Automatic Essay Score

Python Project Lab - Group 5 - A23

Names:

Luciano Costa

Anzor Kerefov

Note: We used ChatGPT 4 as support in the development of our project.

The development steps and results

Data Validation and Reading

Action: Checked for the existence of a specific excel file and loaded the data into a Pandas DataFrame.

Results: Successful data reading provided an initial overview, descriptive statistics, identified missing values, etc.

Text Cleaning and Preparation

Action: Cleaned the 'essay' column by removing strings containing '@' and excessive spaces, followed by saving it in a new excel file.

Results: Created a dataset with cleaned essays, ready for subsequent analysis.

Data Integration

Action: Merged the original dataset with the cleaned essays, reorganizing columns to facilitate comparison and analysis.

Results: A combined dataset with original and cleaned essays was saved for future reference.

Natural Language Processing

Action: Tokenized the cleaned essays using the pre-trained spaCy model and saved the tokens in a new excel file.

Results: Enriched the DataFrame with a new column containing the extracted tokens from the essays.

Score Normalization

Action: Normalized the essay scores based on predefined score ranges for each essay set.

Results: Generated normalized scores within a 0 to 1 range, enabling fair comparisons between essays from different sets.

Word Count, Lexical Diversity and Average Sentence Length

Action: Calculated the word count, lexical diversity, and average sentence length for each essay, adding these metrics as new columns.

Results: Updated the dataset with comprehensive quantitative information about the essays, including details on overall length, vocabulary variance, and sentence structure.

Grammatical Complexity Analysis

Action: Calculated metrics such as the average number of nouns, verbs, and adjectives per sentence.

Results: Enhanced the dataset with new columns reflecting grammatical complexity, providing deeper insights into the linguistic structure of essays. Saved the enriched dataset for further analysis and model training.

Spelling Error Analysis

Action: Used a spell checker to identify and count spelling errors in each essay.

Results: Included a new column indicating the number of spelling errors, which can be indicative of the essay's quality.

Visualization of Error Distribution

Action: Created histograms to visualize the distribution of spelling errors in the essays.

Results: The majority of the bars are concentrated at the beginning of the x-axis, indicating a smaller number of errors, it suggests that most of the essays contain few spelling mistakes, with a concentration up to 12 errors.

Data Consolidation

Action: Extracted and combined columns of interest from various files into a single DataFrame.

Results: Created a final dataset containing a wide range of features for analysis.

Machine Learning Model Evaluation

Action: Used a Random Forest Regressor to predict normalized scores based on various essay features. It trains the model on the training set, ranks the importance of features, and visualizes them in a bar chart.

Results: 'Word_count' emerged as the predominant feature, although with moderate importance (0.46 out of 1), while other features exhibited relatively low significance, reflecting in varied accuracy levels across essay groups.

Interaction Feature Calculation

Action: Developed new interaction features like lexical density, noun-verb ratio, adjective-noun ratio, and spelling error density to delve deeper into textual characteristics.

Results: Enhanced the analytical depth of the dataset by integrating these interaction features, providing a more nuanced understanding of the essays beyond basic metrics.

Correlation Analysis

Action: Computed Pearson correlation coefficients between essay features and the final normalized scores.

Results: The analysis reveals a moderate positive correlation for lexical density and word count, but overall, most features show weak correlations with the final scores, suggesting that the key determinants of essay quality might not be captured by these metrics alone.

Feature Analysis Across Essay Sets

Action: Analyzed essay features across different essay sets, calculating their mean and standard deviation, and conducted an ANOVA test for each feature to examine variability between groups.

Results: Identified significant differences in essay characteristics across sets, as evidenced by varying means, standard deviations, and ANOVA test outcomes, indicating distinct linguistic and structural profiles per essay set.

Impact of Essay Characteristics on Scoring

Action: The script processes and arranges the correlation coefficients between a range of essay characteristics, including interactive features, and the essays' normalized scores across different essay sets. It subsequently presents these correlations for each group, illustrating the varying impact of distinct features on the final scores across various essay categories.

Results: The correlation analysis suggests that the quality of an essay, as reflected in the normalized final score, is strongly associated with its length, vocabulary richness, and expressiveness, while spelling accuracy and grammatical complexity have a lesser impact. Essays that are well-developed, with good lexical density and word count, tend to be rated higher, whereas excessive lexical diversity and the presence of spelling errors are not as penalized as one might expect.

A Linear Regression Analysis

Action: This script performs linear regression analysis on essay data grouped by essay set, using statsmodels. It uses various writing characteristics as independent variables to predict the normalized scores of essays. The regression results for each essay group are then compiled and displayed, highlighting the relationships between essay features and scores.

Results: The regression analysis for essay groups reveals varied model fits, highlighted by R-squared values between 0.506 and 0.726. Lexical density and word count significantly influence scores in many groups, while the effects of features like spelling errors and sentence length differ by group. Features such as wordiness show varied significance, hinting at a context-dependent impact on scores. High condition numbers in some models hint at potential multicollinearity, possibly impacting coefficient reliability.

Predictive Analysis of Essay Scores by group

Action: This code conducts linear regression analysis for specified essay groups from a dataset, utilizing statsmodels. For each group, it segregates features and the normalized score, splits the data into training and testing sets, fits the model on the training set, and predicts scores on the test set. It calculates and displays the Mean Squared Error (MSE) and R-squared (R^2) values to assess model performance for each group.

Results: The linear regression results across essay groups show significant variations in both Mean Squared Error (MSE) and R-squared (R^2), indicating differences in prediction accuracy and the model's ability to explain the variance in normalized scores. Group 1 exhibits the best performance with an R^2 of 0.740, indicating a strong predictive capability, while Group 3 has the weakest performance with an R^2 of 0.446. This reflects the varying complexity of essay features and their relationship with scores across different datasets.

GBM Model: Optimization & Evaluation Approach

Action: This code optimized hyperparameters for a Gradient Boosting Machine (GBM) model via GridSearchCV with 5-fold cross-validation, aiming to identify the best parameter combination based on negative mean squared error. Post-optimization, the model's performance is evaluated on a test set, calculating MSE and R^2 to quantify its predictive accuracy.

Results: The Gradient Boosting Machine (GBM) model was fine-tuned with optimized hyperparameters, including a learning rate of 0.1, max depth of 3, min samples leaf of 2, min samples split of 2, and 100 n_estimators. These parameters aim to strike a balance between model complexity and overfitting, ensuring efficient learning.

On the test set, the GBM model yielded a Mean Squared Error (MSE) of 0.0120, indicating a moderate level of prediction error. The R-squared (R^2) value of 0.4739 suggests that less than half of the variability in essay scores is explained by the model, indicating potential for improvement.

Neural Network Progress: 5-Epoch Essay Scoring

Action: This code is designed to train a neural network model with PyTorch to predict normalized essay scores from their tokenized text representations, using a 5-epoch training process. It tokenizes the essay texts, creates a vocabulary for token-to-index mapping, and splits the data into training and testing sets. The model, featuring an embedding layer followed by a linear layer, aims to minimize Mean Squared Error (MSE) over these 5 epochs to improve its predictive accuracy for essay scores.

Results: Across 5 training epochs, the model demonstrated an initial decrease in loss from 0.0635 to 0.0412, showing early signs of effective learning. Despite experiencing fluctuations in the subsequent epochs, with slight increases in loss, it ultimately improved, closing at a loss of 0.0507 in the final epoch. This end result, especially the reduction to 0.0507, signifies a commendable achievement in predictive accuracy, suggesting that the model has effectively adapted to better predict essay scores. The final loss indicates a positive outcome, reflecting substantial progress in the model's learning and prediction capabilities over the training period.

Essay Group Scoring: Neural Network Efficacy Comparison

Action: For each essay group, this code trains a neural network model using PyTorch, splits data into training and testing sets, and evaluates each model's performance with Cohen's Kappa Score after 5 epochs. DataLoaders facilitate batch processing, and optimization minimizes MSE loss. Finally, models and their kappa scores are stored for performance analysis, highlighting the tailored approach to predicting essay scores across different groups.

Results: The training outcomes for neural network models across eight essay groups showed varying loss trends and Cohen's Kappa Scores, indicating differing model performances. Group 1 showed a significant improvement in loss reduction across epochs, culminating in a Kappa Score of 0.622, suggesting strong agreement in predictions. Conversely, Group 8 displayed minimal loss but a Kappa Score of 0.0, indicating no agreement beyond chance. Other groups exhibited mixed results, with Kappa Scores ranging from moderate (Group 3: 0.567) to lower (Group 6: 0.336), reflecting the models' varying success in accurately predicting essay scores.

Essay Score Prediction with Random Forest across eight sets

Action: This code employs a Random Forest Regressor to predict normalized essay scores, diverging from previous neural network approaches. It uses CountVectorizer to transform essay texts into term frequency vectors, fitting the model on these features. Predefined parameters optimize the Random Forest, and performance is evaluated using Mean Squared Error (MSE) for each essay group. This method highlights a classical machine learning technique applied to natural language processing tasks for essay scoring.

Results: The results from using a Random Forest Regressor to predict essay scores reveal a range of Mean Squared Errors (MSE) across eight essay sets, indicating varying prediction accuracies. Essay Set 8 showed the highest accuracy with the lowest MSE of 0.0048, while Essay Set 3 had the highest MSE at 0.0428, suggesting it was the most challenging set for the model to predict accurately. The lower the MSE, the closer the predicted scores are to the actual scores, showcasing the model's effectiveness. These outcomes demonstrate the Random Forest's varying capacity to handle the unique characteristics of each essay set.

SVR Essay Score Prediction across eight sets

Action: This code applies a Support Vector Regression (SVR) model to predict normalized essay scores. It employs CountVectorizer to transform essay texts into term frequency vectors for modeling. The SVR is configured with specific parameters (C=1.0, epsilon=0.2) and trained on these text features. After training, the model's performance is evaluated across eight essay sets by calculating the Mean Squared Error (MSE) for each set, offering insights into the predictive accuracy of SVR on diverse essay data.

Results: The MSE results from using SVR for essay score prediction across eight sets reveal a spectrum of accuracies. Sets 1 and 8 showcase the highest precision with the lowest MSEs, indicating strong model performance. In contrast, set 3 displays the highest MSE, suggesting significant prediction challenges. The other sets (2, 4, 5, 6, and 7) exhibit moderate MSEs, pointing towards reasonable but improvable model effectiveness. These outcomes highlight SVR's varied ability to predict essay scores, with notable successes in some sets and challenges in others.

MSE Comparison: Chart Visualizes Random Forest vs. SVR

Action: A bar chart was generated comparing the Mean Squared Error (MSE) results between two previous models, Random Forest and Support Vector Regression (SVR), for predicting essay scores across eight different sets. For each essay set, two bars are plotted side by side representing the MSE achieved by each model, allowing for a direct visual comparison of performance. The chart is enhanced with detailed annotations of specific MSE values, making it easier to interpret the results and highlighting the differences in the efficacy of the models across the essay sets.

Results: The bar chart presents a close comparison of Mean Squared Error (MSE) values for the Random Forest and Support Vector Regression (SVR) models across eight essay sets. Despite some variations, the performance of both models is quite similar, with SVR slightly outperforming Random Forest in most sets, particularly in Set 8, which has the lowest MSEs for both models. The chart demonstrates that while there are some differences in MSE values between the two models, they are generally marginal, suggesting that both models are comparably effective for this prediction task across the various essay sets.

Refining LSTM Essay Scoring with Pre-trained Word2Vec

Action: This code implements an essay scoring model using a Recurrent Neural Network (LSTM). It loads a pre-trained Word2Vec model for word representations, creates a dataset with essays and their normalized scores, and defines an LSTM model to perform the essay scoring. The model is trained for 20 epochs using the Mean Squared Error (MSELoss) loss function and Adam optimizer. During training, the model is updated based on batch predictions and the actual scores of the essays. The training progress is printed at each epoch, displaying the average loss over the training dataset.

Results: The provided results show the loss decreasing consistently over the training epochs, indicating improvement in the model's performance. The initial loss of 0.0694 decreases gradually, reaching 0.0553 by the 20th epoch. This reduction suggests that the model is effectively learning the patterns within the data and refining its predictions accordingly. The decreasing trend of the loss demonstrates the effectiveness of the training process in optimizing the model's parameters to minimize prediction errors.

Enhancing LSTM Essay Scoring: Changes and Performance Evaluation

Action: Some alterations were made to the code to evaluate the performance of the essay scoring model. We reduced the word embedding dimension from 300 to 50 while maintaining the same model architecture as the previous one. Additionally, we decreased the hidden vector size from 64 to 32. We also switched the optimizer from Adam to AdamW and increased the batch size from 64 to 128. Moreover, the number of training epochs was increased from 20 to 50. These changes were made to observe how different configurations affect the model's ability to learn and generalize from the training data.

Results: The training outcome depicts a declining loss throughout the epochs, suggesting the model is progressively learning from the data. The average loss decreases from around 0.079 initially to about 0.055 by the end of the 50 epochs. However, it is noteworthy that after a certain point, the loss appears to have plateaued around 0.055 and did not decrease significantly below this value.