lemma_bigrams	title_and_measurement_DWH_lemma	search_term_lemma	Cosine similarity between the bigrams in the product title and measurements in Depth x Width x Height and the words in the search term, vectorised via tf-idf
cosine_title_and_color_fam_lemma_bigrams	title_and_color_fam_lemma	search_term_lemma	Cosine similarity between the bigrams in the product title and colour family and the words in the search term, vectorised via tf-idf
cosine_title_and_color_lemma_bigrams	title_and_color_lemma	search_term_lemma	Cosine similarity between the bigrams in the product title and colour and the words in the search term, vectorised via tf-idf
cosine_title_and_brand_lemma_bigrams	title_and_brand_lemma	search_term_lemma	Cosine similarity between the bigrams in the product title and brand and the words in the search term, vectorised via tf-idf
cosine_title_and_material_lemma_bigrams	title_and_material_lemma	search_term_lemma	Cosine similarity between the bigrams in the product title and material and the words in the search term, vectorised via tf-idf
cosine_product_title_lemma_trigrams	product_title_lemma	search_term_lemma	Cosine similarity between the trigrams in the product title and the words in the search term, vectorised via tf-idf
cosine_product_description_lemma_trigrams	product_description_lemma	search_term_lemma	Cosine similarity between the trigrams in the product description and the words in the search term, vectorised via tf-idf
cosine_title_and_assembled_measurement_lemma_trigrams	title_and_assembled_measurement_lemma	search_term_lemma	Cosine similarity between the trigrams in the product title and assembled measurements and the words in the search term, vectorised via tf-idf
cosine_title_and_measurement_LWD_lemma_trigrams	title_and_measurement_LWD_lemma	search_term_lemma	Cosine similarity between the trigrams in the product title and measurements in Length x Width x Depth and the words in the search term, vectorised via tf-idf
cosine_title_and_measurement_DWH_lemma_trigrams	title_and_measurement_DWH_lemma	search_term_lemma	Cosine similarity between the trigrams in the product title and measurements in Depth x Width x Height and the words in the search term, vectorised via tf-idf
cosine_title_and_color_fam_lemma_trigrams	title_and_color_fam_lemma	search_term_lemma	Cosine similarity between the trigrams in the product title and colour family and the words in the search term, vectorised via tf-idf
cosine_title_and_color_lemma_trigrams	title_and_color_lemma	search_term_lemma	Cosine similarity between the trigrams in the product title and colour and the words in the search term, vectorised via tf-idf
cosine_title_and_brand_lemma_trigrams	title_and_brand_lemma	search_term_lemma	Cosine similarity between the trigrams in the product title and brand and the words in the search term, vectorised via tf-idf
cosine_title_and_material_lemma_trigrams	title_and_material_lemma	search_term_lemma	Cosine similarity between the trigrams in the product title and material and the words in the search term, vectorised via tf-idf
euclidean_product_title_lemma_bigrams	product_title_lemma	search_term_lemma	Euclidean distance between the bigrams in the product title and the words in the search term, vectorised via tf-idf
euclidean_product_description_lemma_bigrams	product_description_lemma	search_term_lemma	Euclidean distance between the bigrams in the product description and the words in the search term, vectorised via tf-idf
euclidean_title_and_assembled_measurement_lemma_bigrams	title_and_assembled_measurement_lemma	search_term_lemma	Euclidean distance between the bigrams in the product title and assembled measurements and the words in the search term, vectorised via tf-idf
euclidean_title_and_measurement_LWD_lemma_bigrams	title_and_measurement_LWD_lemma	search_term_lemma	Euclidean distance between the bigrams in the product title and measurements in Length x Width x Depth and the words in the search term, vectorised via tf-idf
euclidean_title_and_measurement_DWH_lemma_bigrams	title_and_measurement_DWH_lemma	search_term_lemma	Euclidean distance between the bigrams in the product title and measurements in Depth x Width x Height and the words in the search term, vectorised via tf-idf
euclidean_title_and_color_fam_lemma_bigrams	title_and_color_fam_lemma	search_term_lemma	Euclidean distance between the bigrams in the product title and colour family and the words in the search term, vectorised via tf-idf
euclidean_title_and_color_lemma_bigrams	title_and_color_lemma	search_term_lemma	Euclidean distance between the bigrams in the product title and colour and the words in the search term, vectorised via tf-idf
euclidean_title_and_brand_lemma_bigrams	title_and_brand_lemma	search_term_lemma	Euclidean distance between the bigrams in the product title and brand and the words in the search term, vectorised via tf-idf
euclidean_title_and_material_lemma_bigrams	title_and_material_lemma	search_term_lemma	Euclidean distance between the bigrams in the product title and material and the words in the search term, vectorised via tf-idf
euclidean_product_title_lemma_trigrams	product_title_lemma	search_term_lemma	Euclidean distance between the bigrams in the product title and the words in the search term, vectorised via tf-idf
euclidean_product_description_lemma_trigrams	product_description_lemma	search_term_lemma	Euclidean distance between the bigrams in the product description and the words in the search term, vectorised via tf-idf
euclidean_title_and_assembled_measurement_lemma_trigrams	title_and_assembled_measurement_lemma	search_term_lemma	Euclidean distance between the bigrams in the product title and assembled measurements and the words in the search term, vectorised via tf-idf
euclidean_title_and_measurement_LWD_lemma_trigrams	title_and_measurement_LWD_lemma	search_term_lemma	Euclidean distance between the bigrams in the product title and measurements in Length x Width x Depth and the words in the search term, vectorised via tf-idf
euclidean_title_and_measurement_DWH_lemma_trigrams	title_and_measurement_DWH_lemma	search_term_lemma	Euclidean distance between the bigrams in the product title and measurements in Depth x Width x Height and the words in the search term, vectorised via tf-idf
euclidean_title_and_color_fam_lemma_trigrams	title_and_color_fam_lemma	search_term_lemma	Euclidean distance between the bigrams in the product title and colour family and the words in the search term, vectorised via tf-idf
euclidean_title_and_color_lemma_trigrams	title_and_color_lemma	search_term_lemma	Euclidean distance between the bigrams in the product title and colour and the words in the search term, vectorised via tf-idf
euclidean_title_and_brand_lemma_trigrams	title_and_brand_lemma	search_term_lemma	Euclidean distance between the bigrams in the product title and brand and the words in the search term, vectorised via tf-idf
euclidean_title_and_material_lemma_trigrams	title_and_material_lemma	search_term_lemma	Euclidean distance between the bigrams in the product title and material and the words in the search term, vectorised via tf-idf

A notable departure from the previous assignment, as well as the submitted proposal was the removal of the Spacy distance features, as the similarity measure used in the Spacy pipeline is also the Cosine similarity. This means that these features are highly correlated, and likely to decrease the model's accuracy. This is also confirmed in the results table included in the Discussion section. Manually computing the Cosine similarity after pre-processing was preferrable in the context of this assignment, as Spacy was very time-consuming, and less customisable.

After all the steps described above, a total of 99 features were calculated – excluding the Spacy features which were removed. With them, the feature-base increases to 115.

Feature normalisation via Sci-kit learn's Standard Scaler was also applied, according to the discussion in Lecture (Van Stein, 2024, p. 15) of feature normalisation via the Standard Scaler in Scikit-learn was followed. Since the feature set contains both semantic similarity measures, as well as count measures (shared ngrams, shared words, length of query), there are multiple different scales at play. The similarity measures vary from each other in their scales, but are usually bounded from above and from below, sometimes by negative values. They also tend to be decimal values. On the other hand, counts are only bounded from below by 0, and are indices. Thus, standardising the measurement scales should ensure that features are accurately scored. In the Results section, we see that standardisation did not make much of a difference in RMSE reduction – but this could also be due to the fact that tree regressors can usually handle non-normalised data well (Data Science Stack Exchange, 2017)

3.3 Model Selection and Hyperparameter Optimisation

In the previous assignment, the Bagging Regressor – an aggregator ensemble model (Singh, 2023) – in combination with a Random Forest Regressor achieved the lowest RMSE, while the standalone Random Forest was still very close. As such, Random Forest was used in this project as well, with the addition of another ensemble method: gradient boosting, using the XG-Boost library. The results for each model and each intervention in terms of feature selection and hyperparameter optimisation are reported in the Results section.

3.4 Evaluation Procedures

The error metric used to analyse the models was the Root Mean Squared Error (RMSE), as was used to score submissions in the original competition. To do this, the training dataset was split into a training and validation set with a proportion of 80:20. The model 'learned' on the training data and was then used to predict y-values on the validation data in order to compute the RMSE between the raters' relevance scores and the model's produced relevance scores.

4 Results

4.1 Comparison of Models

The following table plots the RMSE of different models, as well as different interventions. A slight drop in RMSE over Assignment 2 was achieved simply by including all new computed features. A steeper decrease is further observed when compared to the baseline solution we were working with in Assignment 2 (Chen, 2016).

Table 5: Comparison of different interventions and regression models

	Random Forest Regressor	**Random Forest in Bagging**	**XGBoost Random Forest Regressor**
Baseline Features (Chen)	0.495896031815335	0.4953635564453467	0.4957902167409902
New pre-processing steps and new features with standardisation	0.48474158246632254*	0.4884180433759997*	0.4853366278361741**
Only isolated attribute matching features	-***	-***	0.48563165153353566**
Only composite string features	0.45879239500159497*	0.4583859733941143*	0.45802250401198324**
After hyperparameter optimisation on only composite string features	0.4615434455973259*		0.4872323041876082**

*After dropping features which contained NaN values
**Including features with NaN values, as XG-Boost can handle these internally.
***Too many NaN values to drop, cannot use these models.

The composite string features and matching features appear to be highly correlated (as seen by the correlation plot below), which may explain the drop in RMSE when only one set is kept – since overly correlated features can create bias (Ashraf, 2023). Indeed, this makes sense since the matching and composite string features are two ways of capturing the same underlying phenomenon: the similarity of a product's attribute (e.g.: colour) and an attribute (e.g.: a colour) in a search term.

4.2 Feature Correlations

Since the drop in RMSE is so dramatic and only observed when dropping one kind of features, and not the other one, this raised overfitting concerns – despite the fact that usually XG-Boost handles correlated features better than other models (Carey, 2021). One reason for this could be that if too many of the composite strings were missing enough product attributes, we could end up with many repetitions of the semantic similarity of just the title and search term (since the attribute was not present in order to be concatenated into the composite string). Indeed, when plotting the correlation matrix between the composite string features, these are also very highly correlated: As such, we cannot trust this conspicuously low RMSE and must investigate. This example is very instructive, as discussions in the Kaggle forum for the Home Depot competition have indicated that the more features added to this model the better, however,

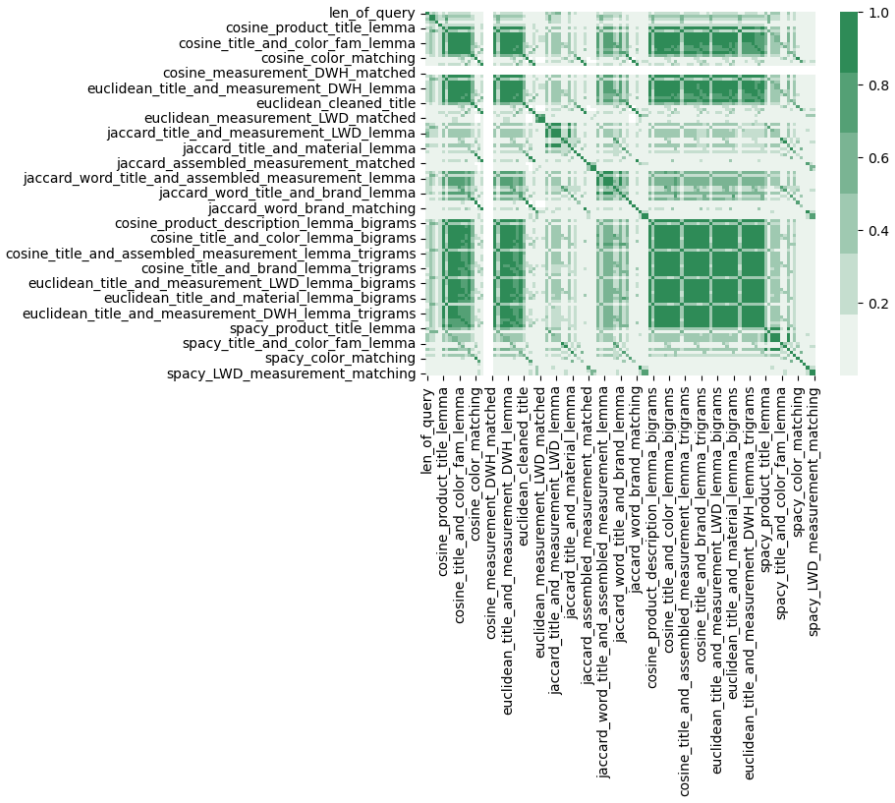

Figure 3: Correlations between all composite-string features

without careful selection of which features to calculate and in which way, we are likely to end up with an over-fitted model.

4.3 Final Approach and Best Model

Combining all of this, we reach the final approach, accounting for all confounding variables. First of all, the correlation between all independent variables (the feature scores) is calculated, and the features with the highest correlations are dropped *before* training the model. Subsequently, a random search is performed to tune the best hyperparameters for each dataset. The model is then fitted to the training data, and the following results arise, which are presented in Table 6. Puzzlingly, the default parameters for XG-Boost perform better than those found via hyperparameter optimisation, but since the calculation of the RMSE is on 'unseen' data – the validation set – this is perhaps not so worrying. On a more extensive project, this would of course be investigated further. However, since the focus of this assignment was on the effects of pre-processing and feature extraction, it is beyond the scope. Only the most significantly different results are plotted in Table 6; the rest are available in Code Blocks 40 and 41 of the submitted notebook.

Table 6: Comparison of different interventions and regression models

	Random Forest Regressor	**Random Forest in Bagging**	**XGBoost Random Forest Regressor**
Baseline Features (Chen)	0.495896031815335	0.4953635564453467	0.4957902167409902
Only composite string features without highly correlated features	-	-	**0.4849875655716115***
Only composite string features without highly correlated features, after hyperparameter optimisation	-	-	0.49638260945789586*

*After dropping features which contained NaN values

The features used in the best performing model (those not dropped due to high correlations) are highlighted in bold in Tables 3, 4, and 5. The highest correlations between the remaining features are shown below, and appear to be acceptably low:

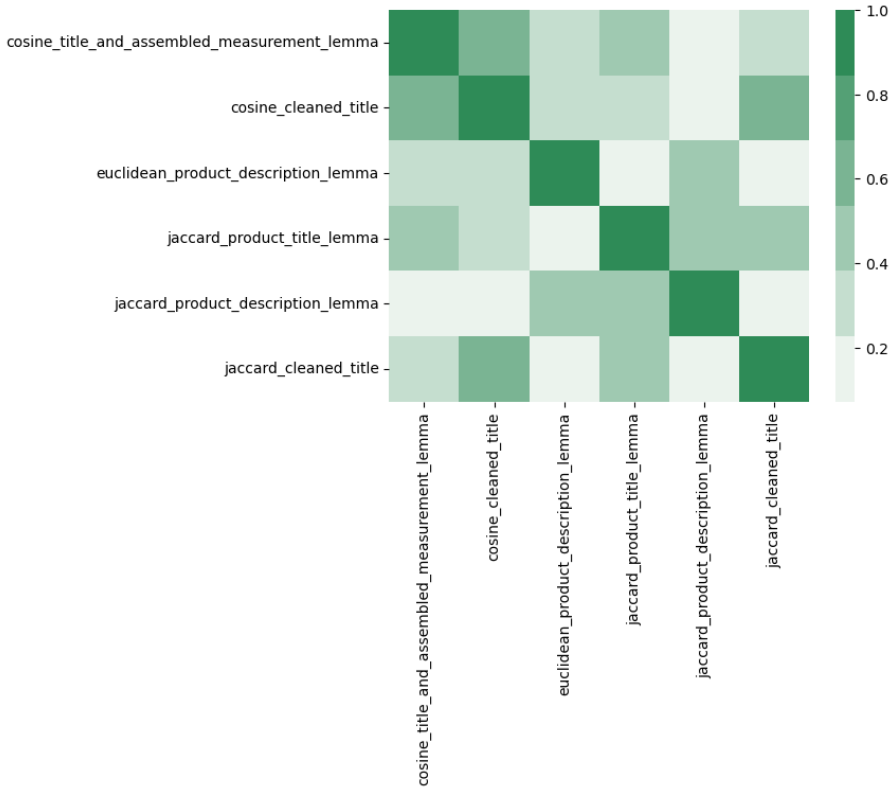

Figure 4: Correlations of features in final model

Note: The diagonal is the self-correlation of features, and can be disregarded

4.4 Top Predictive Features

The following graph shows the top-10 features used by the best performing model (XG-Boost regression with X feature set).

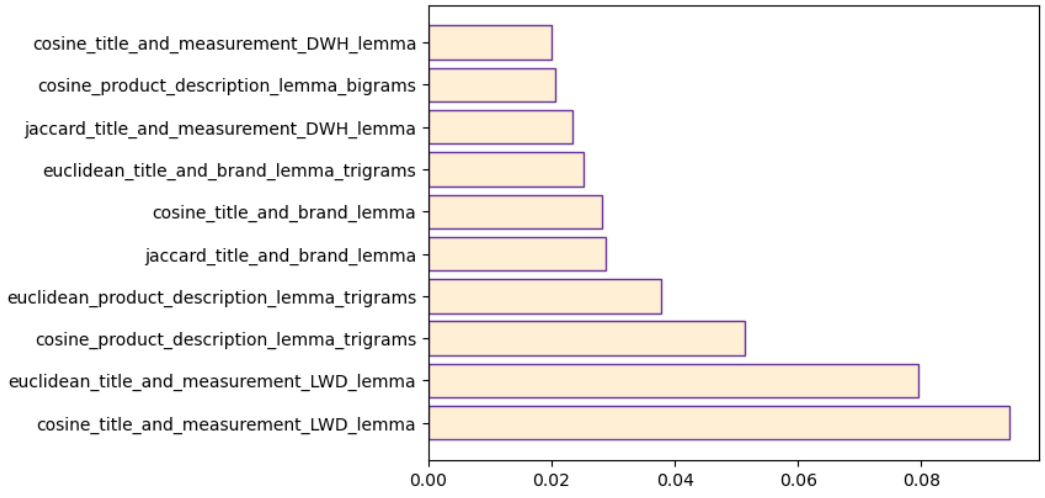

Figure 5: Top 10 features for the best performing model (XG-Boost with composite string features without NaN values)

It appears that the semantic similarity between the vectors of the product's title and measurements and the vector of the search term, in terms of words or trigrams are the most predictive features. From what we know about the data, this appears to be a reasonable outcome, since in Code Block 12, a query which attempts to extract measurements in the search term, finds a total of 1043 such entries in the training data. This means that a relatively large amount of search terms contain measurements, meaning that it is not unexpected that this is an informative feature.

5 Conclusion

In summation, to answer the research question at hand: *How do data pre-processing and feature engineering impact the error in the regression model in the Home Depot Product Search Relevance task?*. The findings in this assignment certainly point to the fact that data pre-processing and feature engineering majorly affect the error in the model. On the one hand, extracting more features and experimenting with which ones reduce the RMSE the most resulted in a better predictive model than the baseline, as well as the one presented in Assignment 2. On the other hand, the surprisingly low RMSE which appeared to originate from spurious relationships between the predictive features, showed that overfitting due to a lack of understanding of the data and the underlying structure of features extracted are very

real risks. Indeed, had the set of the features which reduced the RMSE the most been 'blindly' selected and subsequent model selection and hyperparameter used without interrogation, the bias in the model in the form of feature correlations would have gone unnoticed. This shows that, yes, data pre-processing and feature engineering influence the accuracy of the model dramatically, for good or for worse.

A wider implication, therefore, for any machine learning project is that careful analysis of the data via inspection and descriptive statistics, as well as reasonable familiarity with the domain of the task – in this case natural language processing or even more specifically, the search function on online shopping websites – is tantamount to a good result. Thereafter, careful consideration of feature-engineering is tantamount to producing an interpretable and accurate result.

Even though certain choices may produce lower error estimates, and therefore seem like the correct interventions at first sight, it is still important to be able to interpret the results and theorise why certain metrics have that value, most crucially to avoid developing a biased model which produces a lower error due to overfitting. Overall, this project has (personally) instilled the idea that deep understanding of every design decision in a model (though sometimes difficult due to constraints or lack of domain knowledge) is absolutely essential.

Importantly, since feature engineering is one of the less 'opaque' steps in building a machine learning project – especially compared to understanding the algorithms and their tuning procedures –, it is perhaps the most important one to focus on in most projects. Moreover, many of the most-used algorithms come ready to use out-of-the-box and can usually be tuned via semi-automated processes, meaning that focusing on understanding the data as well as ensuring that the extracted features are as suitable as possible for the desired result, is a better use of time in projects of limited scope and perhaps limited time-limits.

6 References

Ashraf, A. (2023, September 22). Correlation in machine learning - all you need to know. Medium. https://medium.com/@abdallahashraf90x/all-you-need-to-know-about-correlation-for-machine-learning-e249fec292e9

Beri, A. (2021, January 27). Stemming vs lemmatization. Medium. https://towardsdatascience.com/stemming-vs-lemmatization-2daddabcb221

Bhoi, N. (2022, December 14). Stemming vs lemmatization in NLP. Medium. https://nirajbhoi.medium.com/stemming-vs-lemmatization-in-nlp-efc280d4e845

Carey, V. (2021, February 8). How to fix feature bias. Medium. https://towardsdatascience.com/how-to-fix-feature-bias-9e47abccb942

Chen, C. (2012) kaggle-HomeDepot github repo. Github. https://github.com/ChenglongChen/kaggle-HomeDepot/blob/master/Fig/FlowChart.jpg

Chen, W. X. (username 'Moomin') (2016, January 21). Sklearn_random_forest. Kaggle. https://www.kaggle.com/code/wenxuanche random-forest/script

Harb, M., Khomenko, R., Gámez, S., & Simkievich, A. (2016, March 1). Crowdflower Winners' interview: 3rd Place, team quartet. No Free Hunch (Kaggle Blog). Accessed via Wayback Machine: https://web.archive.org/web/20190914022539/http://blog.kaggle.com/2015/07/22/crowdflower-winners-interview-3rd-place-team-quartet/

Simkievich, A. (2012). Home Depot Product Search Relevance. Kaggle. https://www.kaggle.com/competitions/home-depot-product-search-relevance/discussion/20427

Singh, A. (2023, November 22). A comprehensive guide to ensemble learning (with python codes). Analytics Vidhya. https://www.analyticsvidhya.com/blog/2018/06/comprehensive-guide-for-ensemble-models/

Soerendip (username). (2017, January 10). Would you recommend feature normalization when using boosting trees?. Data Science Stack Exchange. https://datascience.stackexchange.com/questions/16225/would-you-recommend-feature-normalization-when-using-boosting-trees

Van Stein, N. (2024). "Lecture 10: Preprocessing", Data Science, Leiden University, Class lecture. https://brightspace.universiteitleiden.nl/d2l/le/lessons/240344/topics/2889784