➢ Abstract:

Efficient retrieval and ranking of relevant information are crucial in today's information-driven era. This coursework addresses the task of predicting relevance scores between user queries and product results using Natural Language Processing (NLP) and Text Mining techniques. The methodology involves data preprocessing steps, such as handling missing values, removing special characters, lowercase conversion, stopwords removal, lemmatization, and addressing misspelled words. Word embeddings with the Word2Vec algorithm are employed for feature extraction, capturing semantic relationships in the textual data. Several regression models, including Linear Regression, Random Forest, XGBoost, GradientBoostingRegressor, Lasso Regression, and Ridge Regression, are evaluated using the Root-Mean-Square Error (RMSE) metric. The experiment results reveal that the XGBoost Regressor, with a learning rate of 0.1, outperforms other models, achieving a minimum RMSE of 0.5275612556308176. This highlights the effectiveness of XGBoost in capturing complex relationships between textual data and relevance scores.

➢ Introduction:

In today's information-driven era, efficient retrieval and ranking of relevant information have become paramount. In many applications, such as search engines and recommendation systems, accurately predicting the relevance of search results to user queries plays a crucial role in delivering satisfactory user experiences. The task at hand is to predict the relevance score between a query and a specific product result, where higher scores indicate greater relevance.

The objective of this coursework is to develop a model capable of predicting the relevance scores for unknown query-result combinations. This prediction task falls within the domain of Natural Language Processing (NLP) and Text Mining, where techniques are applied to understand and extract meaning from textual data.

To evaluate the effectiveness of our relevance prediction model, i employ the Root-Mean-Square Error (RMSE) as the evaluation metric. This metric measures the average difference between the predicted relevance scores and the ground truth scores provided in the dataset. By minimizing the RMSE, I aim to develop a model that accurately captures the relevance between queries and results.

➢ Dataset Description:

The project’s data contains three parts:

-train.csv:

This dataset has 5 columns: id, product_uid, product_title, search_term and relevance.

-product_descriptions.csv:

This dataset has 2 columns: product_uid and product_description

-attributes.csv:

This dataset has 3 columns: product_uid, name and value

➢ Methodology:

-Data Preprocessing:

To ensure the quality and suitability of the data for relevance prediction, the following preprocessing steps were applied:

1. Handling Missing Values: Any missing values in the dataset, such as null or empty fields, were identified and appropriately handled. This involved either imputing missing values based on specific strategies or removing instances with missing values, depending on the nature of the data.

2. Removing Special Characters, Punctuation Marks, and Non-Alphanumeric Symbols: All special characters, punctuation marks, and non-alphanumeric symbols were removed from the textual data. This step eliminated noise and unnecessary distractions, allowing the model to focus on the essential textual content.

3. Lowercasing: All text data was converted to lowercase. This step normalized the text by treating uppercase and lowercase letters as the same, ensuring consistency in subsequent processing steps. For example, "Relevance" and "relevance" would be treated as identical words.

4. Remove Stopwords: Stop words, such as "and," "the," and "is," were removed from the text. These words, which carry little semantic meaning, were excluded to reduce noise and improve the efficiency of subsequent analyses.

5. Lemmatization: The process of lemmatization was applied to reduce words to their base or root form. This step involved converting words to their canonical form, which helps consolidate different inflections of a word. For instance, "running," "runs," and "ran" would be lemmatized to "run".

6. Handling Misspelled Words: Specifically, misspelled words were addressed in the search terms. Given that search queries are more prone to misspellings, a dedicated step was taken to correct or replace misspelled words within the search term. This step aimed to enhance the relevance prediction process by mitigating the potential impact of misspellings on the model's performance.

By implementing these preprocessing steps, the textual data was transformed into a clean and standardized format, ready for relevance prediction. These steps eliminated noise, handled missing values, addressed variations in word forms, and enhanced the overall quality of the dataset. The preprocessed data served as a foundation for subsequent modeling and analysis, enabling accurate predictions of relevance scores.

-Feature Extraction:

To capture the rich semantic information present in the textual data, word embeddings were employed to represent the text as dense vector representations. Specifically, we utilized the Word2Vec algorithm for this purpose.

Word2Vec is a popular technique in Natural Language Processing (NLP) that learns word embeddings by leveraging the distributional patterns of words within a large corpus. It maps each word in the vocabulary to a high-dimensional vector, wherein the vector positions encode semantic relationships between words. By leveraging this approach, we aimed to capture contextual and semantic similarities between words, enabling the model to learn meaningful representations of the textual data.

By applying word embeddings using the Word2Vec algorithm, i aimed to capture the semantic relationships between words and represent the textual data as dense vectors. This allowed the model to effectively learn patterns and similarities between queries and results, facilitating accurate relevance score predictions.

-Models:

To predict the relevance scores for the unknown query-result combinations, I experimented with various regression models known for their effectiveness in handling continuous target variables. The following models were implemented:

1. Linear Regression: Linear regression is a classical statistical model that assumes a linear relationship between the input features and the target variable. I utilized linear regression as a baseline model to establish a benchmark for performance comparison.

2. Random Forest: Random Forest is an ensemble learning method that combines multiple decision trees to make predictions. I employed the Random Forest algorithm with different values for the number of estimators (10, 50, 100) to evaluate its performance and find the optimal configuration.

3. XGBoost (Extreme Gradient Boosting): XGBoost is a powerful gradient boosting framework that excels in handling structured data. I utilized the XGBRegressor, a variant of XGBoost for regression tasks, and experimented with different learning rates (0.1, 0.01, 0.001) to find the optimal balance between learning speed and accuracy.

4. Gradient Boosting Regressor: Similar to XGBoost, Gradient Boosting Regressor is another ensemble learning technique that combines multiple weak prediction models, typically decision trees, to create a strong predictive model. I explored the performance of GradientBoostingRegressor with different values for the number of estimators (10, 50, 100).

5. Lasso Regression: Lasso regression is a linear regression variant that incorporates L1 regularization to shrink the coefficients of less important features, promoting feature selection. I utilized Lasso regression with different values for the regularization parameter (0.1, 0.01, 0.001) to determine the optimal level of regularization.

6. Ridge Regression: Ridge regression is another linear regression variant that applies L2 regularization to prevent overfitting and reduce the impact of multicollinearity among the input features. I experimented with different values for the regularization parameter (0.1, 0.01, 0.001) in Ridge regression.

➢ Experiment and Results:

In this section, i present the experimental setup and the results obtained from evaluating different regression models for relevance score prediction. I utilized the preprocessed textual data and the corresponding relevance scores to train and evaluate the models. The performance of each model was assessed using the Root-Mean-Square Error (RMSE) metric.

-Model Evaluation and Hyperparameter Tuning:

To identify the optimal hyperparameters for each regression model, i employed GridSearch, a popular technique for hyperparameter optimization. GridSearch allows systematic exploration of different hyperparameter combinations, enabling me to find the best parameters for each model based on the evaluation metric.

```
Applied Model: LinearRegression, Best Parameters: {}, Root Mean Squared Error: 0.5325185756054259
Applied Model: RandomForestRegressor, Best Parameters: {'n_estimators': 100}, Root Mean Squared Error: 0.5278339231738426
Applied Model: XGBRegressor, Best Parameters: {'learning_rate': 0.1}, Root Mean Squared Error: 0.5275612556308176
Applied Model: GradientBoostingRegressor, Best Parameters: {'n_estimators': 100}, Root Mean Squared Error: 0.5281457443993135
Applied Model: Lasso, Best Parameters: {'alpha': 0.001}, Root Mean Squared Error: 0.5326782359231418
Applied Model: Ridge, Best Parameters: {'alpha': 0.1}, Root Mean Squared Error: 0.5319226877410366
```

1. Random Forest: I utilized GridSearch to evaluate different values for the number of estimators ('n_estimators') in the range [10, 50, 100] for the Random Forest model. Through this process, i determined that the optimal parameter setting for Random Forest was 'n_estimators': 100.

2. XGBoost: GridSearch was also employed to explore different learning rates ('learning_rate') for the XGBoost model. I considered the values [0.1, 0.01, 0.001] and discovered that the best-performing XGBoost model had a learning rate of 0.1.

3. GradientBoostingRegressor: Similar to XGBoost, i used GridSearch to evaluate various values for the number of estimators ('n_estimators') within the range [10, 50, 100] for the GradientBoostingRegressor model. My analysis indicated that the optimal parameter for this model was 'n_estimators': 100.

4. Lasso and Ridge Regression: For Lasso and Ridge regression, GridSearch was employed to explore different values for the regularization parameter ('alpha'). I considered the ranges [0.1, 0.01, 0.001] for both models. Based on the results, I found that the best parameter setting for Lasso was 'alpha': 0.001, while for Ridge regression, 'alpha': 0.1 yielded the best performance.

-Model Performance:

After evaluating each model with the optimal parameters obtained through GridSearch, i compared their performances based on the RMSE metric. Among the models tested, the XGBoost Regressor demonstrated the best performance, achieving a minimum RMSE of 0.5275612556308176. This indicates that the XGBoost model's predictions were, on average, the closest to the ground truth relevance scores compared to other models.

```
# Print the best model and its RMSE
print("Best Model: {}, Lowest Root Mean Squared Error: {}".format(type(best_model).__name__, best_rmse))
```

```
Best Model: XGBRegressor, Lowest Root Mean Squared Error: 0.5275612556308176
```

The results highlight the effectiveness of XGBoost in capturing the complex relationships between the textual data and the relevance scores, surpassing other regression models in relevance score prediction. The optimal configuration of XGBoost with a learning rate of 0.1, identified through GridSearch, contributed to its superior performance.

These findings demonstrate the importance of careful model selection, hyperparameter tuning, and the use of GridSearch to achieve the best performance in relevance score prediction. The XGBoost Regressor, with its optimal configuration, serves as a powerful tool for accurately predicting the relevance scores between queries and results in our context.

➢ Student:

Name: Vasileios

Surname: Douvikas

ID: 3308220006