Nursing Home Search Recommendation System

1 Introduction

In this project, we are going to build a search engine and a recommendation system specifically for a group of nursing homes, so that users can interact with the systems, query the group of nursing homes, and automatically get recommendations based on the returned search results. The motivation of this project is based on the social background of the increasing demand for nursing homes and the shortage of nursing homes. Thus, a tool to search for the appropriate nursing homes to meet the society needs are therefore in need. By the completion of this project, the elder population with concerns and needs for nursing homes, as well as families or children who may be the cost sponsors and concerned about the quality their family/parents' life in nursing homes will benefit a lot from it. To achieve the goal, we implemented several models for the part of building the search engine, including *BM25* as baseline, *TF-IDF, Random Forest, Decision Tree, SVR, Gradient Boosting Decision Tree, Fastrank, Lightgbm*, and some customized models. In particular, *Fastrank* model performs best among all the models, and exceeds that of *BM25*. Though some of the models that we implemented are also have been used for the search engine building in some references, we customized and tuning the models according to the characteristics of our dataset so that they can fit better. For the building of the recommendation system, k-nearest neighbors (*KNN*) model is the main one used. Therefore, from the overall perspective, this project combines many useful techniques for building the search engine and recommendation systems, and the deliverables have practical significance.

2 Data

The data we have selected comes from the healthcare sector, focusing on nursing homes. The dataset contains not only some specific information about nursing homes, such as their names, cities, detailed addresses, etc. but also the ratings of nursing homes by government/official agencies from different dimensions so that it can meet our two objectives of building a simple search engine and a recommendation system based on the ratings. In this section, section 2.1 introduces the sources of data and how to obtain them, section 2.2 shows some data features and presentation,section 2.3 shows the data pre-processing we have done so far, and section 2.4 introduces the data annotation for the search engine system.

2.1 Source of the data

In this project, we used the dataset obtained from https://data.cms.gov/. This website CMS data provides direct access to public data released by the Centers for Medicare & Medicaid Services (CMS). We used the Provider Information dataset in this project, which contains general information about currently active nursing homes, including the number of certified beds, quality measure scores, staffing, and other information used in the five-star rating system. The data is presented as one row per nursing home. It was updated by CMS with 2022-09-01 and released on 2022-09-28.

2.2 Data characterization and presentation:

2.2.1 Overview of the dataset

The selected dataset contains 15,151 data with 95-dimensional features. In addition to the basic information about nursing homes, it also contains rating features of nursing homes in different dimensions. Based on the preliminary analysis of the dataset, we divided the features of 95 dimensions into four major categories: basic information about nursing homes, information about nursing home personnel and facilities, rating information, and penalized information. The specific dictionary of the dataset is shown in Table 1 below. (Due to space limitation, only 5 features of each category are shown here.)

Table 1. Specific dictionary of the dataset

Feature Category	Feature Name	Feature Data Type	Feature Meaning
Basic Information	Federal Provider Number	object	Federal Provider Number
	Provider Name	object	Nursing homes' name
	Provider Address	object	Nursing homes' address
	Provider City	object	City of the nursing home
	Provider State	object	State of the nursing home
Personnel and Facilities Information	Number of Certified Beds	int64	Number of certified beds in nursing homes
	Average Number of Residents per Day	float64	Average daily number of elderly people contained in nursing homes
	With a Resident and Family Council	object	Whether the nursing home set up residents and family committees
	Provider Resides in Hospital	object	Whether the medical staff of the nursing home lives in the nursing home
	Date First Approved to Provide Medicare and Medicaid Services	int64	Date First Approved to Provide Medicare and Medicaid Services
Rating Information	Rating Cycle 1 Health Deficiency Score	object	Grades in health deficiency in the first scoring cycle
	Rating Cycle 1 Health Revisit Score	object	Grades in health revisit in the first scoring cycle
	Rating Cycle 1 Total Health Score	object	Total health rating for the first scoring cycle
	Overall Rating	float64	Five-Star Quality Rating System
	Total Weighted Health Survey Score	float64	Total Weighted Health Survey Score

Penalized Information	Number of Facility Reported Incidents	int64	Number of incidents reported by nursing homes
	Number of Substantiated Complaints	int64	Number of Substantiated Complaints
	Number of Fines	int64	Number of Fines
	Total Amount of Fines in Dollars	float64	Total Amount of Fines in Dollars
	Total Number of Penalties	int64	Total Number of Penalties

2.2.2 Missing values of the dataset

The 95-dimensional features contained in the dataset were observed for missing values using the info method, and some of them were found to contain a large number of cases of missing values. The data with missing values were summarized, and the main results are shown in the following Table 2.

Table 2. Summary statistics of missing values

Feature Name	Number of missing values	Percentage of missing values
Ownership Type	1	< 0.01%
Average Number of Residents per Day	79	0.52%
Average Number of Residents per Day Footnote	15072	99.48%
Special Focus Status	14624	96.52%
Overall Rating	193	1.27%
Overall Rating Footnote	14958	98.73%
Health Inspection Rating	193	1.27%
Health Inspection Rating Footnote	14958	98.73%
QM Rating	253	1.67%
QM Rating Footnote	14898	98.33%
Long-Stay QM Rating	613	4.05%
Long-Stay QM Rating Footnote	14538	95.95%
Short-Stay QM Rating	2937	19.38%
Short-Stay QM Rating Footnote	12214	80.62%
Staffing Rating	316	2.09%
Staffing Rating Footnote	13293	87.74%
Reported Staffing Footnote	14540	95.97%
Reported Nurse Aide Staffing Hours per Resident per Day	611	4.03%
Reported LPN Staffing Hours per Resident per Day	611	4.03%
Reported RN Staffing Hours per Resident per Day	611	4.03%
Reported Licensed Staffing Hours per Resident per Day	611	4.03%
Reported Total Staffing Hours per Resident per Day	611	4.03%
Total number of nurse staff hours per resident per day on the weekend	611	4.03%
Registered Nurse hours per resident per day on the weekend	611	4.03%
Reported Physical Therapist Staffing Hours per Resident Per Day	611	4.03%
Total nursing staff turnover	2693	17.77%
Total nursing staff turnover footnote	12458	82.23%
Registered Nurse turnover	3636	24.00%
Registered Nurse turnover footnote	11515	76.00%
Number of administrators who have left the nursing home	4634	30.59%
Administrator turnover footnote	10517	69.41%
Case-Mix Nurse Aide Staffing Hours per Resident per Day	611	4.03%
Case-Mix LPN Staffing Hours per Resident per Day	611	4.03%

Case-Mix RN Staffing Hours per Resident per Day	611	4.03%
Case-Mix Total Nurse Staffing Hours per Resident per Day	611	4.03%
Adjusted Nurse Aide Staffing Hours per Resident per Day	684	4.51%
Adjusted LPN Staffing Hours per Resident per Day	684	4.51%
Adjusted RN Staffing Hours per Resident per Day	684	4.51%
Adjusted Total Nurse Staffing Hours per Resident per Day	684	4.51%
Adjusted Weekend Total Nurse Staffing Hours per Resident per Day	684	4.51%
Rating Cycle 2 Standard Health Survey Date	107	0.71%
Rating Cycle 3 Standard Health Survey Date	236	1.56%
Total Weighted Health Survey Score	107	0.71%
Number of Citations from Infection Control Inspections	75	0.50%

According to the above Table 2, it can be found that the presence of missing values in the features is very common and the general percentage of missing values varies widely, so it is necessary to discuss the missing values by features in the data preprocessing section. The treatment of missing values will be described in detail in 2.3 Data pre-processing.

2.2.3 Distribution of five-star rating feature levels

Considering that the grades of the five-star rating features will be used when making predictions for the recommendation system, the grade distribution of these grade features is first viewed to facilitate feature selection at a later stage. The specific distribution is shown in Figure 1 below.

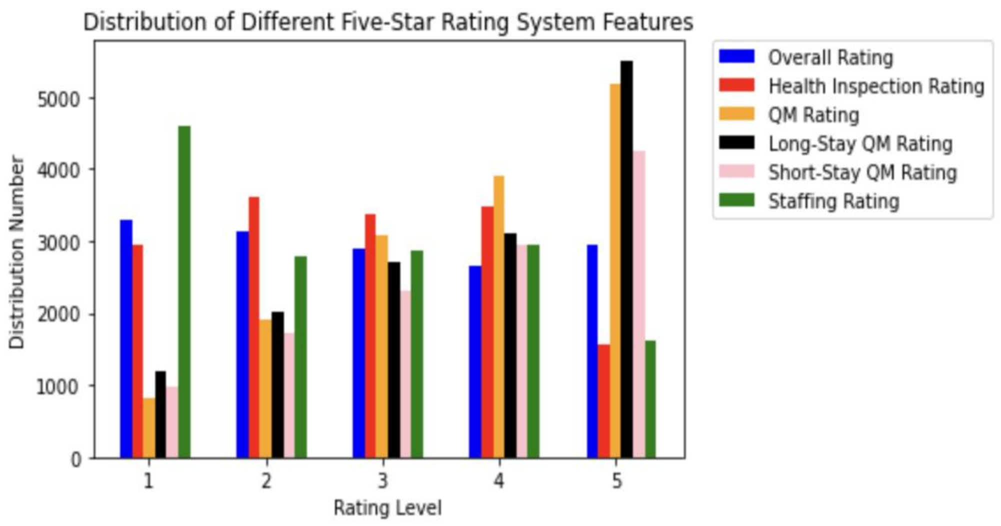


Figure 1: CMS created the Five-Star Quality Rating System to help consumers, their families, and caregivers compare nursing homes more easily. Nursing homes with 5 stars are considered to have much above average quality and nursing homes with 1 star are considered to have quality much below average. According to the above figure, it can be seen that the distribution of the ratings of these five-star rating systems is quite different, for example, the overall rating is more evenly distributed among the five grades, while QM Rating, Long-Stay QM Rating, and Short-Stay QM

Rating show a greater difference in distribution and show a trend of higher distribution with higher stars.

As seen in Figure 1 above, there are some scores with the same trend and distribution. Therefore, when preprocessing the data, we decided to remove some of the similar features to avoid the creation of multicollinearity cases. The specific data preprocessing process will be described in section 2.3 Data preprocessing.

2.3 Data pre-processing

2.3.1 Missing value processing

According to 2.2.2Missing values of the dataset, we can see that the dataset has a high number of missing values and therefore needs to be filled with missing values. The features with missing values are extracted and a bar chart is drawn for the percentage of missing values to see the specific distribution of missing values. The details are shown in Figure 2 below.

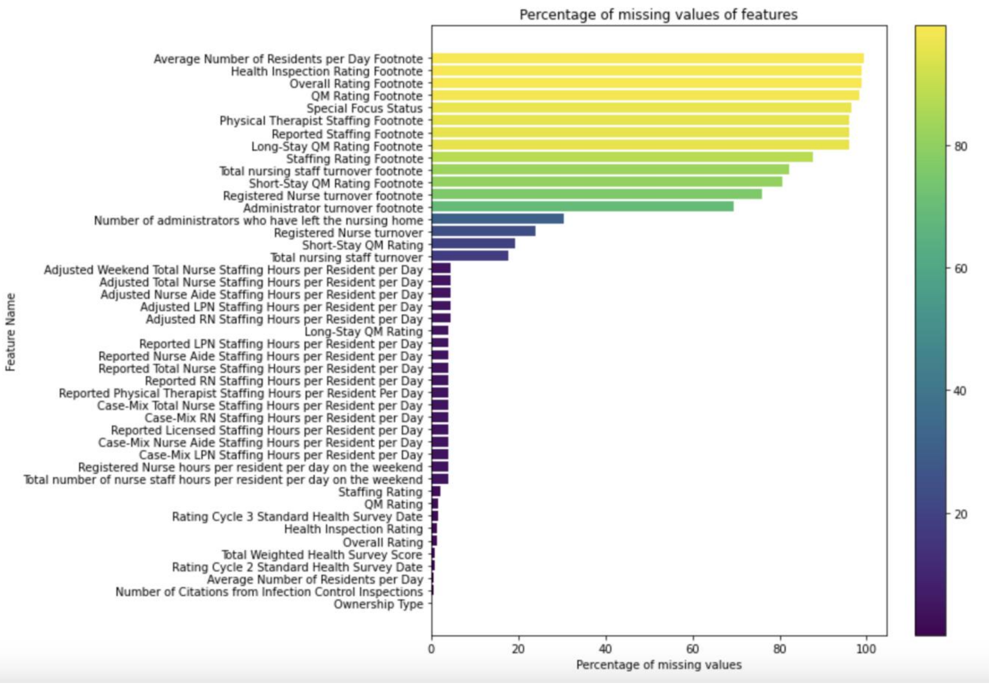


Figure 2: The figure shows the percentage of missing values for different features, in descending order of the percentage of missing values.

According to Figure 2 above, it can be found that those with more missing values are Footnote features, i.e., annotations of specific features, and often the annotations correspond to the number of missing features. Therefore, these features with footnote correspondence are filled according to the footnote category, and these footnote features are deleted after filling. For the remaining features with missing values, we use a uniform criterion that features with 30% of missing values are directly excluded, and features with less than 30% of missing values are filled with median/mean value if they are numerical features, and with a separate '0' category if they are categorical features.

2.3.2 Initial feature screening

Step 1. Eliminate irrelevant features

There are multiple features in the data that are meaningless to the model, such as Rating Cycle 1 Standard Health Survey Date, etc. These meaningless features are chosen to be directly excluded from the process. The specifically excluded variables are shown in Table 3 below.

Table 3. Meaningless features for rejection

Feature Name
Rating Cycle 1 Standard Survey Health Date
Rating Cycle 2 Standard Survey Health Date
Rating Cycle 3 Standard Survey Health Date
Processing Date

Step 2. Check the correlation between the features

From the description of the previous data, it is clear that there is some correlation between some of the characteristics. Therefore, we chose to plot the correlation heat map for numerical features to see the correlation between the features. The specific heat map is shown in Figure 3 below.

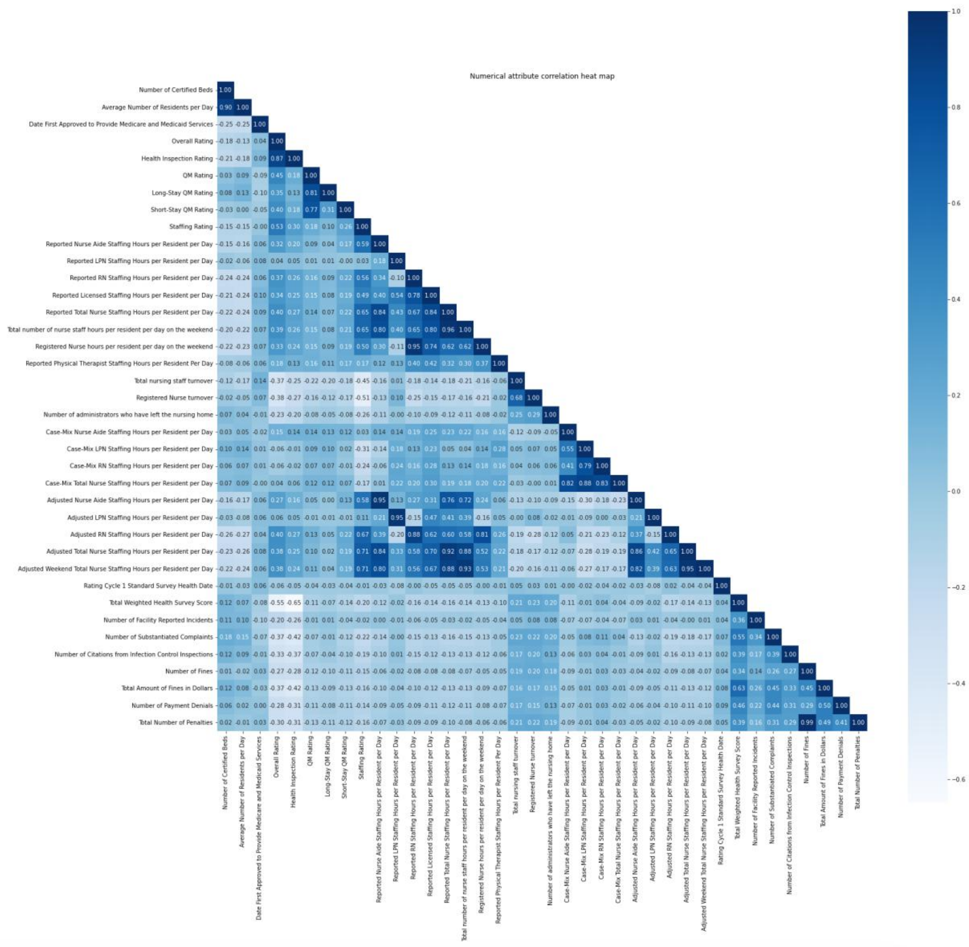


Figure 3: The above figure shows the heat map of the correlation coefficient between features. For the calculation of correlation coefficients, we chose Pearson correlation coefficients, 0.8 to 1.0 very strong correlation, 0.6 to 0.8 strong correlation, 0.4 to 0.6 moderate correlation, 0.2 to 0.4 weak correlation, and 0 to 0.2 very weak or no correlation.

According to Figure 3 above, it is found that there are indeed extremely strongly correlated groups of features. We choose to keep only one of the features for such strongly correlated feature groups, so some of the features are eliminated. The specific excluded features are shown in Table 4 below.

Table 4. Extremely strong correlation characteristics for rejection

Feature Name
Number of Fines

Although there are many other extremely strongly correlated feature groups, considering that these features have some realistic significance, such as 'Registered Nurse hours per resident per day on the weekend' and ' Reported RN Staffing Hours per Resident per Day', both of which have correlation coefficients as high as 0.95, but one is the average number of hours registered nurses (RNs) spend per resident on the weekend and the other is the overall average number of hours per resident per day, both of which have a strong impact on the final nursing home rating, so both characteristics need to be retained.

2.4 Annotation data

Because the dataset itself does not contain relevance scores to the query, we need to do the annotation manually. We selected 40 queries, used the bm25 model to rank each query, and then selected the top 120 documents for manual 5-point relevance score annotation. The specific 40 queries selected are shown in the following Table 5.

Table 5. Queries

Query 1-20	Query 21-40
Monroe street	Non-profit providers
Burns nursing home	Which provider is in ALABAMA
1600 west hobbs street	Nearby nursing homes for area with zip code of 36065
Medicare type nursing home	Does EAST ALABAMA MEDICAL CENTER SILLED NURSING FACILI prvide Medicaid
Morgan avenue southwest decatur AL	Medicare and Medicaid in nursing homes
Nursing home in Birmingham	Which company does DIVERSICARE OF FOLEY belong to
Nursing home in LA	Beds for HATLEY HEALTH CARE INC
New nursing home	Non-profit nursing homes with both Medicare and Medicaid in AK
High rate nursing home in IL	For-profit nursing homes with more tthan 200 beds in AK
Which city has more nursing homes	Are COTTAGE OF THE SHOALS and RUSSELLVILLE HEALTH CARE INC in the same city
Does Keller Landing have a resident or family council	Rating for HIGHLANDS HEALTH AND REHAB
Which nursing home's ownership type is government	Physical Therapist in COOSA VALLEY HEALTHCARE CENTER
San Diego	Going to which state is more suitable for finding nursing homes
Nursing home has more beds	Which provider in CA is Non profit - Corporation
What's the phone number of Sunset Manor	Which state is NORTHWAY HEALTH AND REHABILITATION, LLC in
What's the zip code and location of the Health care inc	How is EVERGREEN NURSING HOME overall
What nursing home received none fine in AL	Which nursing home in AK has the most capacity
Rating of west gate village	How long has been since MERRY WOOD LODGE provided Medicare and Medicaid Services
Is there any nursing home better than Merry wood lodge in elmore	How many nursing homes does CLANTON have
QM Rating greater than 4	Which nursing home is the best in CLANTON

3 Related Work

Considering the main problems we are planning to solve for this project, the discussion of the related work can be divided into two parts: (1) Search Engine building for nursing homes and (2) Recommendation System building for the nursing homes.

For building search engine for the specific set of nursing homes, or other objects, there is a lot of existing techniques, such as using Machine Learning techniques including *SVM, Artificial Neural Network*, and *XGBoost* to predict or determine the relevance of the webpage based on the given query (Ramana et al., 2021), or using models like *BM25, TF-IDF*, or utilizing Deep Structured Semantic Model (DSSM) to build extra similarity features to enhance search engine relevance (Ye et al., 2016), etc. However, despite of the specific search objects (nursing-homes in the case of our project), there are still some differences between the existing models/techniques and ones that we are going to use in this project. For instance, according to Ramana et al., *SVM* is a good technique to predict whether each web page in the testing set was relevant to the given query or not. However, based on the features of our dataset, the *SVR* (regression model) may also be a good suit for predicting the ratings of the retrieved documents in terms of the relevance between each of them and the given query since we can also treat rating values as continuous variable. Additionally, in our project, we are going to tune the hyper-parameters in the model based on our specific dataset instead of using default ones, which should be more customized and perform better on our data features. Based on the reference and assumptions, for this project, we tried and implemented some methods mentioned above, which includes *BM25, TF-IDF, SVR* etc.

For building recommendation system for the nursing homes listed on our dataset, or other objects, the existing techniques are including top-n framework, like collaborative filtering, etc. According to Zolaktaf et al., their work is to improve the general top-n framework by using historical rating data to learn user long-tail novelty preferences, so that the coverage of the recommendation system could be increased. Besides, more detailed branches of general techniques, such as location-based service (LBS) recommendation system, are also introduced by Kuo et al., which is integrating the application of LBS with recommendation technologies to present a location-based service recommendation model (LBSRM) to best recommend and evaluate relevance according to the change of locations. Moreover, for building recommendation systems when there is no sufficient historical data for users (for example, new users), except for the cold start, Han and Karypis suggest a new algorithm of feature-based that can overcome the limitations of the existing top-n recommendation algorithm. With all those related techniques, in our project, we tried to absorb multiple existing techniques with the traditional version of the models to predict and recommend the relevant nursing homes based on the users' search results. However, instead of using the most commonly used techniques, we tried to combine the prediction results from regression models with machine learning techniques, such as KNN to build the recommendation system since the model will be more suitable to use for our dataset. Besides, our methodology and models will consider as many features (such as location of the nursing homes, the predicting over all ratings etc.) as possible when doing the relevance evaluation and recommendations, so that the recommendation system will be more customized for this specific data set.

4 Methods

In this project, we tried multiple models and techniques and compared them according to evaluation criteria (evaluation criteria are described in detail in the **Evaluation and Results** section) in order to find the models and techniques with the best performance in search engine and recommender system construction, respectively.

4.1 Search Engine

4.1.1 Feature Introduction

When performing feature filtering, we first thought that some features of the text and query itself could bring some optimization to the results. Therefore, for this consideration, we chose to add three common features: the BM25 code score, the TF-IDF code score, and the coordinate match score for the query-i.e. how many query terms appear in the code.

Secondly, we consider that users generally prefer to search for nursing homes that have a better overall rating, so excluding the text itself, if we add some features that score different dimensions of nursing homes, this can make nursing homes with a higher overall rating rank more highly in the search results, which is in line with the actual goal. Therefore, after several attempts, we finally kept the following three nursing home dimensional scoring features: Health Inspection Score, QM Score, and Staffing Score. We believe that a more comprehensive assessment of a nursing home's rating can be made through these three-dimensional scores.

In summary, the final characteristics brought into the model are shown in the following table 6.

Table 6. Feature Pipeline

Feature Name	Feature Meaning	Realization Method
BM25 Score	the BM25 code score	pyterrier
TF-IDF Score	the TF-IDF code score	pyterrier
CoordinateMatch	the coordinate match score for the query i.e. how many query terms appear in the code	pyterrier
Health Inspection Score	Health Inspection Score, a 1-5 star rating, with 5 stars being the best and 1 star being the worst. The better the score, the higher the score in the health inspection, the fewer medical safety incidents in the nursing home, and the better the facilities are equipped to handle emergencies properly.	define merge_hir(row) apply.doc_score(merge_hir)
QM Score	QM Score, a 1-5 star rating, with 5 stars being the best and 1 star being the worst. The better the score, the higher the overall quality of the nursing home.	define merge_qm(row) apply.doc_score(merge_qm)
Staffing Score	Staffing Score, is a 1-5 star rating, with 5 stars being the best and 1 star being the worst. The better the score, the higher the staffing score of the nursing home, i.e. the staff is reasonably distributed, each elderly person has sufficient staff to accompany them, the overall quality of the staff is good, the percentage of regular staff is high, and the staff mobility is low and stable.	define merge_sr(row) apply.doc_score(merge_sr)

4.1.2 Model Introduction

For the search engine part, the data preprocessing has been described in detail in the Data section, so we will not go into too much detail here. For the model selection part of the search engine, we mainly used *BM25, TF-IDF, Random Forest, Decision Tree, SVR, Gradient Boosting Decision Tree, Fastrank, Lightgbm*, and *custom models*. The specific models used are shown in the following table 7.

Table 7. Specific Models Used

Model name	Specific Used Way	Parameters
BM25	import pyterrier	k_1 = 1.8 b = 0.75 k_3 = 8
TF-IDF	import pyterrier	-

Random Forest	import sklearn	n_estimators = 400
		verbose = 1
		random_state = 42
		n_jobs = 2

Decision Tree	import sklearn	max_depth = 4
		min_samples_leaf = 1
		min_samples_split = 2
		random_state = 42

SVR - Linear	import sklearn	C = 0.15
		gamma = 1e-21

SVR - RBF	import sklearn	C = 5
		gamma = 0.0012

Gradient Boosting Decision Tree	import sklearn	learning_rate = 0.1
		max_depth = 1
		n_estimators = 100

Fastrank	import fastrank	init_random = True
		normalize = True
		seed = 1234567

Lightgbm	import lightgmb	silent = False
		min_data_in_leaf = 1
		min_sum_hessian_in_leaf = 1
		max_bin = 255
		num_leaves = 31
		objective = "lambdarank"
		metric = "ndcg"
		ndcg_eval_at = [10]
		ndcg_at = [10]
		eval_at = [10]
		learning_rate = .1
		importance_type = "gain"
		num_iterations = 100
		early_stopping_rounds = 5

Custom Model	custom	k_1 = 8 b = 0.05 k_3 = 10
Uni Model (Custom Model + Fastrank)	custom pipeline	same as Custom Model and Fastrank

4.1.3 Custom Model Introduction

We have tried to improve the *BM25* model with custom model improvements to enhance the model effect. The model equation is shown below:

$$IDF\ Part\ =\ ln(\frac{numOfDoc\ -\ termDocFreq\ +\ 0.5}{termDocFreq\ +\ 0.5})$$

$$TF\ Part\ = (termFreqDoc\ -\ \frac{termFreq}{numOfDoc})\ *\ \frac{k_1\ +\ 1}{Normalizer\ +\ (termFreqDoc\ -\ \frac{termFreq}{numOfDoc})}$$

$$QTF\ Part\ =\ \frac{(k_3 +\ 1)\ *\ keyFreq}{k_3\ +\ keyFreq}\ \div\ ln(numOfUniqueTerm)$$

$$Normalizer\ = k_1\ *\ (1\ -\ b\ +\ b\ *\ \frac{docLen}{avgDocLen})$$

$$Score\ = IDF\ Part\ *\ TF\ Part\ *\ QTF\ Part$$

We took the *BM25* scoring function as the basis and improved some of its parts. As we know, the *BM25* scoring function consists of three parts: the variant form of IDF, the variant form of normalized TF, and the normalized QTF. Our custom function mainly improves the TF part and adds a part that quantifies the unique terms in all documents.

Firstly, we replaced *Term Frequency* in the TF section of the original *BM25* scoring function with *Term Frequency* - $\frac{Term\ Total\ Frequency}{Number\ Of\ Documents}$. Because, to assess whether a term is important in a document, we think we cannot use only the frequency of the term in that document. Suppose, a term is very frequent in all documents and it is not a stopword, then it is difficult for us to distinguish the difference in documents, which is very common in some specialized fields. Under our dataset, for example, the "nursing homes" can be a good example of this problem. In this case, the most straightforward approach is that we choose a baseline as the average frequency of occurrence of a term, and then use the frequency of occurrence of the term in each document to compare with the baseline, thus measuring the importance of the term in a document. In the refinement, we chose the $\frac{Term\ Total\ Frequency}{Number\ Of\ Documents}$ as the baseline. We achieved the “normalization” goal by subtracting the $\frac{Term\ Total\ Frequency}{Number\ Of\ Documents}$ from the original *Term Frequency*. After making the substitution, if the term appears more times in a document than the baseline, then we can say that the term is important in that document, which is an improvement on the original *BM25* function.

Secondly, we have added a partial response to the *Number Of Unique Terms* in the entire document. Considering that if the number of unique terms is large, the importance of the term in one document should be diminished, thus we need to consider the number of unique terms. Because the number of unique terms can be very large, we cannot divide it directly by itself, which would eliminate the effect of the other parts, so we used ln-processed. After multiplying three parts, the optimized version of *BM25* is divided by *ln(Number Of Unique Terms)* to obtain the final score.

4.2 Recommendation System

4.2.1 Feature Processing

Because many features in the dataset are of object type, and our prediction target, i.e., the overall nursing home rating, is a regression problem, we choose to label some of the object types used, converting them from strings to labeled data. We use the LabelEncoder method in the sklearn library to perform the labeling process. The specific features for labeling are shown in Table 8 below.

Table 8. Label Encoder Features

Feature Name
Provider City
Provider State
Provider County Name
Ownership Type
Provider Resides in Hospital
Continuing Care Retirement Community
Special Focus Status
Most Recent Health Inspection More Than 2 Years Ago
Abuse Icon
Provider Changed Ownership in Last 12 Months
With a Resident and Family Council
Automatic Sprinkler Systems in All Required Areas

4.2.2 Model Introduction

First, we normalized the data to improve the model results, using the StandardScaler() method of sklearn.preprocessing. Then we used the train_test_split() function of sklearn.model_selection to split the target variable "Overall Rating" in the ratio of 7:3 between the test and training sets, with random_state = 7 to keep the splitting in each run the results are consistent.

In terms of models, considering the final regression prediction model, we chose a variety of models including linear regression to observe the prediction results. The specific models and the methods used for the models are shown in Table 9 below. Some of the model's parameters are adjusted and optimized by GridSearchCv(n_jobs = -1, cv = 5).

Table 9. Specific Models Used

Model name	Specific Used Way	Parameters
Linear Regression	import sklearn	-

Decision Tree Regression	import sklearn	max_depth = 6
		min_samples_leaf = 2
		min_samples_split = 2

Gradient Boosting Regression	import sklearn	learning_rate = 0.1
		max_depth = 1
		n_estimators = 100

SVR - Linear	import sklearn	C = 0.15

		gamma = 1e-21

SVR - RBF	import sklearn	C = 5
		gamma = 0.0012

Stacking	import sklearn	First layer: Linear Regression, Decision Tree Regression, Gradient Boosting Regression Second Layer: Linear SVR, RBF SVR

4.2.3 Recommender system

For the construction of the recommendation system, we first determined that the features involved in the recommendation system are mainly the profile information of the nursing home (the state, city, and Ownership Type after labeling) and the Overall Rating of the nursing home (the rating feature we predicted earlier). To build the specific recommendation system, we choose to use the KNN model as the basis and use the KNN model to calculate the nursing homes with the highest similarity to the given nursing home and return the information of these nursing homes and the distance between them and the given nursing home. For the implementation of this part, we created the make_recommendation() function for implementation. See the following table 10 for the input parameters and usage of make_recommendation.

Table 10. Introduction of make_recommendation() function

Input parameters	Parameters' meaning	Parameter usage
model_knn	TheKNN model used.	We mainly use NearestNeighbors(metric = "cosine", algorithm = "brute") here, i.e., we use cosine similarity to calculate the degree of similarity between different nursing homes, using the brute force algorithm, because our training set has a suitable amount of data and a fast linear search speed.
data	Recommended data sets for the system	Here we use the nursing home dataset, but according to the previous introduction, only the profile information and Overall Rating features that the nursing home has been labeled are included.
mapper	The dictionary corresponding to the data set and the query method	Here we choose the corresponding nursing home query "docno", so we need to create a dictionary to store the data set index corresponding to "docno" so that we can use the index to retrieve the nursing home information we need after KNN.
nur_homes	Given a nursing home	We use the given docno of the nursing home
n_recommendations	Need to return how many similar recommendations	Due to the time limit of the search query, we choose n_recommendations = 3 for subsequent use.

4.3 The integration of search engines and recommendation systems

According to our initial project goal, we hope to be able to achieve the integration of a search engine and a recommendation system, i.e. after a user enters a query, we return 5 search results based on the model, and at the same time recommend three nursing homes that are most similar to this nursing home based on the nursing home of each search result.

Regarding the implementation of this part, we want to find the model that performs better according to the evaluation criteria as the final search model. Then the prediction of the Overall Rating for nursing homes (using the model with better prediction effect), then the nursing home profile information and Overall Rating are generated as features to be brought into the dataset of the recommendation system, using the my_recommendation function constructed above for The results returned by the search engine are used to recommend nursing homes one by one. The specific process is shown in Figure 4 below:

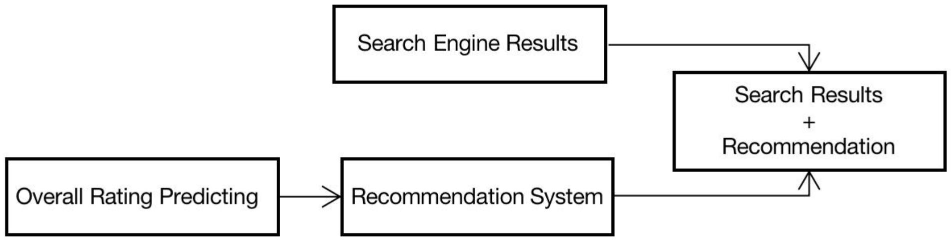


Figure 4: The above diagram shows the flow chart of fusing search engine results with the recommendation system. Based on the search results (docno) returned by the search engine, it is brought into the recommendation system to recommend the 3 most similar nursing home information for different search results.

5 Evaluation and Results

We have selected different evaluation methods for the IR system and recommendation system. In the search engine evaluation method section, two different baselines are selected, one baseline is selected in the recommendation system section, and quantitative evaluation metrics are selected for both sections. In this section, we introduce the evaluation method of the IR system in 5.1 and the evaluation method of the recommendation system in 5.2. The presentation results of the fusion search engine and recommendation system are presented in 5.3.

5.1 Evaluation for IR system:

To determine the performance of the algorithm, we calculate the nDCG@10 and nDCG of the rank data and the test data as a quantitative evaluation rule. The higher the nDCG@10 and nDCG means the better the model prediction. the nDCG is calculated as shown below:

$$DCG_p = rel_1 + \sum_{i=2}^{p} \frac{rel_i}{log_2 i}$$

$$nDCG_p = \frac{DCG_p}{IDCG_p}$$

5.1.1 Relevance Baseline

As the baseline for the most basic comparison of IR system performance, we choose the most straightforward cosine similarity evaluation. That is, we calculate the cosine similarity of all documents and queries and output the results according to their values from highest to lowest.

5.1.2 BM25

BM25 is an algorithm used to evaluate the relevance between search terms and documents, it is an algorithm proposed based on a probabilistic retrieval model and then describe the *BM25* algorithm in simple words: we have a query and a batch of documents Ds, now we want to calculate the relevance score between the query and each document D. The specific steps of the algorithm are shown below:

Step 1. Calculate the correlation between the word and D use IDF function as follows:

$$IDF(q_i) = log\frac{N - n(q_i) + 0.5}{n(q_i) + 0.5}$$

Step 2. Calculate the relevance between the word and query:

$$R(q_i, d) = \frac{f_i \times (k_1 + 1)}{f_i + k_1 \times (1 - b + b \times \frac{dl}{avgdl})} \times \frac{qf_i \times (k_2 + 1)}{qf_i + k_2}$$

Step 3. Calculate the weight of each word:

Step 4. For each word's score, we do a summation to get the score between the query and the document. The score function is shown below:

$$Score(Q, d) = \sum_{i}^{n} IDF(q_i) \times R(q_i, d)$$

The *BM25* used in the baseline is the simplest *BM25*, using the default parameters $k_1 = 1.8, b = 0.75, k_3 = 8$.

5.1.3 Baseline Results

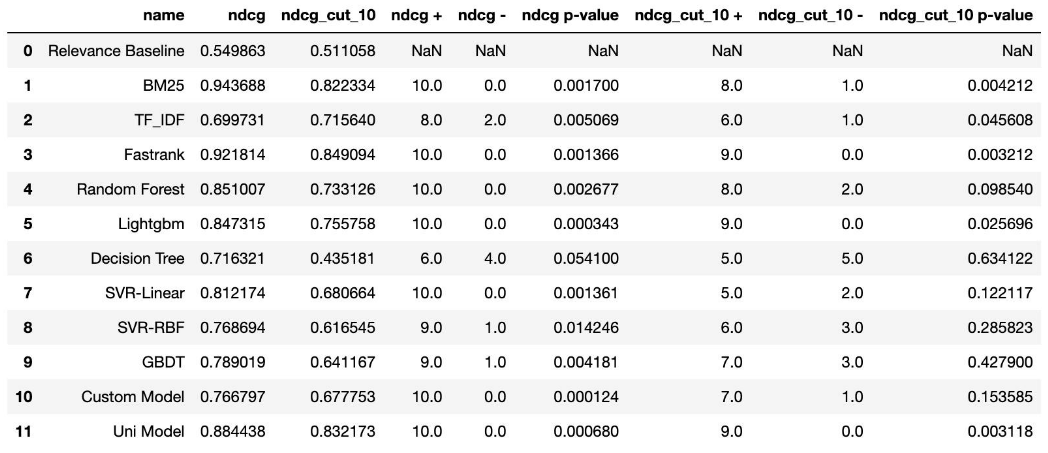
We first choose not to add features and run the two models of baseline directly to see the original baseline results, which are shown in Table 11 below.

Table 11. Baseline Results

Model Name	nDCG@10	nDCG
Relevance Baseline	0.475808	0.538366
BM25	0.802909	0.941683

5.1.4 Model Results

After using the feature pipeline mentioned in **Methods** and adding the methods described in **Methods**, we get the results shown in Figure 5 below.



	name	ndcg	ndcg_cut_10	ndcg +	ndcg -	ndcg p-value	ndcg_cut_10 +	ndcg_cut_10 -	ndcg_cut_10 p-value
0	Relevance Baseline	0.549863	0.511058	NaN	NaN	NaN	NaN	NaN	NaN
1	BM25	0.943688	0.822334	10.0	0.0	0.001700	8.0	1.0	0.004212
2	TF_IDF	0.699731	0.715640	8.0	2.0	0.005069	6.0	1.0	0.045608
3	Fastrank	0.921814	0.849094	10.0	0.0	0.001366	9.0	0.0	0.003212
4	Random Forest	0.851007	0.733126	10.0	0.0	0.002677	8.0	2.0	0.098540
5	Lightgbm	0.847315	0.755758	10.0	0.0	0.000343	9.0	0.0	0.025696
6	Decision Tree	0.716321	0.435181	6.0	4.0	0.054100	5.0	5.0	0.634122
7	SVR-Linear	0.812174	0.680664	10.0	0.0	0.001361	5.0	2.0	0.122117
8	SVR-RBF	0.768694	0.616545	9.0	1.0	0.014246	6.0	3.0	0.285823
9	GBDT	0.789019	0.641167	9.0	1.0	0.004181	7.0	3.0	0.427900
10	Custom Model	0.766797	0.677753	10.0	0.0	0.000124	7.0	1.0	0.153585
11	Uni Model	0.884438	0.832173	10.0	0.0	0.000680	9.0	0.0	0.003118

Figure 5: The above figure shows the results of nDCG and nDCG@10 obtained by running the models on the test set for each model. And we use pyterrier.Experiment() and choose the *Relevance Baseline* as the baseline to get the improvement of each model compared with the baseline. P-value less than 0.05 can be considered a significant impact on the model.

The results in the above figure show that using part of the model after adding features can indeed improve the final model effect. We will discuss and analyze the specific analysis in the **Discussion** section.

5.1.4 Different Feature Pipeline Results

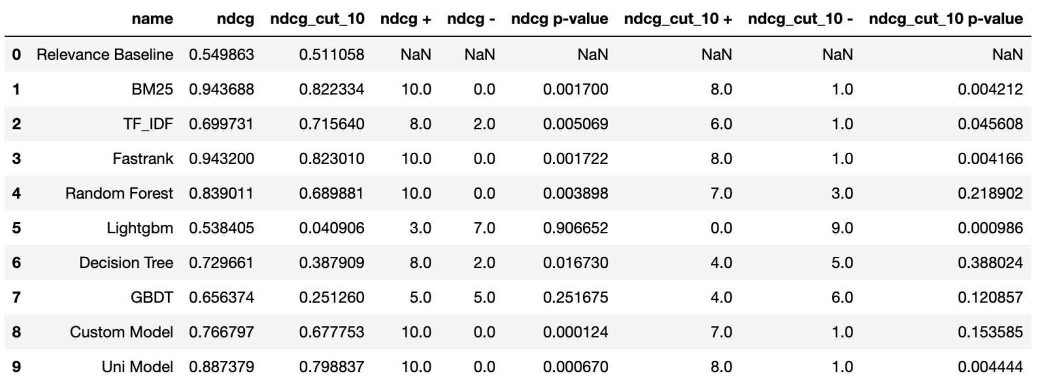
In addition to the features we determined to use in **Methods**, we also tried to add other features to enhance the model effect. We believe that the state in which the nursing home is located, the type of nursing home, the legal name of the nursing home, and the weighted total rating of the nursing home in the three rounds of the health survey should all have some effect on the search engine results. This is because the presentation of search results should be highly correlated with the nursing home region. Also, the type of nursing home determines to some extent the focus of the nursing home business, for example, a for-profit nursing home may have better overall conditions because it charges higher fees, while a non-profit nursing home may charge lower fees, so the type of nursing home may be part of the user's concern. The weighted total score of the three rounds of health surveys is a score obtained by combining the results of multiple rounds, which should be a metric that users will care more about and can better rank nursing homes. The features we chose to include are shown in Table 12 below.

Table 12. Feature Pipeline

Feature Name	Feature Meaning	Realization Method
Provider State	the provider state	define merge_pros(row) apply.query(merge_pros)
Provider Ownership Type	the provider ownership type	define merge_prot(row) apply.query(merge_prot)
Legal Business	the legal business name of provider	define merge_legalname(row)

Name		apply.query(merge_legalname)
Total Weighted Health Survey Score	The weighted total score of the three rounds of health survey scores	define merge_twhss(row) apply.doc_score(merge_twhss)

Bringing the data constructed according to the feature pipeline into the multiple models, we obtained the results shown in Figure 6 below.



	name	ndcg	ndcg_cut_10	ndcg +	ndcg -	ndcg p-value	ndcg_cut_10 +	ndcg_cut_10 -	ndcg_cut_10 p-value
0	Relevance Baseline	0.549863	0.511058	NaN	NaN	NaN	NaN	NaN	NaN
1	BM25	0.943688	0.822334	10.0	0.0	0.001700	8.0	1.0	0.004212
2	TF_IDF	0.699731	0.715640	8.0	2.0	0.005069	6.0	1.0	0.045608
3	Fastrank	0.943200	0.823010	10.0	0.0	0.001722	8.0	1.0	0.004166
4	Random Forest	0.839011	0.689881	10.0	0.0	0.003898	7.0	3.0	0.218902
5	Lightgbm	0.538405	0.040906	3.0	7.0	0.906652	0.0	9.0	0.000986
6	Decision Tree	0.729661	0.387909	8.0	2.0	0.016730	4.0	5.0	0.388024
7	GBDT	0.656374	0.251260	5.0	5.0	0.251675	4.0	6.0	0.120857
8	Custom Model	0.766797	0.677753	10.0	0.0	0.000124	7.0	1.0	0.153585
9	Uni Model	0.887379	0.798837	10.0	0.0	0.000670	8.0	1.0	0.004444

Figure 6: The above figure shows the results of nDCG and nDCG@10 obtained by running the models on the test set for each model. And we use pyterrier.Experiment() and choose the *Relevance Baseline* as the baseline to get the improvement of each model compared with the baseline. P-value less than 0.05 can be considered a significant impact on the model.

The results in the above figure show that using part of the model after adding features made the results worse. The reasons for this will be discussed and analyzed in the **Discussion**.

5.2 Evaluation for Recommender system:

For the evaluation of the recommender system, we will use the five-fold cross-validation method, i.e., we divide the dataset into five parts equally, and use one part at a time as the test set and train the model on the remaining four parts. To determine the performance of the algorithm, we calculate the mean absolute error (MAE) of the prediction data and the test data as a quantitative evaluation rule. The lower the MAE means the lower the model error, the better the model prediction. the MAE is calculated as shown below:

$$MAE = \frac{1}{N}\sum_{i=1}^{N} | y_{pred_i} - y_{true_i} |$$

5.2.1 Naive Baseline

As one of the most common Baselines for comparing the performance of recommendation systems, the Naive Baseline always gives scores based on a statistical basis, assuming that the scores for nursing homes obey a normal distribution so that scores for other nursing homes can be predicted based on the mean and variance of the existing scores.

5.2.2 Baseline Result

We first choose not to add features and run the two models of baseline directly to see the original baseline results, which are shown in Table 13 below.

Table 13. Baseline Result

Model Name	MAE
Naive Baseline	1.657314

5.2.3 Model Results

After using the feature pipeline mentioned in **Methods** and adding the methods described in **Methods**, we get the results shown in Figure 4 below.

Multiple regression models described in **Methods** were used to regress the predictions of Overall Rating and the MAE values corresponding to the predictions of each model were calculated, as shown in Table 14 below.

Table 14. Model Results

Model Name	MAE
Linear Regression	0.352659
Decision Tree Regression	0.011522
Gradient Boosting Regression	0.235785
SVR - Linear	0.354004
SVR - RBF	0.314650
Stacking	0.033098

The results in the table above show that there are still significant differences in the results presented by the different models. We will discuss and analyze the specific analysis in the **Discussion** section.

5.2.4 PCA Results

The model we used in the previous section has 70-dimensional features, which is a high-dimensional model. Therefore, we considered whether we could do some dimensionality reduction work while guaranteeing the effectiveness of the model. So we try to use the PCA method. We use the PCA method of sklearn.decomposition module to achieve dimensionality reduction.

In the PCA method, the selection of dimensionality is the key. In this paper, we try to manually select reasonable dimensions. Therefore, we first plotted the cumulative explained variance linear graph for observation. The specific image is shown in Figure 7 below.

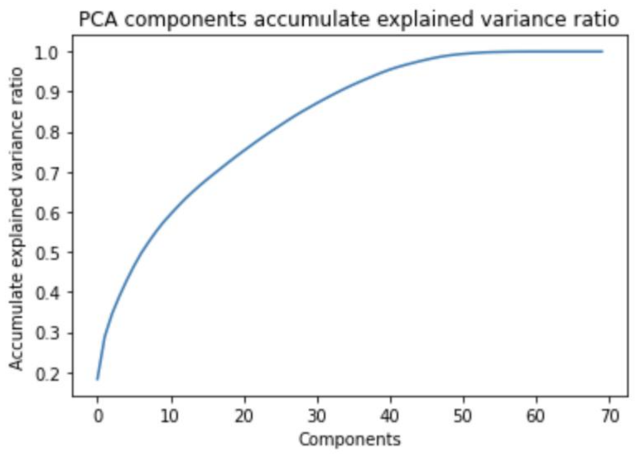


Figure 7: The above image is a trend plot of the cumulative explained variance with the PCA dimension. The horizontal coordinate is the change in the number of dimensions (the range of change is 0-70), and the vertical coordinate is the percentage of cumulative explained variance (the range of change is 0-1).

The above figure shows that the cumulative percentage of the basic explained variance no longer changes when the number of features reaches 50. Using the explained_variance_ratio_.sum() function, we find that the explained variance can reach 0.99245 at n_components, which is not much different from using 70 features. Therefore, we choose n_components = 50 for the PCA() function to achieve dimensionality reduction.

The PCA processed data were divided using the training set and test set division method introduced in the **Methods** method and then brought into the model to obtain the MAE results of each model. The specific model results are shown in Table 15 below.

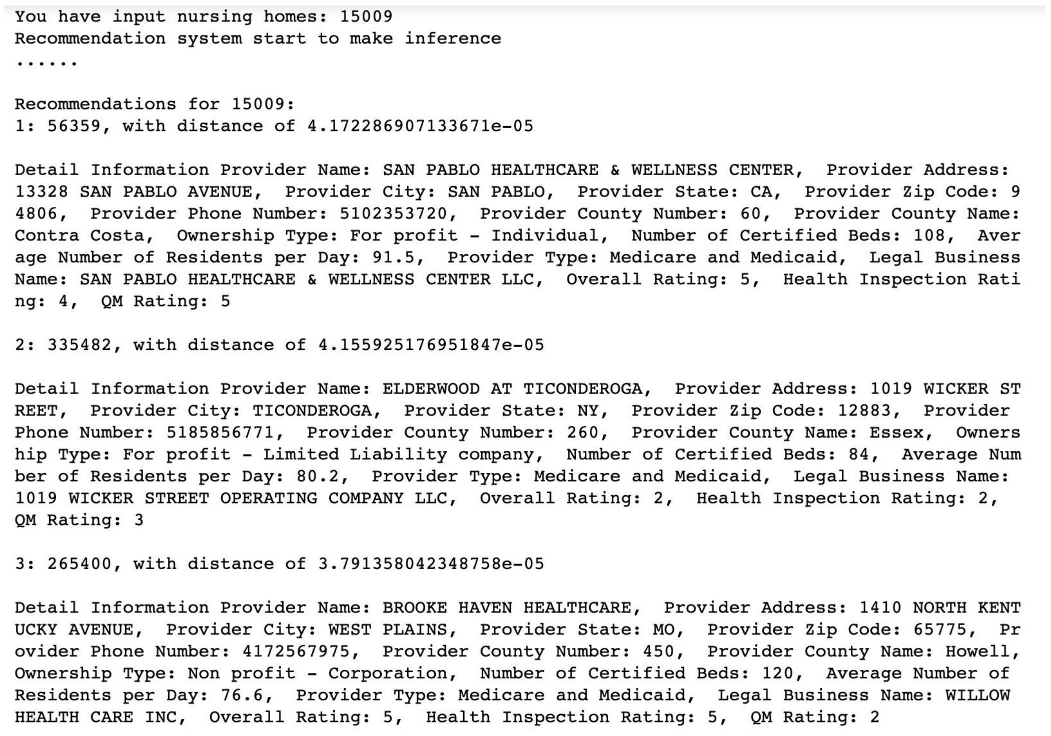
Table 15. Model Results

Model Name	MAE
Linear Regression	1.639706
Decision Tree Regression	0.570721
Gradient Boosting Regression	0.554015
SVR - Linear	0.372789
SVR - RBF	0.337901
Stacking	0.355614

The above table shows that the PCA-processed data are then substituted into each model causing the MAE of each model to increase, i.e., the model results become worse. This change is understandable, because the more comprehensive the features are, the better the prediction results will be. However, it can be found that the impact on Decision Tree Regression is very large, and it changes from the model with the best prediction to the model with very poor prediction. The reasons for this will be discussed and analyzed in the **Discussion**.

5.2.5 Recommendation System Results

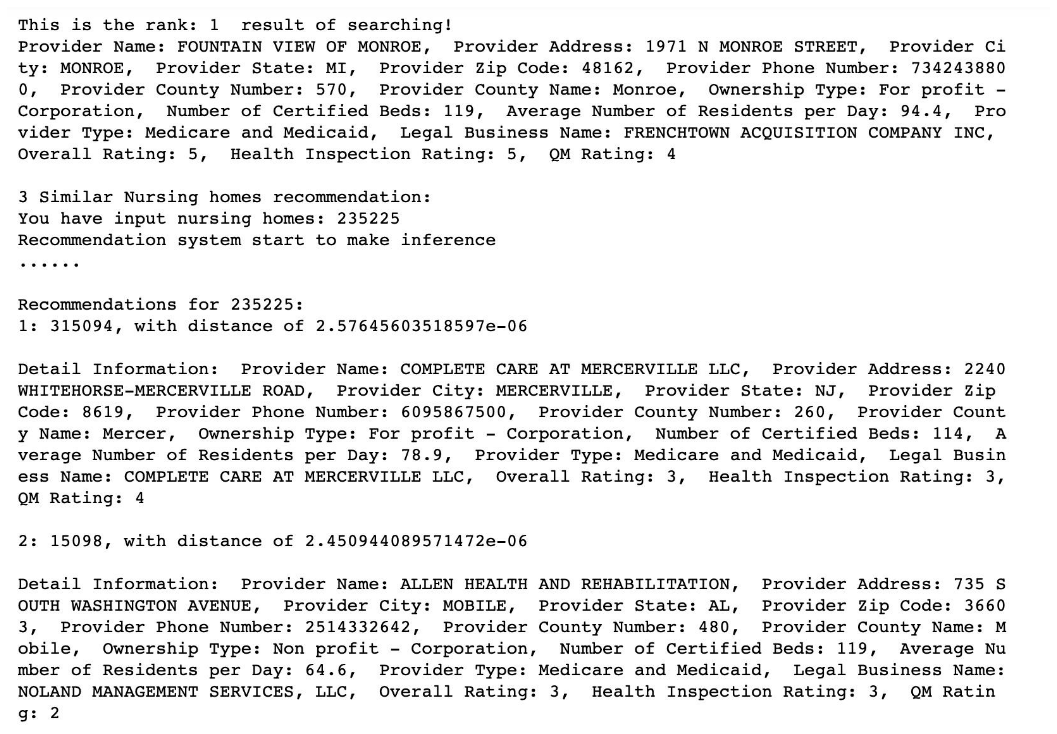
After implementing the method according to Methods, our recommendation system returns the following result(Figure 8).



```
You have input nursing homes: 15009
Recommendation system start to make inference
......

Recommendations for 15009:
1: 56359, with distance of 4.172286907133671e-05

Detail Information Provider Name: SAN PABLO HEALTHCARE & WELLNESS CENTER,  Provider Address:
13328 SAN PABLO AVENUE,  Provider City: SAN PABLO,  Provider State: CA,  Provider Zip Code: 9
4806,  Provider Phone Number: 5102353720,  Provider County Number: 60,  Provider County Name:
Contra Costa,  Ownership Type: For profit - Individual,  Number of Certified Beds: 108,  Aver
age Number of Residents per Day: 91.5,  Provider Type: Medicare and Medicaid,  Legal Business
Name: SAN PABLO HEALTHCARE & WELLNESS CENTER LLC,  Overall Rating: 5,  Health Inspection Rati
ng: 4,  QM Rating: 5

2: 335482, with distance of 4.155925176951847e-05

Detail Information Provider Name: ELDERWOOD AT TICONDEROGA,  Provider Address: 1019 WICKER ST
REET,  Provider City: TICONDEROGA,  Provider State: NY,  Provider Zip Code: 12883,  Provider
Phone Number: 5185856771,  Provider County Number: 260,  Provider County Name: Essex,  Owners
hip Type: For profit - Limited Liability company,  Number of Certified Beds: 84,  Average Num
ber of Residents per Day: 80.2,  Provider Type: Medicare and Medicaid,  Legal Business Name:
1019 WICKER STREET OPERATING COMPANY LLC,  Overall Rating: 2,  Health Inspection Rating: 2,
QM Rating: 3

3: 265400, with distance of 3.791358042348758e-05

Detail Information Provider Name: BROOKE HAVEN HEALTHCARE,  Provider Address: 1410 NORTH KENT
UCKY AVENUE,  Provider City: WEST PLAINS,  Provider State: MO,  Provider Zip Code: 65775,  Pr
ovider Phone Number: 4172567975,  Provider County Number: 450,  Provider County Name: Howell,
Ownership Type: Non profit - Corporation,  Number of Certified Beds: 120,  Average Number of
Residents per Day: 76.6,  Provider Type: Medicare and Medicaid,  Legal Business Name: WILLOW
HEALTH CARE INC,  Overall Rating: 5,  Health Inspection Rating: 5,  QM Rating: 2
```

Figure 8: The above figure is a display of our search engine return results. We entered nursing home docno = 15009, so the returned results are the 3 nursing home docno that are most similar to nursing home 15009 and their details..

5.3 The Integration of Search Engines and Recommendation Systems Results



```
This is the rank: 1  result of searching!
Provider Name: FOUNTAIN VIEW OF MONROE,  Provider Address: 1971 N MONROE STREET,  Provider Ci
ty: MONROE,  Provider State: MI,  Provider Zip Code: 48162,  Provider Phone Number: 734243880
0,  Provider County Number: 570,  Provider County Name: Monroe,  Ownership Type: For profit -
Corporation,  Number of Certified Beds: 119,  Average Number of Residents per Day: 94.4,  Pro
vider Type: Medicare and Medicaid,  Legal Business Name: FRENCHTOWN ACQUISITION COMPANY INC,
Overall Rating: 5,  Health Inspection Rating: 5,  QM Rating: 4

3 Similar Nursing homes recommendation:
You have input nursing homes: 235225
Recommendation system start to make inference
......

Recommendations for 235225:
1: 315094, with distance of 2.57645603518597e-06

Detail Information:  Provider Name: COMPLETE CARE AT MERCERVILLE LLC,  Provider Address: 2240
WHITEHORSE-MERCERVILLE ROAD,  Provider City: MERCERVILLE,  Provider State: NJ,  Provider Zip
Code: 8619,  Provider Phone Number: 6095867500,  Provider County Number: 260,  Provider Count
y Name: Mercer,  Ownership Type: For profit - Corporation,  Number of Certified Beds: 114,  A
verage Number of Residents per Day: 78.9,  Provider Type: Medicare and Medicaid,  Legal Busin
ess Name: COMPLETE CARE AT MERCERVILLE LLC,  Overall Rating: 3,  Health Inspection Rating: 3,
QM Rating: 4

2: 15098, with distance of 2.450944089571472e-06

Detail Information:  Provider Name: ALLEN HEALTH AND REHABILITATION,  Provider Address: 735 S
OUTH WASHINGTON AVENUE,  Provider City: MOBILE,  Provider State: AL,  Provider Zip Code: 3660
3,  Provider Phone Number: 2514332642,  Provider County Number: 480,  Provider County Name: M
obile,  Ownership Type: Non profit - Corporation,  Number of Certified Beds: 119,  Average Nu
mber of Residents per Day: 64.6,  Provider Type: Medicare and Medicaid,  Legal Business Name:
NOLAND MANAGEMENT SERVICES, LLC,  Overall Rating: 3,  Health Inspection Rating: 3,  QM Ratin
g: 2
```

Figure 9: The above figure shows a query example of our final implementation of the integration of a search engine and recommendation system. We choose the query "Monroe street" and return only the first five results, recommending three similar nursing homes for each search. (Only some of the results are captured here, the complete results can be viewed in Jupyter notebook).

6 Discussion

In this section, we will discuss the analysis of the results in **Evaluation and Results**. Again, we have divided it into two sections, the discussion about search engine results will be in section 6.1 and the discussion about recommendation system results will be in section 6.2.

6.1 Discussion for IR System

6.1.1 Original Features Model Results

First of all, regarding the evaluation criteria of both nDCG and nDCG@10 models, we think the results of nDCG@10 are more worthy of reference. This is because, in general, when users perform search result viewing, they will only view some of the results that are ranked at the top, not all of them. Therefore, nDCG@10 can provide better user feedback in this case. Therefore, we will focus more on the analysis of nDCG@10 results when conducting the subsequent discussion.

From the Baseline results in Table 10, the *Relevance Baseline* model performs very poorly in both nDCG@10 and nDCG, but *BM25* already has a better nDCG performance. We believe that, excluding the better effect of the *BM25* model itself, because our annotation is labeled according to the results returned by *BM25*, our manual evaluation may give a higher relevance label to the document, i.e., the subjective factor of annotation is more influential.

Observing the model results presented in Figure 4, we can find that, firstly, adding the feature pipeline we selected in **Methods** does improve the model results, such as *Relevance Baseline* and *BM25* for both nDCG and nDCG@10. This justifies our current choice of features, i.e., the inclusion of scores for different dimensions of nursing homes can improve the search engine results. This is because users prefer to search for better-rated and better-quality nursing home results, so the returned results are optimized with the inclusion of the nursing home rating dimension.

Second, we conducted a cross-sectional comparison of the results of the different model runs. Overall, the better running models are *BM25*, *Fastrank*, and *Uni Model (Custom Model >> Fastrank)*. These three models perform well in both nDCG and nDCG@10, especially the *Fastrank* model. The performance of the rest of the models is not outstanding, but there is still a significant improvement compared to the *Relevance Baseline* model, except for the *Decision Tree* model, the nDCG of all the remaining models is significantly improved compared to the *Relevance Baseline* model (ndcg p-value < 0.05). In the nDCG@10 metric, *BM25*, *TF_IDF*, *Fastrank*, *Lightgbm*, and *Uni Model* all showed significant improvement compared to *Relevance Baseline*.

The reasons for the above model results, consider the following points. First, *BM25* and *Fastrank* are inherently more generalizable and will run better in many datasets. And the data volume of this dataset is large enough, so it can provide a better learning base for these two models to better accomplish the ranking task. Second, we tried to run the *Fastrank* model again after removing the scoring features and found that it showed a more significant decrease in nDCG and nDCG@10. Therefore, it can be assumed that the ranking function itself should be linear for this dataset, and the rating features of different dimensions of nursing homes are the main features for constructing the linear function. Therefore, after adding these features, the coordinate ascent linear list rank function fits better, and also because of the linear function, there is no overfitting, which finally presents the best results of the *Fastrank* model.

The reason for the poor performance of other models we consider may be that there are not enough features extracted and the model dimensionality is low. And the models themselves may not be suitable for ranking, such as *Decision Tree, GBDT*, etc.

Although our *Custom Model* is improved based on the *BM25* model, the effect is far from the *BM25* model. We think there may be a bias in our understanding of the dataset. For example, nursing homes may appear in many documents, but it is still an important term to distinguish different documents. Therefore, after we perform "term normalization", the differentiation between different documents decreases, resulting in poor results of the final model run. In addition, we also consider that it may be because the number of unique terms in the documents is small and there are few duplicate terms in different documents, so the importance of each term is high, and our model leads to a decrease in the differentiation of each term for the documents, which finally presents the poor results of both nDCG and nDCG@10. However, the *Uni Model* we added performs better than *BM25*, and we believe that to some extent the *Fastrank* model works too well to enhance the parts of the *Custom Model* that do not perform well.

6.1.2 Different Feature Pipeline Results

According to Figure 5, we find that the results of each model show different degrees of degradation after adding the new features. In addition to the models with average performance, *Fastrank*, which was performing well, also showed a slight decrease in nDCG@10. However, the overall nDCG showed an improvement, and the trend of the *Uni Model* was similar to that of *Fastrank*. We believe that the reason for this result is that the newly added features can indeed optimize the overall ranking function, but the accuracy decreases for the documents that are already ranked at the top. That is, for the documents that are already at the bottom of the rankings, the distinction is improved. This is probably because the model itself is already better at distinguishing these features, such as "Provider State", and the model is already better at extracting this feature and matching it based on the query. So adding such features will not make the model better.

However, considering that we will only click the top search results in practice, we focus more on the results of nDCG@10. Therefore, in the final model feature selection, we still choose the original feature pipeline introduced in **Methods**.

6.2 Discussion for Recommender system

6.2.1 Model Results

As we can see in table 12, the MAE of the *Naive Baseline* is very large, which proves that the Overall Rating is most likely not normally distributed. And from the model results of table 13, the *Decision Tree Regression* and *Stacking* models are better than the other models, and we think the reasons for this result are: first, the data set itself is large enough and there are most of the labeled features, and according to the introduction of the **Data** section, it can be found that the correlation between the features is not very high (except for individual features), so it fits well with the scenario of using *Decision Tree Regression*. The *Stacking* model itself is an integrated model, which combines the advantages of various models, so, normally, the performance is good. However, we think that *Decision Tree Regression* performs so well, there is a possibility that overfitting has occurred because the data volume is large, *Decision Tree Regression* ignores the correlation between features, and the feature dimension is also high, so we try to reduce the feature dimension. The remaining *Linear Regression, Gradient Boosting Fecision Tree*, and *SVR* models do not work as well as *Decision Tree Regression* and *Stacking*, but they all perform well overall, probably because the dataset itself has a more comprehensive feature dimension, so the results of each model are better.

6.2.2 PCA Model Results

Table 14 shows all the models with different degrees of degradation after PCA. Normally, the performance of the model deteriorates after PCA, because there are fewer features and the overall interpretability of the model deteriorates, which makes it impossible to predict the target variables comprehensively. However, the *Decision*

Tree Regression has dropped significantly, and we believe that this result is not only caused by the poor results of PCA itself but also can confirm that the *Decision Tree Regression* has been overfitted in the previous model results, which is why it has dropped so drastically. The MAE of the *SVR* model also increases to some extent, but the change is not very drastic, which is consistent with the change of the model after PCA. The effect of the *Stacking* model also deteriorates significantly, but the overall effect of the model is still better among all models.

In terms of model stability, we would prefer to choose the post-PCA data for the model to make predictions. Because we hope to achieve the rating of unknown nursing homes through this prediction, although *Decision Tree Regression* will have a very accurate rating in the front, it is difficult for us to determine whether this result is overfitting data to measure the prediction result for new data. So we will finally choose the data after PCA to use the Stacking model for prediction, and after the *Stacking* model adjustment, we found that after removing the *Gradient Boosting Regression model/Decision Tree Regression* of the first layer, the MAE was reduced to about 0.337363, which is the post-PCA the best performing model.

7 Conclusion

Overall, our project is building a search engine and recommendation system for a particular group of nursing homes targeting on the elder population and their relatives who might have concerns about and in need of the nursing-homes. To realize and accomplish the idea, we divided the task into two parts: using models and information retrieval techniques for building search engine, and using models and information retrieval techniques for build recommendation system. To build the search engine, multiple methods are applied and trained on our dataset, including BM25, TF-IDF, Random Forest, Decision Tree, SVR, Gradient Boosting Decision Tree, Fastrank, Lightgbm, and customized models. Through the detailed evaluations, we found that the Fastrank performs best on our dataset among all the models we tried, with the NDCG score of 0.94. For building the recommendation system, the regression model is used for obtaining the prediction results of overall ratings for each nursing-home, then we treat the results as another feature and combined it with other labeled features from the dataset to make the input information as much comprehensive as possible, and then fit K-nearest neighbors (KNN) model to accomplish the recommendations based on the similarities between them and the search results. With all these deliverables, the interactive actions between users and the systems that we built can be achieved, and the practical significance of the project can also be accomplished, which will benefit the population who are in need of querying about the information of this particular group of nursing homes. For future work, it is possible to build a more aesthetically pleasing interaction page, for example, building a webpage, or even an app, to make it easier and more attractive to be used. Additionally, converting features or requesting the update on the dataset to use more recommendation system building techniques to improve the performance of recommendations may be another good idea to consider.

8 Other Things we Tried

8.1 Model Improvement

We tried to optimize the prediction model of the Overall Rating, but the final results presented were not satisfactory. In addition to the methods mentioned in the previous **Methods**, we also tried to introduce *Lasso Regression, Ridge Regression, Elastic Net Regression,* and *Random Forest* models, but we did not document them in the main text because their results performed too poorly compared to the models mentioned in the text.

In addition to this, we constructed a new composite prediction model based on the results of the model MAE. We selected *Linear Regression, Lasso Regression, Ridge Regression, Elastic Net Regression, Decision Tree Regression, Linear SVR,* and *RBF SVR* as the base models, ran the prediction results, and recorded the MAE values of the models. The prediction results of these base models are then assigned weight coefficients according to the percentage of MAE values, and then the prediction results multiplied by these weight coefficients are summed to obtain the final prediction results. We believe that this allows the models with good results to have a larger proportion of the forecasts, and also balances the shortcomings of each model to prevent the appearance of overfitting models. However, the final MAE value for this model ran at 0.517949, which was much less effective than the single model, so we left the model out of the main text.

8.2 Recommendation System

For recommendation system, our original thought is to use item-item Collaborative Filtering as the model for building. However, due to the structure of our dataset, the rating for each nursing-home from users is one on one, that is, there are no multiple users will give the rating to the same nursing-home. Thus, item-item CF may not be a good model to use due to the property of our dataset. To solve this problem, we even tried to dive deeper to our dataset. We found that even if the same user will not give the rating to multiple different nursing-homes, there are features in the dataset called "Rating Cycle 1/2/3", which are indicating that there are three rounds of rating for the same nursing-home from the same user. We tried to treat three rounds of rating for the same nursing-home as the rating scores from three different users for the same nursing home. Though the way works, considering the practical significance, it may not be a good idea to modify the real meaning of the features since the ratings are still from the same user, which means a lot of bias are existing for these rating scores. Additionally, since there are a lot of missing values in these three features, after dealing with missing values (filling them with mean rating scores for all nursing-homes for each round), the performance of CF is poor. Thus, we decided to give up using the CF as the model to build the recommendation system though it's one commonly used model for building the recommendation system.

9 What You Would Have Done Differently or Next

During the implementation process of this project, there are some points that we spent a lot of time on but end up finding that they are not that meaningful to the project deliverables as compared to other parts. Thus, if there is any chance to start the project over, the part that we can improvement is that we may spend less time on comparing models for the prediction of the overall ratings for each nursing homes when building the recommendation system, instead, we may spend more time on trying more recommendation system building techniques and comparing those different recommendation system building models using the existing features. Furthermore, if we would have more time, one way that we might work on is to build a webpage, which can realize a more pleasing interactive process between users and the systems that we built.

Reference

[1] X. Ye, J. Li, Z. Qi and X. He, "Enhancing Retrieval and Ranking Performance for Media Search Engine by Deep Learning," 2016 49th Hawaii International Conference on System Sciences (HICSS), 2016, pp. 1174-1180, doi: 10.1109/HICSS.2016.148.

[2] C. V. Ramana, G. Meghana, M. N. Sai, A. Prasad, and V. Mohanarao, “Building search engine using machine learning technique ,” IJIRT, Sep-2021. [Online]. Available: https://ijirt.org/master/publishedpaper/IJIRT152713_PAPER.pdf. [Accessed: 19-Nov-2022].

[3] E.-H. (S. Han and G. Karypis, “Feature-based recommendation system,” Proceedings of the 14th ACM international conference on Information and knowledge management - CIKM '05, 2005.

[4] M.-H. Kuo, L.-C. Chen, and C.-W. Liang, “Building and evaluating a location-based service recommendation system with a preference adjustment mechanism,” Expert Systems with Applications, vol. 36, no. 2, pp. 3543–3554, 2009.

[5] Z. Zolaktaf, R. Babanezhad, and R. Pottinger, “A generic top-n recommendation framework for trading-off accuracy, Novelty, and coverage,” 2018 IEEE 34th International Conference on Data Engineering (ICDE), 2018.