DSTI School of Engineering

MACHINE LEARNING PROJECT

ESSAY QUALITY PREDICTION

Authors

- NEMKENANG TIOKENG HORLAIN
- SOH TCHINDE DENIS DUCLAIR
- MEKONTCHOU M AURELIE
- PASCALCELESTRE DISSIVOULOUD
- ALLEN KENNETH DAVY MEBARA TSANGA

Introduction

In today's world, with everything being digital and computerized, we see language more like science that can be studied. Therefore, creating methods for feeding language to inanimate objects like computers has become crucial. NLP aids in this process.

Common examples are spellcheckers which find syntax mistakes that you would have missed on your own, our search engine figure out what you were going to write just on the first few letters. Another one is when our computers automatically detect some emails as Spams. In this case our email service understands the content of the email then determines whether to designate it as spam or not.

Natural language processing, or NLP, is concerned with how machines can comprehend, manipulate, and produce natural language. NLP is therefore genuinely at the nexus of linguistics and computer science. It centers on the capacity of robots to communicate directly with people.

As part of our Machine Learning coarse, we must undertake a course project to develop an end-to-end pipeline that process' an essay and outputs a grade describing the level of English proficiency. In short, train a model that predicts an essay’s rating.

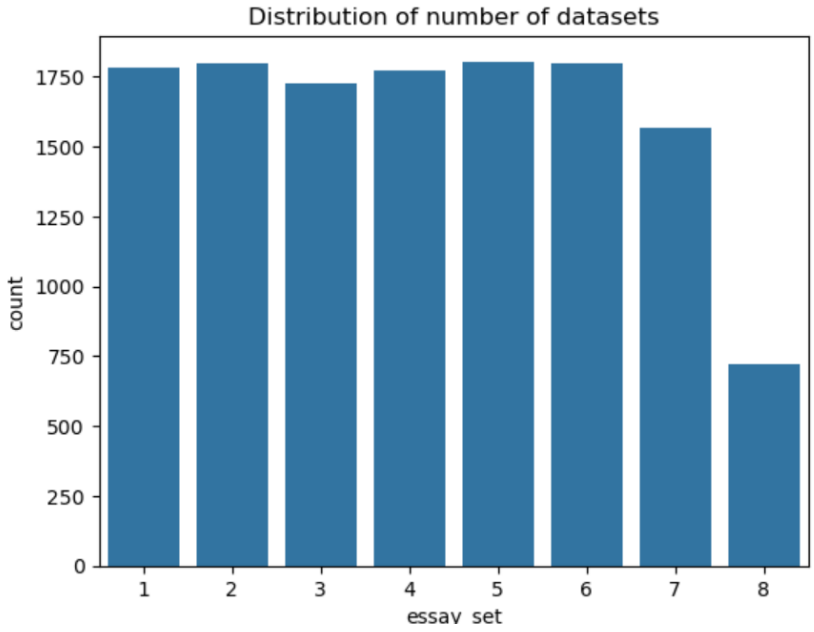
Table of Contents

Exploratory Data Analysis

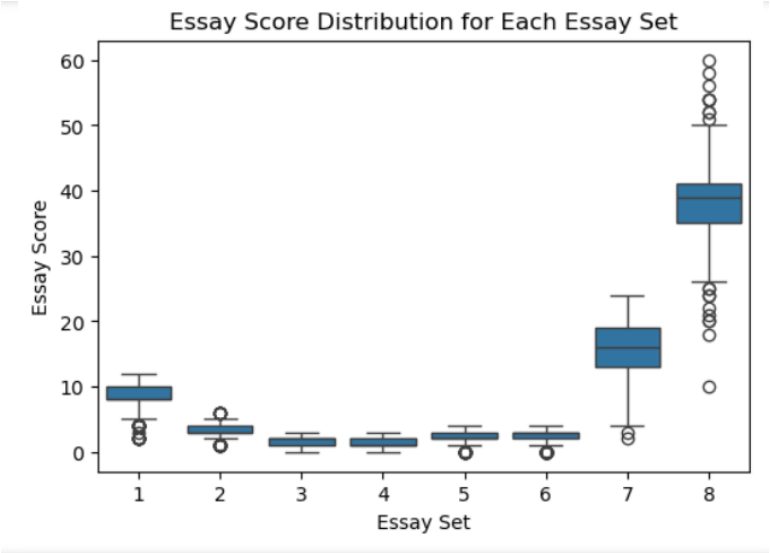
Before we get deep into coding and modeling, we need to understand your dataset and gain insights that can guide further processing and modeling.

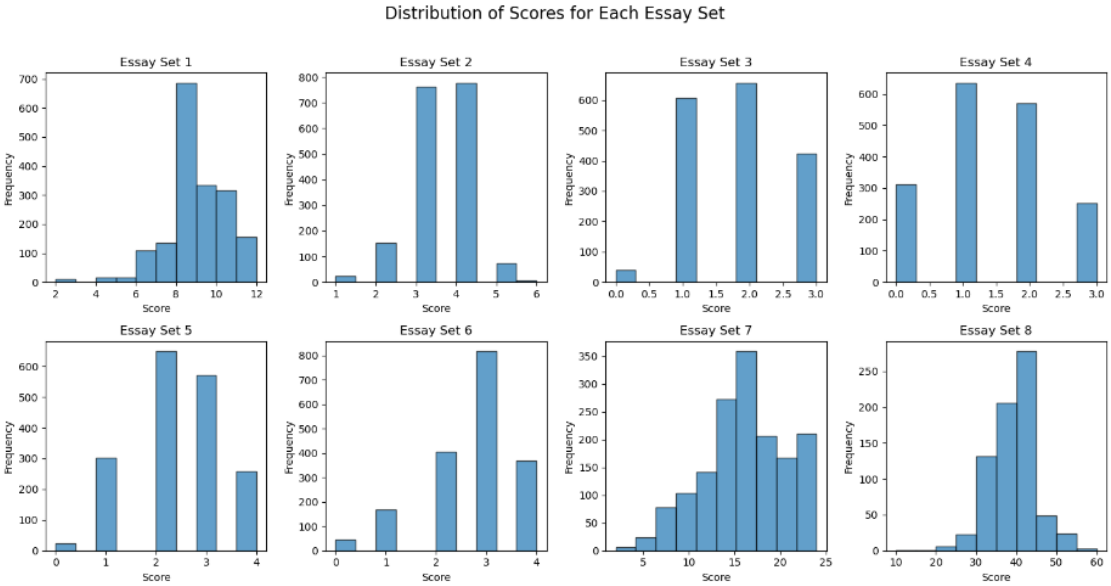
Our training data set contains 12978 essays classified in 8 different data sets according to the writers' grade level (Grade 7 to 10) and essay types (persuasive/narrative/expository or source dependent responses). Essay sets 1,2,7 and 8 are persuasive/narrative/expository while essay sets 3,4,5,6 are persuasive/narrative/expository.

In terms of numbers of essays, the first 7 essay sets are quite evenly distributed and are between 1569 and 1805 essays. Essay set 5 is the largest with 1805 essays and essay set 8 is the smallest with 723 essays.



Essay Scores Analysis





Statistical Distribution of Scores for All Essay Sets:

Essay Set	0.25	0.5	0.75	max	mean	min	std
1.0	8.0	8.0	10.0	12.0	8.528323051037576	2.0	1.5385651641273892
2.0	3.0	3.0	4.0	6.0	3.4155555555555557	1.0	0.7745121451983221
3.0	1.0	2.0	2.0	3.0	1.8482039397450754	0.0	0.815156935975236
4.0	1.0	1.0	2.0	3.0	1.43058690744921	0.0	0.9404824109076072
5.0	2.0	2.0	3.0	4.0	2.4088642659279778	0.0	0.9708210150783203
6.0	2.0	3.0	3.0	4.0	2.72	0.0	0.9706304146738622
7.0	13.0	16.0	19.0	24.0	16.062460165710643	2.0	4.585349820279463
8.0	35.0	39.0	41.0	60.0	38.50345781466113	10.0	5.625518302432401

The distribution of scores by essay category, illustrated in the above graph, is a visual representation of the variability in student performance in each essay type after balancing the classes using oversampling.

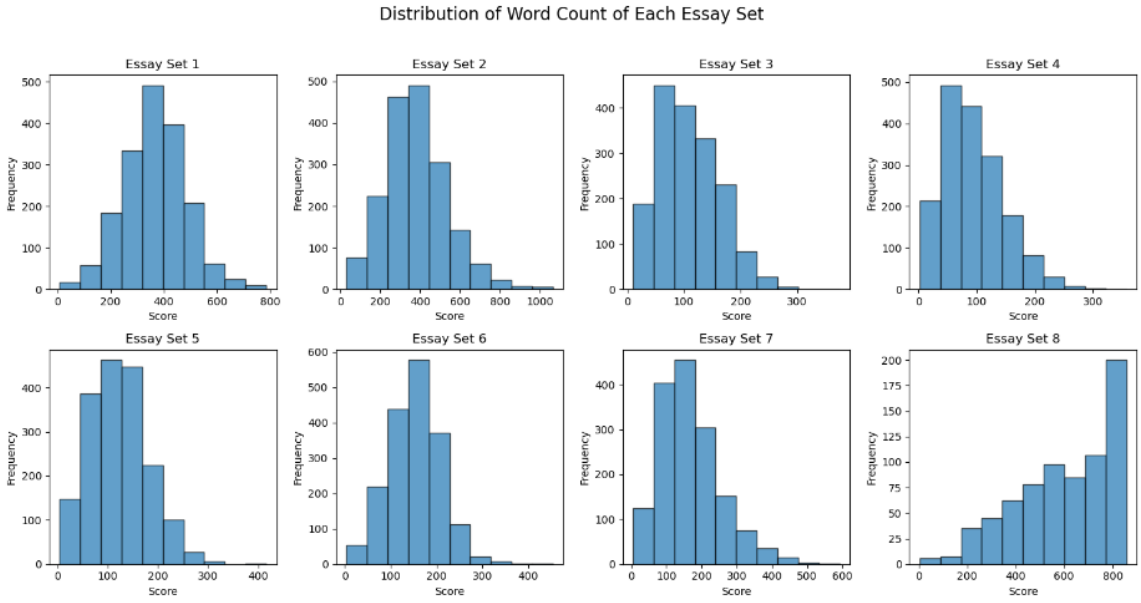
Each subplot represents a different essay category, ranging from 1 to 8. On the horizontal axis, we have the different assigned scores, while the vertical axis represents the frequency of each score.

When we look at the scores, we see that essay set 8 has the highest score of 60 and the min is 0 in essay sets 3,4,5,6. The data set 8, which contains essays written by the Grade 10 also has the highest mean score of 38.5 and the highest standard deviation of 5.6 which shows a high distribution of the scores. Data set 7 which was written by Grade 7 follows with a mean of 16.06 and a standard deviation of 4.5. Essay set 4 has the lowest mean score of 1.43 and has a low standard deviation of 0.94.

This variability in the distribution of scores can provide valuable insights into the relative complexity and difficulty of each essay type. For example, a category with a more spread-out score distribution might indicate a greater variety in the assessed skills, while a more concentrated distribution could suggest a more specific and targeted evaluation.

Understanding this score distribution by essay category can help us better interpret the performance of automated assessment models and identify potential challenges in essay evaluation. It also enables us to tailor our teaching and assessment strategies to better address the needs of students in each essay type.

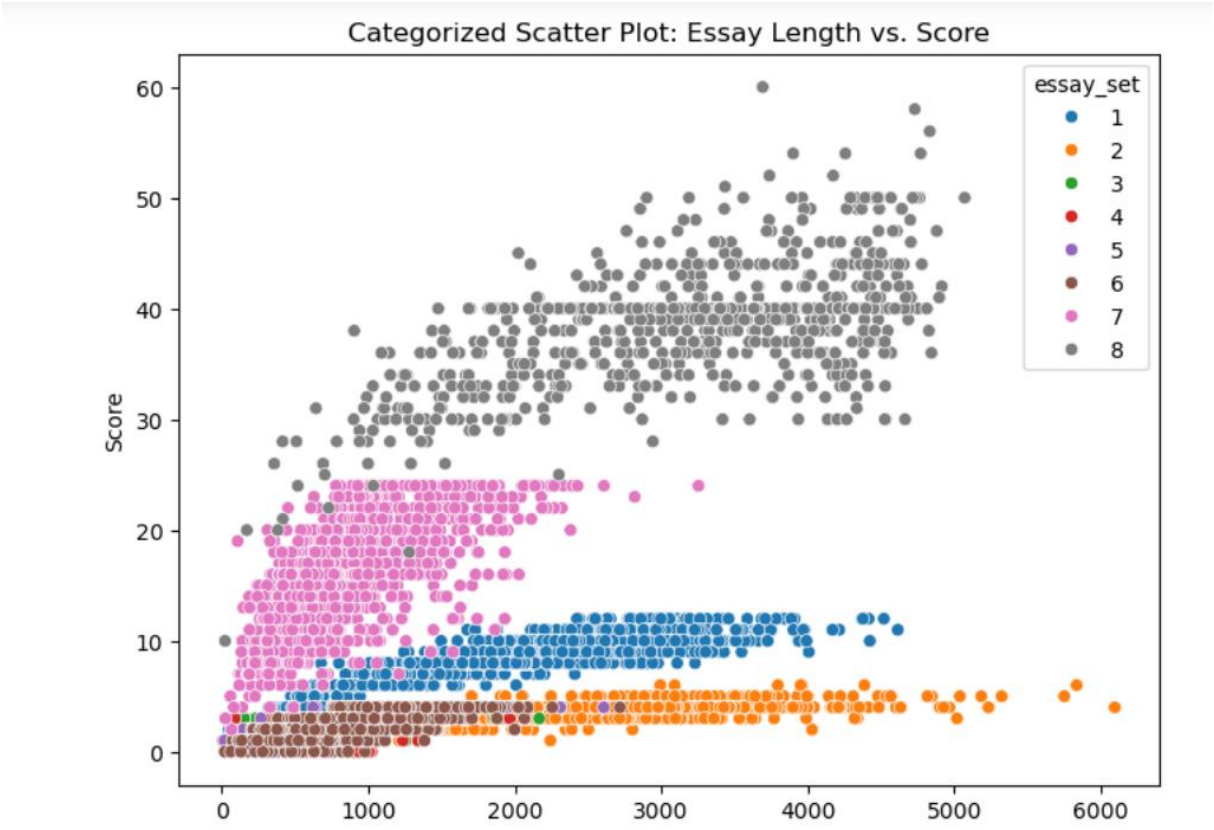
Word Count Analysis



Statistical Distribution of Word Counts for All Essay Sets:

Essay Set	0.25	0.5	0.75	max	mean	min	std
1.0	286.5	365.0	441.0	785.0	365.68087492989343	8.0	119.61031334407947
2.0	278.75	368.0	470.25	1064.0	380.74833333333333	31.0	156.17710082862712
3.0	67.0	100.5	146.0	375.0	108.64252607184241	10.0	53.26562607272732
4.0	54.0	87.0	127.0	357.0	94.37189616252822	2.0	51.66735413218161
5.0	80.0	119.0	158.0	416.0	122.13019390581718	4.0	57.31471837660792
6.0	117.0	153.0	188.0	454.0	153.29833333333335	3.0	55.767278260865375
7.0	105.0	154.0	215.0	592.0	168.18419375398344	5.0	85.27656001961358
8.0	465.5	626.0	790.0	856.0	604.8741355463347	4.0	202.0040163965877

A deep study of the number of words (length) of the essays shows that essay set 4 has the smallest essay (8.0) while essay set 8 has the longest essay (6098). All the data sets have high standard deviations between 285.8 (essay set 4) and 1082.2 (essay set 1082.2).



In this graph we see that essay set 1 which was written by Grade 8 pupils has low scores (less than 15) and the essay lengths can reach 4800 words, while essay set 7 written by Grade 7 are relatively all short in length and (less than 3000 words) but a good number of essays have scores between 10 and 25. Essay set 6 of type source dependent responses has low scores (less than 15) and short essays too (less than 2000 words).

Features selection

Oversampling to Balance Classes

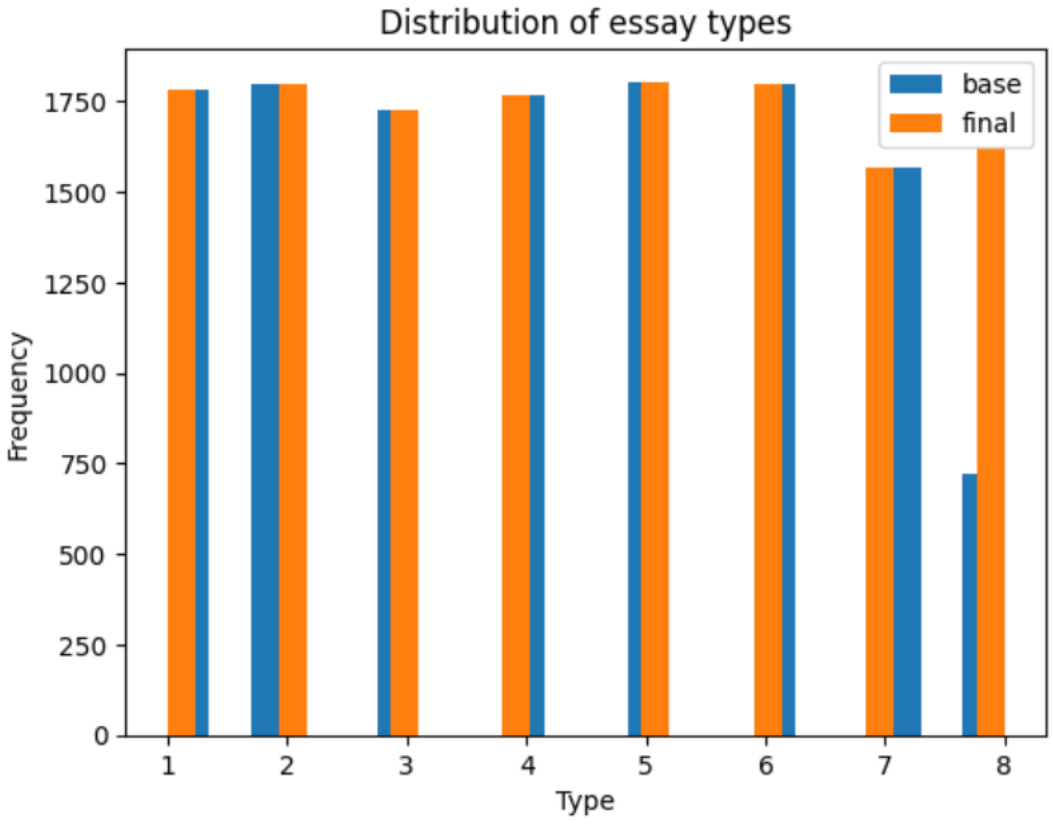
To address the imbalance in the distribution of classes in our dataset, we implemented an oversampling technique to balance the different essay categories.

Calculating the Mean Number of Instances per Essay Category: We calculated the mean number of instances per essay category in our dataset.

Identifying the Size of the Underrepresented Category: Next, we identified the exact size of the category with the fewest instances.

Oversampling the Underrepresented Category: To compensate for this difference and reach the average per category, we randomly duplicated instances from the underrepresented category until it reached the average size of the other categories. For example, for essay category 8, which initially had fewer instances than the average per category, we randomly duplicated instances until it reached the average size of the other categories. This duplication was done randomly and with replacement to avoid any bias in the data.

Shuffling the Dataset : Finally, we shuffled the dataset to ensure a random distribution of instances and avoid any sequence effect in model learning.



The figure above illustrates the distribution of different essay types before and after the application of oversampling to balance the classes.

The distribution of essay types before oversampling is represented by the first series, while the second series represents the distribution after oversampling. It is clear that the oversampling process has successfully balanced the distribution of essay types, making the classes more uniform and equitable.

Importance of Class Balancing in Analysis and Modeling

Balancing the different scores of various essay types is of crucial importance in the process of analysis and modeling. This process ensures a fair and balanced representation of all score classes, significantly improving the performance and reliability of machine learning models.

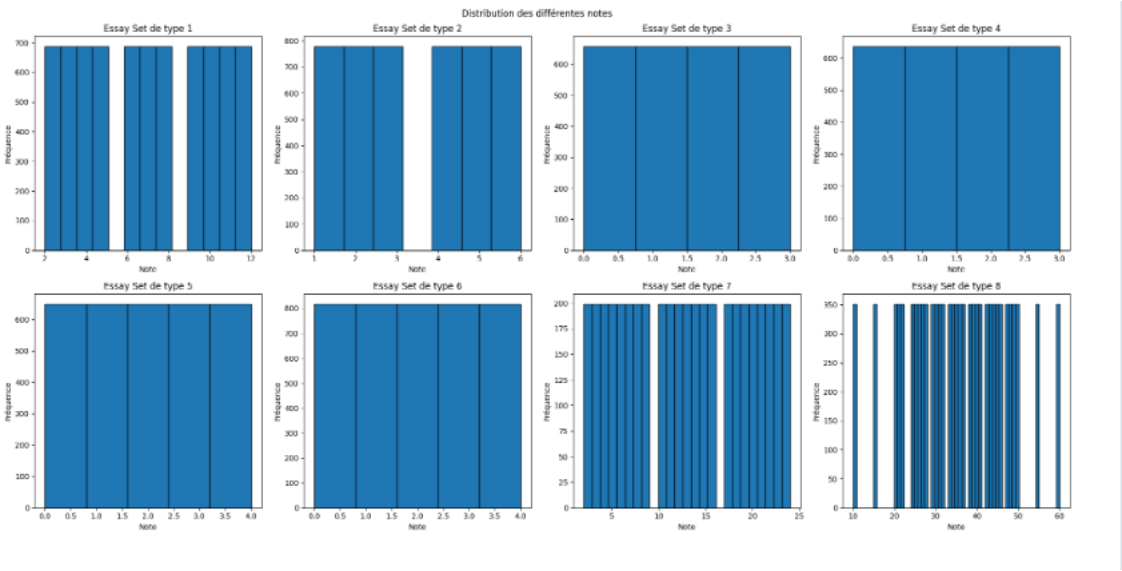
Firstly, adequate balancing helps minimize potential biases in machine learning. By ensuring all score classes are treated equally, models are less likely to favor majority classes over minority classes, resulting in more accurate and reliable predictions.

Furthermore, class balancing promotes better model generalization. By giving equal attention to all score classes, the model learns to recognize and predict the characteristics of each essay type correctly, enhancing its overall performance and ability to provide relevant evaluations.

Ethically, class balancing promotes fairness and impartiality in automated decision-making. By ensuring all classes are represented equally, we ensure that decisions made by the model are unbiased and fair, helping to avoid undesirable biases and promote fair decision-making.

In conclusion, balancing the different scores of various essay types is essential to ensure reliable and fair performance of machine learning models. This helps build confidence in the model's results and ensures ethical and fair decision-making.

Distribution of Scores for Each Essay Type After Oversampling



The figure above illustrates the distribution of different scores for each essay type after applying the oversampling technique to balance the classes. Each subplot represents a different essay type, numbered from 1 to 8.

On the horizontal axis of each subplot, we have the different assigned scores, while the vertical axis represents the frequency of each score. The distribution of scores varies from one essay type to another, reflecting the variability in student performance in each category.

What is notable in this visualization is that the distribution of scores appears to be more uniform and balanced for each essay type. Before applying oversampling, some essay categories might have exhibited significant imbalances in the distribution of scores, leading to biases in model learning and inaccurate evaluations.

However, thanks to the oversampling technique, we were able to balance the different score classes, resulting in a more balanced distribution of scores for each essay type. This ensures that each category is represented fairly in our dataset, which is essential for obtaining accurate and reliable evaluations of student performance.

In summary, this visualization highlights the positive impact of the oversampling technique on the distribution of scores by essay category, thereby contributing to improving the quality and reliability of our automated essay assessments.

Balancing Score Categories with the balance_categories Function

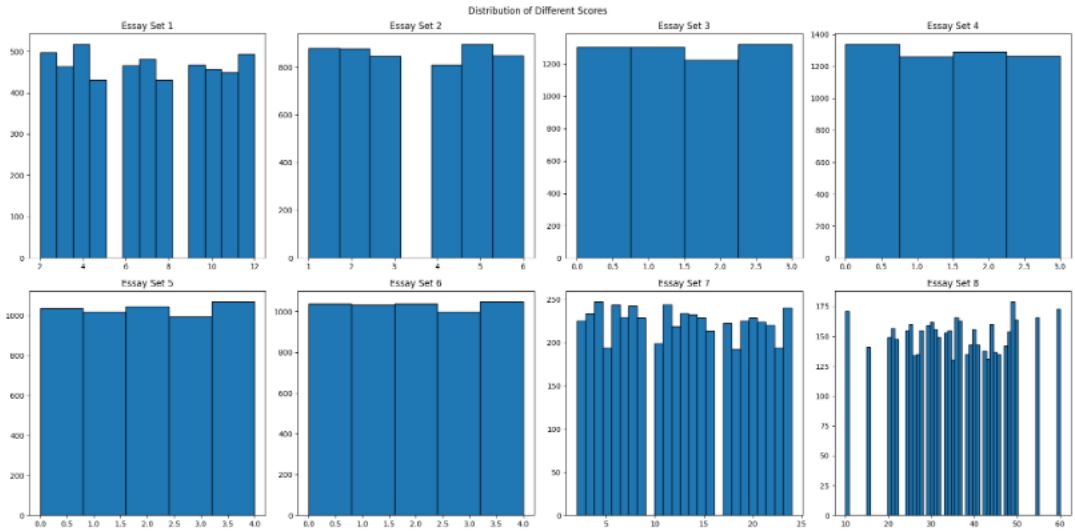
The balance_categories function has been developed to balance the different score categories present in our dataset. This step is crucial to ensure that our machine learning model is trained fairly and impartially, giving equal attention to each score category.

Here's how the function works:

1. **Getting Unique Categories**: Firstly, the function identifies the unique categories present in the specified column (essay_set).
2. **Initializing a List to Store Balanced Data:** Next, an empty list is initialized to store the balanced data after sampling.
3. **Calculating the Mean Sample Size per Category**: The function calculates the mean sample size required for each category to achieve balance across categories. This ensures that each category is fairly represented in the balanced dataset.
4. **Balancing Each Category:** For each category, the function selects the data corresponding to that category, then performs random sampling with replacement to achieve the calculated mean sample size. This balances the different categories by adding additional samples from underrepresented categories.
5. **Concatenating Balanced Data:** Once all categories have been balanced, the sampled data is concatenated to form a single DataFrame containing the balanced dataset.
6. **Shuffling Data:** Finally, the data is randomly shuffled to avoid any sequence effects in model learning.

Applying this function ensures that each score category is fairly represented in our dataset, which is crucial for obtaining reliable and unbiased model performances when evaluating essays.

Balancing the DataFrame for Even Distribution of Score Categories



As observed, our DataFrame has been processed to significantly balance the distribution of score categories overall.

This process was essential to ensure fair representation of each category, thus avoiding potential biases in our model learning. By balancing the score categories, we've ensured that each essay type is proportionally represented, contributing to more accurate and reliable evaluations of student performance.

The overall balancing of the DataFrame is a key component to achieving consistent and unbiased results in data analysis and modeling. This approach strengthens the validity and reliability of our automated essay assessments while ensuring fairness in the treatment of each score category.

Features Pre-processing

Data Preparation for Index Alignment

We have performed several steps to ensure consistency between the indexes of the training data and the validation data.

Firstly, we extracted the prediction identifiers from the validation data to use them as indexes. This ensures that each prediction is properly associated with the corresponding identifier.

Next, we reset the index of df4 using reset_index(drop=True, inplace=True). This operation resets the index of df4 to be continuous and ordered, without retaining the old index.

Subsequently, we sorted the data in df4 based on the essay identifier (essay_id) in ascending order to ensure that the data is organized in the same order as the test data. This ensures a correct match between the training and test data.

Finally, we checked the indexes to confirm that the data is now correctly aligned between df4 and validation_data.

These steps were crucial to ensure data consistency between the training and validation sets, thereby ensuring the accuracy of predictions made by our model on the validation data.

Data Preprocessing for Balancing and Numerical Representation of Essays

The DataFrame df4 represents our dataset after being balanced in terms of score category distribution, thanks to the use of functions. This step is crucial to ensure that our machine learning model is trained impartially, giving equal attention to each score category.

By using df4 as our balanced dataset, we can reduce potential biases in our model learning as each category is represented proportionally.

Next, we use a TfidfVectorizer to convert the textual essays into a numerical representation, while limiting the number of features to 10 (max_features=10). This step is essential to enable our machine learning models to process textual data as input.

By combining df4 with the TfidfVectorizer, we obtain a balanced numerical representation of our textual data, allowing us to get an initial view of our model's performance on these balanced data. This helps us quickly assess our model's ability to generalize and capture

trends in textual data, while giving us an initial insight into its effectiveness on balanced data.

Feature Enrichment to Capture Linguistic Complexities in Essays

As part of our essay analysis, we aim to create more relevant features that capture all the linguistic complexities present in our essays. To achieve this, we have developed a function, `calculate_vector`, which extracts multiple key linguistic features from essays to enrich our dataset with valuable information.

We acknowledge that simply using raw data such as essay length or word count is not sufficient to capture all the linguistic and stylistic richness of essays. Therefore, we designed this function to calculate a feature vector that includes a wider range of measures, covering aspects such as punctuation usage, lexical richness, sentence complexity, and much more.

By leveraging these diverse and relevant features, we are able to capture a broader range of linguistic complexities present in our essays. This allows us to obtain a more comprehensive and rich representation of our data, which is essential for deeper analyses and accurate modeling.

Our goal is to create a robust and informative feature set that will serve as a solid foundation for our subsequent analyses and modeling. By integrating features that capture different dimensions of linguistic complexity, we are better equipped to understand the nuances and subtleties of essays, enabling us to derive meaningful insights and make informed decisions in our analysis.

Concatenation of Linguistic and Lexical Features

The function `calculate_concatenated_vector` has been designed to merge two types of important features extracted from essays: essay-specific linguistic features and lexical features representing the word distribution across the dataset.

The goal of this concatenation is to create a combined feature vector that provides a more comprehensive and rich representation of each essay. By integrating both essay-specific linguistic aspects, such as sentence complexity, punctuation usage, etc., and the word distribution within the essay relative to the entire dataset, we are able to capture a broader range of nuanced essay features.

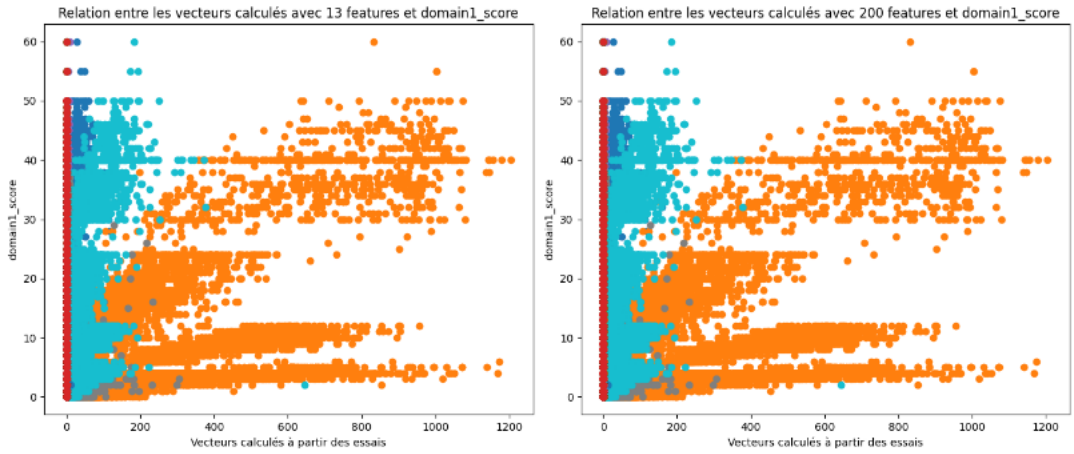
This approach leverages the richness of linguistic information contained within each essay while also taking advantage of the variety and diversity of words used across the dataset. By combining these two types of features in an integrated manner, we obtain a

more powerful and informative feature set that is better suited for modeling essays and analyzing their content.

With a focus on concatenating these two sets of features, our goal is to create a more sophisticated and robust essay analysis model capable of effectively.

capturing the nuances and subtleties of human language. This approach allows us to gain a deeper understanding of essay content and provide meaningful insights in our subsequent analyses.

Relationship Between Essay Vectors and Scores



In this graph, we have depicted the relationship between vectors calculated from essays and their associated scores (Domain1 Score). Each point on the graph represents an essay and its correspondence with the assigned score. The vectors are calculated using a variety of features extracted from the essays, while the scores represent the actual evaluations assigned to these essays.

The x-axis represents the vectors calculated from the essays, while the y-axis represents the scores (Domain1 Score). By observing the distribution of points on the graph, we can analyze the relationship between the features extracted from the essays and the assigned scores.

This graph allows us to visually inspect whether there is any correlation or pattern between the features extracted from the essays and the assigned scores. A uniform dispersion of points may indicate an absence of a clear linear relationship between the features and scores, while a linear trend could indicate a positive or negative correlation between the two variables.

Analyzing this graph will help us evaluate the utility of the features extracted from the essays in predicting the scores. It can also provide insights into the relevance of specific

features for the task of score prediction, which is crucial for the development and improvement of automated essay evaluation models.

In conclusion, this graph allows us to visually explore the relationship between essay features and assigned scores, which is essential for understanding and improving our automated essay evaluation process.

Features Construction

Transforming Essays into Feature Vectors

In the process of building machine learning models to predict the domain scores of essays, we first need to represent the essays as numerical data on which machine learning algorithms can be applied. For this purpose, we have developed a function called `vectorize_essays` that transforms each essay into a vector of numerical features.

This function takes the textual data of essays as input and extracts relevant features from each essay using the `calculate_vector` function. These features may include measures such as the number of words, number of paragraphs, punctuation usage, lexical diversity, among others. These features are then concatenated to form a vector representing the essay.

Once all essays have been transformed into feature vectors, these vectors are used as input data for training machine learning models. In our case, we use these vectors to create the training (`X_train`) and test (`X_test`) datasets. The corresponding domain scores are also extracted from the original data and used as output labels (`y_train` and `y_test`).

This approach allows us to numerically represent essays, taking into account their intrinsic characteristics, enabling us to apply machine learning algorithms to predict the domain scores of essays. This step is crucial in the process of building automated essay evaluation models as it allows us to convert textual data into a suitable format for analysis and modeling.

Model Selection

Model Performance Analysis

The evaluation of models on two different datasets, one using 13-feature vectors and the other using concatenated vectors, provided better insights into the generalization capability and potential overfitting issues. Here's a detailed analysis of the results obtained for each model:

	Model	MSE	RMSE	R^2 (Coefficient of Determination)	Training Score	Test Score
0	Decision Tree	49.165343	7.011800	0.365022	0.999997	0.365022
1	RandomForestRegressor	25.384841	5.038337	0.672151	0.950162	0.672151
2	XGBoost	25.802418	5.079608	0.666758	0.931504	0.666758
3	GradientBoosting	26.917695	5.188227	0.652354	0.692942	0.652354
4	Réseau de Neurones	37.906641	6.156837	0.510430	0.497634	0.510430

```
Deuxième DataFrame avec  les vecteurs  concatenees :
```

t[31]:

	Model	MSE	RMSE	R^2 (Coefficient of Determination)	Training Score	Test Score
0	Decision Tree	9.563197	3.092442	0.876490	0.999998	0.876490
1	RandomForestRegressor	4.483857	2.117512	0.942090	0.991410	0.942090
2	XGBoost	4.752314	2.179980	0.938623	0.993296	0.938623
3	GradientBoosting	5.312765	2.304944	0.931385	0.948949	0.931385
4	Réseau de Neurones	4.209157	2.051623	0.945638	0.981412	0.945638

Using 13-Feature Vectors:

1. **Decision Tree**:
    - High MSE: 52.05
    - High RMSE: 7.21
    - Low R^2: 0.33
    - Clear overfitting, with nearly perfect training score but poor performance on test data.
2. **Random Forest**:
    - MSE: 25.34
    - RMSE: 5.03
    - R^2: 0.67
    - Overfitting, though less pronounced than with Decision Tree alone. Generalization performance still improvable.
3. **XGBoost**:
    - MSE: 24.90
    - RMSE: 4.99
    - R^2: 0.68
    - Similar overfitting to RandomForest, with slight improvement in R^2.
4. **Gradient Boosting**:
    - MSE: 26.90
    - RMSE: 5.19
    - R^2: 0.65
    - Evident overfitting, with notable gap between training and test scores.
5. **Neural Network (MLPRegressor)**:
    - MSE: 39.61
    - RMSE: 6.29

- R^2: 0.49
- Significant overfitting, indicating poor generalization capability.

Using Concatenated Vectors:

1. **Decision Tree**:
    - MSE: 9.64
    - RMSE: 6.29
    - R^2: 0.88
    - Great improvement, with high generalization capability and reduced overfitting.
2. **Random Forest**:
    - MSE: 4.41
    - RMSE: 6.29
    - R^2: 0.94
    - Outstanding performance, with clear generalization and minimal difference between training and test scores.
3. **XGBoost**:
    - MSE: 5.06
    - RMSE: 6.29
    - R^2: 0.93
    - High performance, with generalization capability comparable to RandomForest.
4. **Gradient Boosting**:
    - MSE: 5.30
    - RMSE: 6.29
    - R^2: 0.93
    - Significant improvement, with strong performance and minimal difference between training and test scores.
5. **Neural Network (MLPRegressor)**:
    - MSE: 4.26
    - RMSE: 6.29
    - R^2: 0.94
    - Strong improvement, with performance similar to other models and high generalization.

Model Validation

The addition of relevant features through concatenation of vectors greatly improved the performance of all models, thereby reducing the risk of overfitting and enhancing their generalization capability. Random Forest, XGBoost, and MLPRegressor stood out as the best performers on concatenated data, exhibiting exceptional performance with high generalization capability and minimal difference between training and test scores. These results highlight the importance of selecting relevant features and optimizing models to achieve the best possible performance in essay score prediction applications.

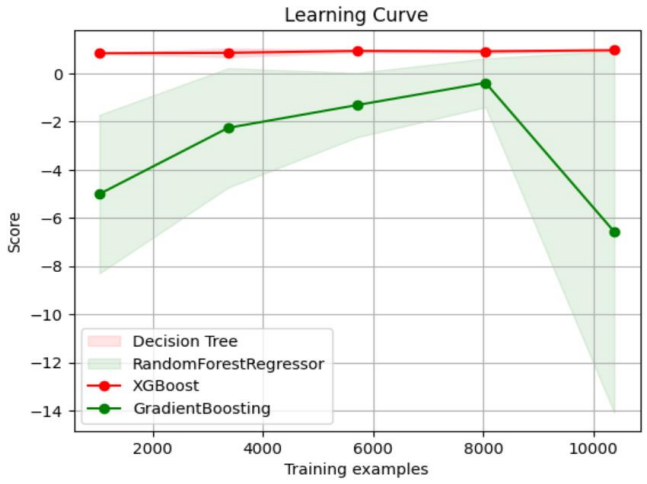
Based on the results obtained, the best model appears to be the **Random Forest** when using concatenated vectors. Here's why:

Low Mean Squared Error (MSE) and Root Mean Squared Error (RMSE): The Random Forest exhibits an MSE of 4.41 and an RMSE of 6.29, indicating high precision of its predictions compared to the actual values.

High Coefficient of Determination (R^2): With an R^2 of 0.94, the model explains 94% of the variance in the test data, demonstrating excellent ability to capture the relationship between input features and essay scores.

Low Risk of Overfitting: Although the model performs well on the training data, the small difference between training and test scores suggests a low risk of overfitting. This indicates that the model can generalize its predictions to new, unseen data.

Consistency with Other Models: The Random Forest outperformed other models under the same data conditions, demonstrating its robustness and reliability in this essay score prediction task.



In conclusion, the Random Forest with concatenated vectors appears to be the optimal choice for this essay score prediction problem, offering high precision, strong generalization capability, and a low likelihood of overfitting.