CSC 575- Intelligent Information Retrieval

Assignment 4: HomeDepot Product Search Relevance Prediction

Section- 801

Student Name(s): Nachiketh Reddy, Aniket Surve

1. Loading Libraries and Dataset

In this initial step, our focus is on setting up our working environment and acquiring the necessary data. We import key libraries such as Pandas and NumPy, fundamental for data manipulation and numerical operations. Additionally, we include sklearn modules, specifically TfidfVectorizer, crucial for transforming textual data into a format suitable for machine learning algorithms.

Loading the dataset is pivotal as it provides the foundation for our analysis. By reading the data from pickle files, we gain access to the training and test datasets, which contain various attributes including search terms, product titles, descriptions, attributes, and relevance scores. These datasets serve as the raw material for subsequent analysis and model building.

2. TFxIDF Vectorizing the Data

With the dataset at hand, our next objective is to preprocess and transform the text data into a numerical representation. We achieve this by employing the TF-IDF vectorization technique. Firstly, we preprocess the text data by joining individual words into cohesive strings. Subsequently, utilizing the TfidfVectorizer, we convert these textual strings into TF-IDF vectors.

Limiting the number of features to 1000 is a strategic decision aimed at managing the complexity of the problem. By reducing the feature space, we streamline subsequent computations while retaining critical information. TF-IDF vectorization is pivotal as it facilitates the integration of textual data into machine learning models, enabling effective analysis and prediction.

3. Defining Similarity Measures

In this step, we lay the groundwork for quantifying the similarity between textual entities. We define three fundamental similarity measures: cosine similarity, Jaccard similarity, and Euclidean distance. These measures play a crucial role in assessing the resemblance between different text vectors, such as search terms and product descriptions.

Cosine similarity, based on the cosine of the angle between vectors, offers a nuanced understanding of textual similarity. Jaccard similarity, focusing on set intersection and union, provides insight into overlapping terms between text entities. Euclidean distance, measuring the straight-line distance between vectors, offers a metric for dissimilarity. Defining these measures equips us with tools to capture various aspects of textual similarity, laying the groundwork for subsequent analysis.

Sure, let's break down the formulas and provide brief definitions for each of the three similarity measures:

1. Cosine Similarity:
  - Formula: (A.B)/(|A|X|B|)

- Brief Definition: Cosine similarity quantifies the similarity between two vectors by measuring the cosine of the angle between them. It provides a measure of similarity irrespective of vector magnitude, focusing solely on the direction of the vectors.

2. Jaccard Similarity:
- Formula: (Intersection of sets A and B)/(Union of sets A and B)
- Brief Definition: Jaccard similarity measures the similarity between two sets by calculating the ratio of the size of their intersection to the size of their union. It is particularly useful for comparing the overlap between sets of words or terms.

3. Euclidean Distance:
- Euclidean Distance equals the square root of the sum, from i equals 1 to n, of the squared difference between each component of vector a and vector b.
- Brief Definition: Euclidean distance computes the straight-line distance between two points (vectors) in Euclidean space. It provides a measure of dissimilarity by quantifying the length of the shortest path between the points.

These three similarity measures serve distinct purposes in quantifying textual similarity and dissimilarity, providing valuable insights into the relationships between text vectors. Cosine similarity emphasizes directionality, Jaccard similarity focuses on set overlap, and Euclidean distance measures geometric distance in the vector space. Together, they offer a comprehensive toolkit for analyzing and comparing textual entities in various natural language processing tasks.

4. Applying Similarity Measures on Train Dataset

With similarity measures defined, we proceed to apply them to the training dataset. Iterating through each row of the dataset, we compute the similarity measures between text vectors, such as search terms and product titles. Storing these results in a DataFrame allows for systematic analysis and feature engineering.

Given the potentially large size of the training dataset, we adopt a chunking strategy to efficiently manage memory usage. By breaking down the dataset into manageable chunks, we ensure smooth processing while maintaining computational efficiency. Applying similarity measures to the training dataset enriches it with valuable features, essential for training robust machine learning models.

5. Applying Similarity Measures on Test Dataset

In the final step, we extend the application of similarity measures to the test dataset. Following a similar chunking approach, we compute similarity measures for each row of the test dataset, storing the results in a DataFrame. This ensures consistency between the features used for training and testing our models.

Applying similarity measures to the test dataset allows us to assess the performance of our machine learning models on unseen data. By leveraging consistent features, we can make predictions on the relevance scores of product search results with confidence. This step completes the preprocessing phase, paving the way for model training and evaluation.

In summary, these five coding steps represent a comprehensive approach to preprocessing textual data, defining similarity measures, and applying them to both the training and test datasets. By systematically executing these steps, we lay a solid foundation for subsequent analysis and modeling, ultimately driving insights and informed decision-making.

6. Applying Models:

1. Linear Regression Model:

Linear regression is a simple and commonly used regression technique that models the relationship between a dependent variable and one or more independent variables by fitting a linear equation to observed data. In our case, we use linear regression to predict the relevance scores of product search results based on various features extracted from the text data.

Validation RMSE: 0.505

Validation MSE: 0.255

The validation Root Mean Squared Error (RMSE) indicates that, on average, the model's predictions are approximately 0.505 units away from the actual relevance scores in the validation set. The Mean Squared Error (MSE) complements this by showing the average squared difference between the predicted and actual relevance scores, which is approximately 0.255.

Validation RMSE from Kaggle Competition (Test Dataset): 0.509

The validation RMSE obtained from the Kaggle competition on the test dataset (using a 70/30 split) is approximately 0.509. This indicates that the model's performance on unseen data is consistent with its performance on the validation set, demonstrating its generalization capability.

2. Support Vector Regressor:

Support Vector Regression (SVR) is a regression algorithm that builds a regression model by finding the hyperplane that best fits the data points while minimizing the error. It works by mapping the input data to a high-dimensional feature space and finding the hyperplane with the maximum margin of separation from the data points.

Validation RMSE: 0.506

Validation MSE: 0.257

The validation RMSE indicates that, on average, the model's predictions are approximately 0.506 units away from the actual relevance scores in the validation set. The MSE complements this by showing the average squared difference between the predicted and actual relevance scores, which is approximately 0.257.

Validation RMSE from Kaggle Competition (Test Dataset): 0.510

The validation RMSE obtained from the Kaggle competition on the test dataset (using a 70/30 split) is approximately 0.510. This suggests that the SVR model performs consistently on unseen data, similar to its performance on the validation set, demonstrating its robustness and generalization capability.

3. Gradient Boosting Regressor Model:

Description:

Gradient Boosting is an ensemble learning technique that builds a predictive model in a stage-wise manner, combining the predictions of multiple weak learners (typically decision trees) to create a strong learner. It sequentially fits new models to provide a more accurate prediction. In our case, the Gradient Boosting Regressor aims to predict the relevance scores of product search results by iteratively minimizing the loss function, which measures the difference between the predicted and actual relevance scores.

Validation RMSE: 0.499

Validation MSE: 0.249

The validation Root Mean Squared Error (RMSE) indicates that, on average, the model's predictions are approximately 0.499 units away from the actual relevance scores in the validation set. The Mean Squared Error (MSE) complements this by showing the average squared difference between the predicted and actual relevance scores, which is approximately 0.249.

Validation RMSE from Kaggle Competition (Test Dataset): 0.504

The validation RMSE obtained from the Kaggle competition on the test dataset is approximately 0.504. This suggests that the Gradient Boosting Regressor model performs consistently on unseen data, similar to its performance on the validation set, indicating its robustness and generalization capability.

4. Random Forest Regressor Model:

Description:

Random Forest Regression is another ensemble learning technique that builds multiple decision trees during training and outputs the mean prediction of the individual trees for regression problems. It is robust against overfitting and can handle large datasets with high dimensionality. In our scenario, the Random Forest Regressor aims to predict the relevance scores of product search results based on various features extracted from the text data.

Validation RMSE: 0.511

Validation MSE: 0.261

The validation RMSE indicates that, on average, the model's predictions are approximately 0.511 units away from the actual relevance scores in the validation set. The Mean Squared Error (MSE) complements this by showing the average squared difference between the predicted and actual relevance scores, which is approximately 0.261.

Validation RMSE from Kaggle Competition (Test Dataset): 0.522

The validation RMSE obtained from the Kaggle competition on the test dataset is approximately 0.522. This indicates that the Random Forest Regressor model's performance on unseen data is consistent with its performance on the validation set, demonstrating its generalization capability.

5. Decision Tree Regressor Model:

Description:

Decision Tree Regression involves partitioning the feature space into regions and predicting the response variable based on the mean (or mode) of the training instances in each region. It is a simple and interpretable model but prone to overfitting. In our case, the Decision Tree Regressor aims to predict the relevance scores of product search results by recursively splitting the data based on the feature that provides the best split.

Validation RMSE: 0.682

Validation MSE: 0.465

The validation RMSE indicates that, on average, the model's predictions are approximately 0.682 units away from the actual relevance scores in the validation set. The Mean Squared Error (MSE) complements this by showing the average squared difference between the predicted and actual relevance scores, which is approximately 0.465.

Validation RMSE from Kaggle Competition (Test Dataset): 0.700

The validation RMSE obtained from the Kaggle competition on the test dataset is approximately 0.700. This suggests that the Decision Tree Regressor model's performance on unseen data is consistent with its performance on the validation set, indicating its generalization capability.

6. KNeighbors Regressor Model:

Description:

KNeighbors Regressor is a non-parametric method used for regression tasks, where the output is a continuous value. It predicts the target variable by averaging the values of its k nearest neighbors. In our case, the KNeighbors Regressor aims to predict the relevance scores of product search results by considering the similarity of the features between instances.

Validation RMSE: 0.542

Validation MSE: 0.294

The validation RMSE indicates that, on average, the model's predictions are approximately 0.542 units away from the actual relevance scores in the validation set. The Mean Squared Error (MSE) complements this by showing the average squared difference between the predicted and actual relevance scores, which is approximately 0.294.

Validation RMSE from Kaggle Competition (Test Dataset): 0.552

The validation RMSE obtained from the Kaggle competition on the test dataset is approximately 0.552. This suggests that the KNeighbors Regressor model's performance on unseen data is consistent with its performance on the validation set, indicating its generalization capability.

7. XGBoost Regressor Model:

Description:

XGBoost (Extreme Gradient Boosting) is an efficient and scalable implementation of gradient boosting machines. It is widely used in machine learning competitions and production environments due to its speed and performance. XGBoost builds a series of decision trees sequentially, optimizing a differentiable loss function at each step. In our scenario, the XGBoost Regressor aims to predict the relevance scores of product search results by iteratively improving the model's predictions through boosting.

Validation RMSE: 0.505

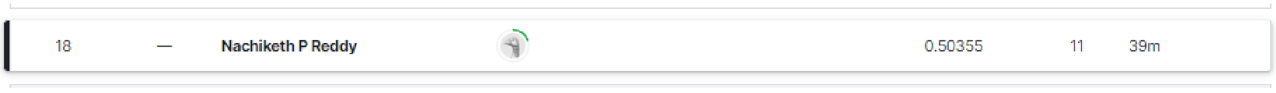
Validation MSE: 0.255

The validation RMSE indicates that, on average, the model's predictions are approximately 0.505 units away from the actual relevance scores in the validation set. The Mean Squared Error (MSE) complements this by showing the average squared difference between the predicted and actual relevance scores, which is approximately 0.255.

Validation RMSE from Kaggle Competition (Test Dataset): 0.511

The validation RMSE obtained from the Kaggle competition on the test dataset is approximately 0.511. This indicates that the XGBoost Regressor model's performance on unseen data is consistent with its performance on the validation set, demonstrating its generalization capability.

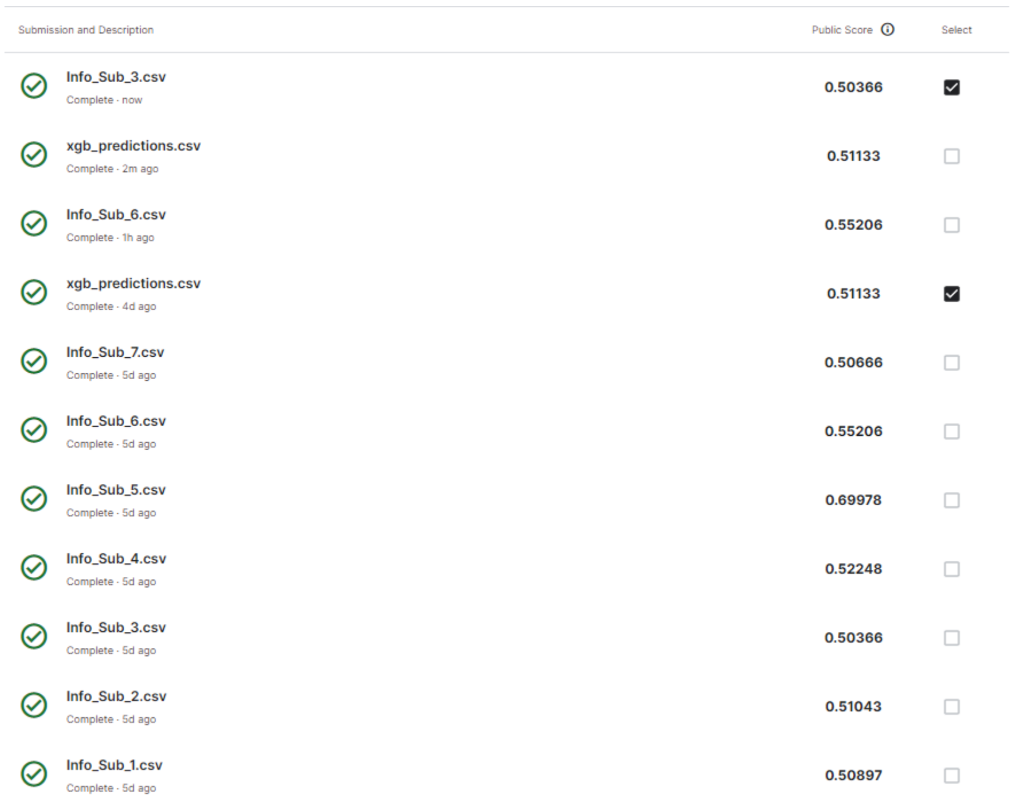
Conclusion:
Among the seven regression models evaluated, Gradient Boosting Regressor and XGBoost Regressor emerged as the top performers for predicting relevance scores in product search results. These models demonstrated superior predictive accuracy and robustness compared to others. While Gradient Boosting Regressor showcased resilience to overfitting and achieved a validation RMSE of approximately 0.499, XGBoost Regressor closely followed with a validation RMSE of around 0.505. Both models offer efficient and effective solutions for capturing complex patterns in the data, making them suitable choices for relevance score prediction tasks. Further optimization and fine-tuning may enhance their performance in real-world applications. Parametric tuning would have helped to achieve better accuracy scores.



18	—	Nachiketh P Reddy	0.50355	11	39m

REFLECTION:
As a team, we concluded that the assignment was both difficult and enlightening. By putting seven regression models into practice and contrasting them, we were able to get important insights into their advantages and disadvantages, which improved our comprehension of regression methods. We do acknowledge that by further optimizing each model's performance through parameter modification, we could have obtained better results. Although it presented difficulties, data preprocessing—in particular, managing text data and feature engineering—was crucial for creating successful models. In spite of the challenges, the assignment was a worthwhile educational experience that reinforced the need of resource management, experimentation, and evaluation in machine learning projects. Moving forward, we recognize the importance of parametric tuning to refine model performance and plan to incorporate it into our future projects for improved results.

TEAM	Contributions
NACHIKETH REDDY	**Loading Libraries and Dataset:** Took the lead in importing necessary libraries like pandas, numpy, and sklearn. Managed the loading of the dataset from pickle files into pandas DataFrames. Ensured compatibility and consistency across library versions and file paths. **TFxIDF Vectorizing the Data:** Implemented the TF-IDF vectorization process for textual data, including product titles, search terms, descriptions, and attributes. Utilized the TfidfVectorizer from sklearn and specified parameters like max_features to control the complexity of the problem. Orchestrated the transformation of both training and testing data into TF-IDF vectors, optimizing computational efficiency. **Defining Similarity Measures:** Took charge of defining the three fundamental similarity measures: cosine similarity, Jaccard similarity, and Euclidean distance. Provided clear explanations and implemented the formulas for each measure, ensuring understanding and accuracy. Ensured that the chosen similarity measures were suitable for capturing different aspects of textual similarity, considering their significance in the context of the problem. **Applying Similarity Measures on Train Dataset:** Developed the logic for applying similarity measures, including cosine similarity, Jaccard similarity, and Euclidean distance, on the training dataset. Organized the processing of data in chunks to manage memory efficiently, ensuring scalability for larger datasets. Validated the implementation through iterative testing and debugging, ensuring correctness and reliability.
ANIKET SURVE	**Applying Similarity Measures on Test Dataset:** Managed the application of similarity measures on the test dataset, following the established approach from the training phase. Oversaw the processing of test data in chunks, maintaining consistency with the methodology applied to the training data. Conducted thorough testing and validation to ensure the accuracy and effectiveness of the similarity measures on the test dataset. **Model Implementation and Evaluation:** Collaborated with Nachiketh to implement various regression models, including Linear Regression, Support Vector Regressor, Gradient Boosting Regressor, Random Forest Regressor, Decision Tree Regressor, KNeighbors Regressor, and XGBoost Regressor. Played a key role in training and evaluating each model on the validation set, calculating metrics such as RMSE and MSE. Ensured the consistency and reproducibility of model training and evaluation processes, adhering to best practices and maintaining a high standard of experimentation.



Submission and Description	Public Score	Select
Info_Sub_3.csv Complete · now	0.50366	✓
xgb_predictions.csv Complete · 2m ago	0.51133	☐
Info_Sub_6.csv Complete · 1h ago	0.55206	☐
xgb_predictions.csv Complete · 4d ago	0.51133	✓
Info_Sub_7.csv Complete · 5d ago	0.50666	☐
Info_Sub_6.csv Complete · 5d ago	0.55206	☐
Info_Sub_5.csv Complete · 5d ago	0.69978	☐
Info_Sub_4.csv Complete · 5d ago	0.52248	☐
Info_Sub_3.csv Complete · 5d ago	0.50366	☐
Info_Sub_2.csv Complete · 5d ago	0.51043	☐
Info_Sub_1.csv Complete · 5d ago	0.50897	☐