STATS202 Project

Duorui Xu, Shichao Chen, Hongrui Wang

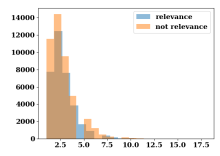
Kaggle: D. Xu S. Chen H. Wang

August 2024

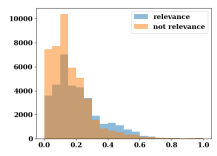
1 Data Analysis

Before fitting the training data to models, data pre-processing was done to analysis the basic patterns of features in the dataset. Several critical steps were involved including basic pattern identification and relation recognition. From the pattern of the data, basic strategy on how to transform the data were made and with second step, the model may be able to eliminate certain dimension. Both processes were conducted with the aim of improving the performance of the models.

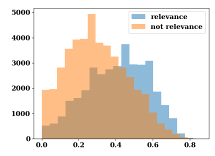
Pattern identification were mainly done by visualizing the dataset. With binary variable is-homepage, a table was constructed in which is-homepage and relevance were columns heading and rows heading. To deal with the high-dimensional predictor provided, histograms were drawn to determine the distribution of ordered and continuous features and how they relate to the relevance.



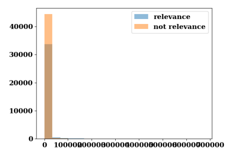
(a) query_length



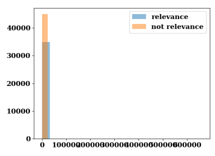
(b) sig1



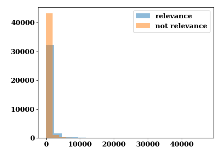
(c) sig2



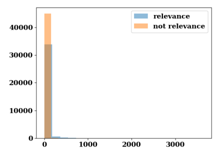
(d) sig3



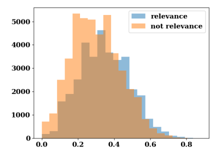
(e) sig4



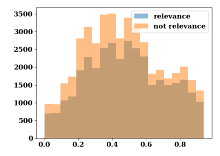
(f) sig5



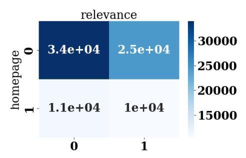
(g) sig6



(h) sig7



(i) sig8



(j) is_homepage

Figure 1: Histograms and table of the distributions of the variables

As shown by the table and plots above, several possible processing can be determine. Due to the extremely unbalanced distribution shown in sig3, sig4, sig5, and sig6, applying logarithm function to them may provide benefit to the model because it provide a more evenly spread data. Additionally, considering the substantial data near the origin, eliminating the other data as outlier may leads to improvement of the model. is_homepage exhibits a pattern where proportion of relevant and non-relevant data is approximately equal when the URL is not a homepage, whereas the amount of relevance data is 50% higher when the URL is a homepage. According to this, it is worthwhile trying fitting the data with is_homepage 0 or 1 to separate models or balancing the data with resampling, SMOTE and similar techniques.

To investigate the potential multicollinearity, the variance inflation factors(VIF) were calculated and compared relatively give that there was no absolute threshold for VIF. logistic regression model was subsequently employed to assess the impact of excluding the variables with higher VIFs. This results informed to the decision on whether the elimination of specific variable should be tested for the final model.

	query_length	is_homepage	sig1	sig2	sig3	sig4	sig5	sig6	sig7	sig8
VIF	2.92	1.92	2.81	6.11	3.22	1.10	3.90	1.35	8.80	6.50

Table 1: Variance Inflation Factors

The VIF of sig2, sig7 and sig8 are notably higher, exceeding a value of 6 while the VIF of other variables are lower than 4. Then, it was reasonable to assume that they may have multicollinear relationship, meaning they were able to be expressed as linear combination of other features. Based on this results, eliminating these variables may be advantageous.

It should be noted that the basic analysis of data above would not guarantee to perform well in the final model.

2 First Model: CNN Model

Our third model used is a CNN model. The model employs feature engineering, stratified cross-validation, data standardization and separate model training for different search contexts (specifically, searches initiated from a homepage versus other sources).

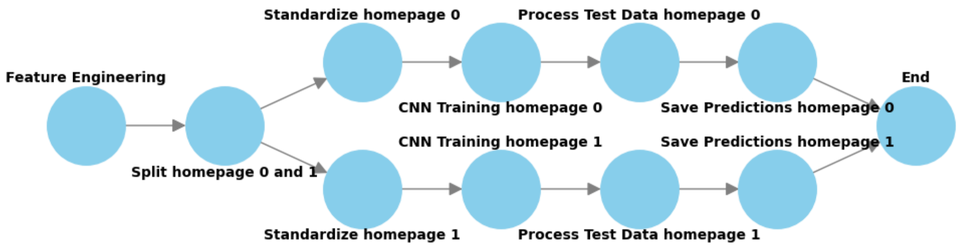
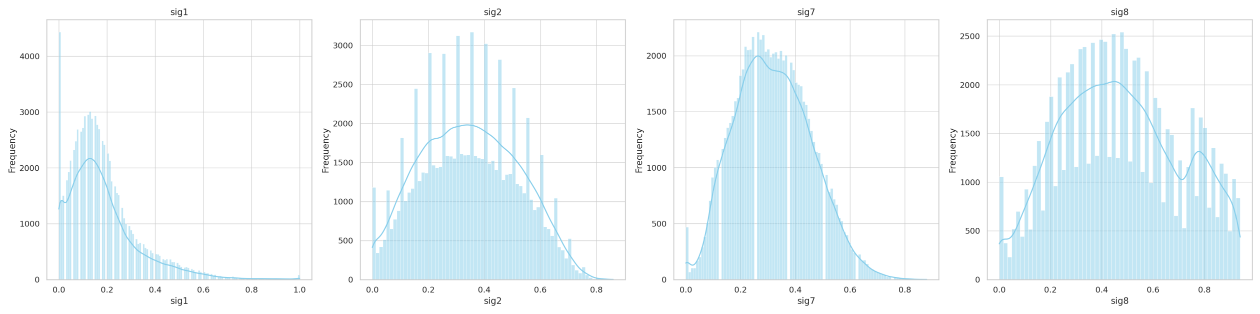


Figure 2: Flow Chart of CNN Model

2.1 Feature Engineering

Upon reviewing the data, we found that sig1, sig2, sig7, and sig8 are all within the range of 0-1, and their distributions are very tight.



Though sig3, sig4, sig5 sig6 aren't within [0, 1], their distributions are tight as well. Therefore, before building the model, we should expand the data scale so that we could capture more detailed characteristics from them.

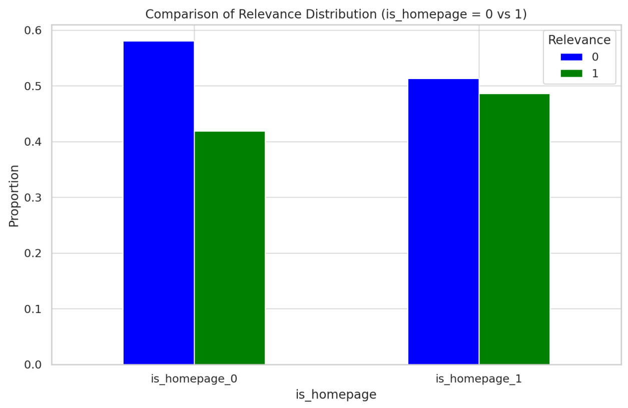
In practical online scenarios, considering that sometimes a correct search result is only achieved under multiple conditions, I have also chosen to use interaction terms. Also, I have decided to utilize Local Statistical Features like standard deviation and mean to capture the underlying characteristics of the data.

Below is a summary for the feature engineering methods I utilized:

Technique	**Description**
Interaction Terms	Multiply pairs of features to capture interactions.
Local Statistical Features	Compute rolling means and standard deviations to capture local statistics.
Multi-Scale Features	Apply transformations (square, sqrt) to create additional features.

2.2 Separately Handling Homepage 0 and 1

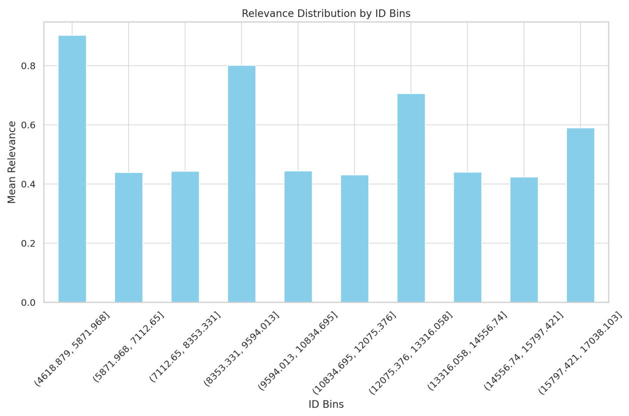
In real-world applications, the context in which a search is initiated (e.g., from a homepage or elsewhere) can significantly impact the relevance of search results. Also, from data analysis, the relevance ratios when homepage is 0 or 1 are slightly different:



Due to these observations, the model was trained separately for ‘is_homepage=0‘ and ‘is_homepage=1‘ cases, allowing for a more tailored approach that improves prediction accuracy.

2.3 Stratified Cross Validation

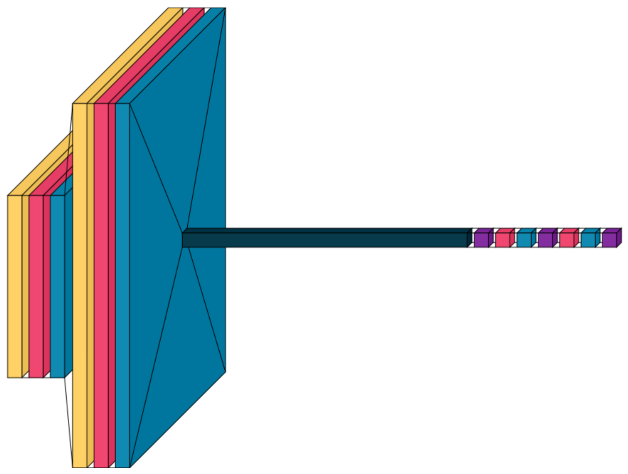
From the data analysis, the relevance ratio is extremely unevenly distributed among the data.



Therefore, stratified 5-fold sampling will be used to efficiently and unbiasedly utilize the data.

3 Model Architecture

The model architecture is a sequential CNN composed of the following layers:



Layer Type	Dimensions	Description
Input Layer	105 × 1	The input layer takes in the raw data with 105 features, each feature corresponding to one input channel. The data is then passed to the first convolutional layer.
Convolutional Layer 1	103 × 64	The first convolutional layer applies 64 filters of size 3, reducing the spatial dimension to 103 while increasing the depth to 64. This layer captures local patterns in the data.
Convolutional Layer 2	101 × 128	The second convolutional layer uses 128 filters, further reducing the spatial dimension to 101 and increasing the depth to 128, extracting more complex features.
Pooling Layer (Optional)	50 × 128	If a pooling layer is applied after the convolutional layers, it typically halves the spatial dimensions, reducing the size to 50 while retaining the depth of 128. This layer helps reduce computational complexity.
Fully Connected Layer 1	128	The first fully connected layer takes the flattened output from the previous layers and uses 128 neurons to perform further processing, capturing high-level interactions between features.
Fully Connected Layer 2	64	This layer has 64 neurons and further refines the features extracted by the previous layers, reducing the dimensionality and preparing for the final classification.
Output Layer	1	The final output layer has a single neuron, typically using a sigmoid activation function for binary classification, outputting a probability score between 0 and 1.
Data Flow	–	The black line in the diagram represents the flow of data through each layer, from the input to the final output.

The model is trained using the Adam optimizer with a learning rate of 0.0001, and the loss function is binary cross-entropy, appropriate for binary classification tasks.

3.1 Results

The model individually achieved 0.68706 on public test data, laying a foundation for further voting strategy.

4 Second Model: Fully Connected Neural Network

The second classifier used is a fully connected multi-layer neural network with basic structure depicted in Figure . A repeatable module that comprise three hidden layers with H1, H2, and H1 hidden nodes respectively, was defined, along with a hyper-parameter module-num to increase the flexibility of the model.

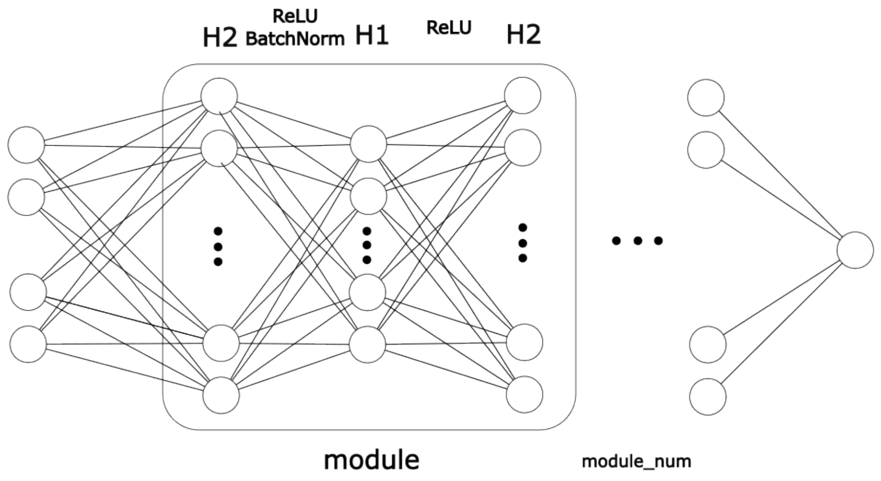


Figure 3: Neural Network Structure

H1, H2, module-num, learning-rate of Adam optimizer, batch-size and epoch were hyper-parameters that is optimized using grid-search. To minimize the time for grid-search, no feature engineering were performed on the data. The trials and outputs of each grid-search are presented in the following table.

parameters	best parameters	accuracy
H1: [50, 75, 130, 500]	500	0.6603702902793884
H2: [40, 90, 100, 150]	150	
Batch_size: [1500, 1000, 500, 100, 5000]	500	
learning_rate: [1e-4, 1e-3]	0.0001	
epoch: [1]	1	
chunk_num: [2, 3]	2	
H1: [550, 600]	600	0.660432755947113
H2: [175, 200, 225]	200	
Batch_size: [500]	500	
learning_rate: [1e-4]	0.0001	
epoch: [1]	1	
chunk_num: [2, 3]	2	

H1: [600]	600	0.6649801135063171
H2: [50, 200]	200	
Batch_size: [500, 750]	500	
learning_rate: [1e-4, 1e-3]	0.0001	
epoch: [5, 10]	10	
chunk_num: [2, 1]	1	
H1: [600]	600	0.665479838848114
H2: [200, 300, 400]	200	
Batch_size: [500, 600, 750]	500	
learning_rate: [1e-4, 1e-3]	0.0001	
epoch: [5, 10, 20, 30]	20	
chunk_num: [2, 1]	1	

Table 2: Grid Search

According to the results, the parameter set (600, 200, 500, 1e-4, 20, 1) is the best for the model.

After tuning the parameters, it is worthwhile to do feature engineering that manually provide additional information to the model. This is favorable if the provided information is closely related to the output as it reduced the complexity of training the model.

The transformation of variables focused on the feature of the dataset that each query-id corresponds to multiple url-id. In this context, data sharing the same query ID is referred to as a group. From this perspective, the relevance and not-relevance within a group were expected to follow a specific distribution. Because of this, providing the mean and standard deviation of the group was of benefit as it, to some extent, provided the model with insights into the distribution of the group. Similarly, the interaction of features were introduced as a alternative of covariance that provide relatively relationship between predictors in the distribution. This approach was intended to enhance the efficiency of the algorithm because calculation of covariance in this case should be done separately for each groups but the interaction can be applied to the whole dataset. Given that the full set of feature pair combinations exceeded 50, only a proportion of them were chosen to be created depending on their impact on model performance after inclusion. Normal transformations including logarithm, exponent and power functions are evaluated and tested to determine whether it was added or not. The structure of the dataset after additional feature was added is shown in figure 2.

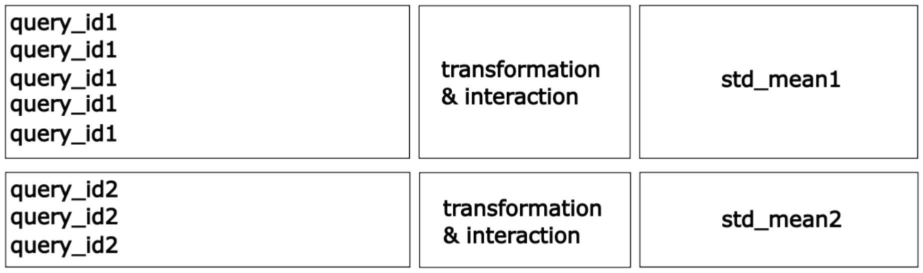


Figure 4: Dataset after feature engineering

features	accuracy
non	0.665479838848114
std-mean, interaction	0.6702645421028137

Table 3: Cross Validation Accuracy with Std-mean and Interaction

Without changing the hyper-parameters the feature engineering brought a improvement of nearly 0.5% to the model.

A small-scale grid-search is done afterwards as the previous tuning of hyper-parameters did not account for the newly generated features. This grid-search was performed with a reduced combination of hyper-parameter due to increased time required for training.

parameters	best parameters	accuracy
H1: [600]	600	0.6725257039070129
H2: [200, 300]	200	
Batch_size: [500, 750]	500	
learning_rate: [1e-4, 1e-3]	0.0001	
epoch: [12, 16, 20]	12	
chunk_num: [1]	1	

Table 4: Grid Search with Feature engineering

The grid-search suggested that (600, 200, 500, 1e-4, 12, 1) is the best hyper-parameter set for this neural network with a cross validation accuracy of 67.25% which has a 0.23% of improvement compared to the former parameters used.

5 Third Model: XGBoost

XGBoost (eXtreme Gradient Boosting) is an advanced implementation of gradient boosting that incorporates several enhancements, making it both more efficient and powerful. The following key features distinguish XGBoost from traditional gradient boosting methods:

- **Regularization:** XGBoost integrates L1 (lasso) and L2 (ridge) regularization terms into its objective function. This helps to mitigate overfitting by penalizing overly complex models, which can be particularly useful in preventing the model from fitting noise in the training data.
- **Weighted Quantile Sketch:** This technique enables XGBoost to effectively handle weighted data when constructing decision trees, improving its flexibility and accuracy in a wide range of applications.
- **Parallel Processing:** One of XGBoost's most significant improvements over traditional gradient boosting is its ability to train models in parallel, drastically reducing training times while maintaining accuracy.
- **Tree Pruning:** XGBoost employs a method known as "max depth" for tree pruning. Instead of allowing the tree to grow to its maximum depth, which could lead to overfitting, unnecessary branches are pruned to create a simpler, more generalizable model.

In our model, the training data was divided into two subsets based on whether the webpage was a homepage. A grid search was then utilized to optimize the hyperparameters for each subset. The best parameters for the training set that did not contain homepage data are illustrated in the figure below, and the best parameters for the training set that contained homepage data are shown in the subsequent figure.

Parameter	Non-Homepage	Homepage
`colsample_bytree`	1.0	1.0
`learning_rate`	0.15	0.25
`max_depth`	4	3
`n_estimators`	70	70
`subsample`	0.8	1.0

Table 5: Optimized Hyperparameters for XGBoost

After training, we observed that the accuracy of the model trained on the non-homepage dataset was significantly lower. A closer examination of the data revealed that the number of samples labeled as relevance 0 was 10% higher than those labeled as relevance 1, indicating a class imbalance. To address this, we switched from using accuracy score to F1 macro score for model evaluation. This adjustment led to an improvement in model performance, particularly in handling the imbalanced classes.

6 Voting Strategy

The voting strategy is an ensemble method where predictions from multiple models are combined to make a final decision. This approach often results in better performance than any single model due to the reduction in variance and the averaging out of individual model errors.

6.1 Mathematical Intuition

Given a binary classification problem with N independent models, each with an accuracy $p > 0.5$, the probability that the majority vote among these models is correct can be expressed using the binomial distribution:

$$P(\text{correct majority vote}) = \sum_{k=\lceil N/2 \rceil}^{N} \binom{N}{k} p^k (1-p)^{N-k} \tag{1}$$

Since $p > 0.5$, this probability is generally higher than the accuracy of any single model. The improvement arises because while individual models may make different errors, the ensemble's majority vote tends to correct these errors more often than not. This reduces the overall variance of the predictions, leading to more accurate and reliable final predictions.

6.2 Application to Our Model

In our project, the individual models (a CNN, a fully connected NN, and an XGBoost model) each achieved accuracies in the range of 0.66 to 0.68. By applying the voting strategy to combine their predictions, we observed a 1% improvement in prediction precision. This improvement demonstrates how the ensemble approach capitalizes on the strengths of multiple models, making the overall system more robust and less prone to the errors that might be made by any individual model.

Appendix

Model Code

The codes of the three models and voting strategy can be found in the GitHub repository: `https://github.com/AlastorrrXu/2024Summer_STATS202_Project_D.-Xu-S.-Chen-H.-Wang`