IME672: DM &KD
Analyzing Typing Behavior to Predict Essay Quality

Members

Name	Roll No
Satyam Maddeshiya	200896
Prateek Kumar Pandey	200710
Yash Goel	201142
Rishabh Verma	200788

Problem Description

- The goal is to predict the score an essay received from its log of user inputs.
- Use process features from keystroke log data to predict overall writing quality
- It helps identify relationships between learners' writing behaviors and writing performance.

Data Understanding

Keystroke Logging Dataset

- `Event ID` indexes the keyboard and mouse operations in chronological order.
- `Down Time` denotes the time (in milliseconds) when a key or mouse was pressed
- `Up Time` indicates the release time of the event.
- `Action Time` represents the duration of the operation (i.e., Up Time - Down Time).
- `Position` registers cursor position information to help keep track of the location of the leading edge.
- `Word Count` displays the accumulated number of words typed in.
- `Text Change` shows the exact changes made to the current text
- `Activity` indicates the nature of the changes (e.g., `Input`, `Remove/Cut`)
- score - The score the essay received out of 6 (the prediction target for the competition)

Event ID	Down Time	Up Time	Action Time	Event	Position	Word Count	Text Change	Activity
1	30185	30395	210	Leftclick	0	0	NoChange	Nonproduction
2	41006	41006	0	Shift	0	0	NoChange	Nonproduction
3	41264	41376	112	I	1	1	I	Input
4	41556	41646	90	Space	2	1		Input
5	41815	41893	78	b	3	2	b	Input
6	42018	42096	78	e	4	2	e	Input
7	42423	42501	78	l	5	2	l	Input
8	42670	42737	67	i	6	2	i	Input
9	42873	42951	78	e	7	2	e	Input
10	43041	43109	68	v	8	2	v	Input
11	43289	43378	89	Space	9	2		Input
12	44560	44605	45	Backspace	8	2		Remove/Cut
13	44661	44762	101	e	9	2	e	Input
14	44954	45032	78	Space	10	2		Input
15	45325	45381	56	t	11	3	t	Input
16	45460	45538	78	h	12	3	h	Input
17	45640	45730	90	a	13	3	a	Input
18	45741	45808	67	t	14	3	t	Input
19	45933	46011	78	Space	15	3		Input

Train Data

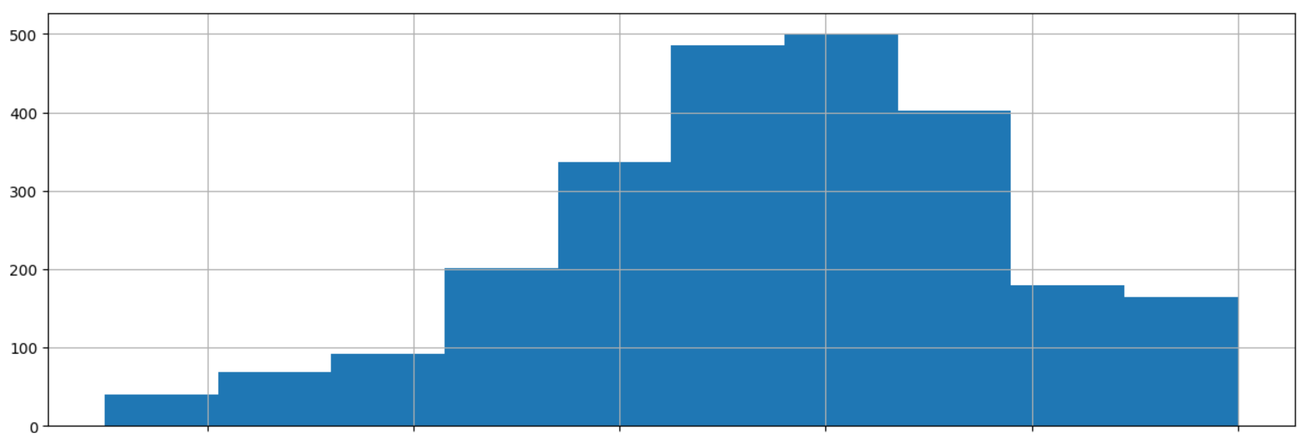
- **<u>Information</u>**

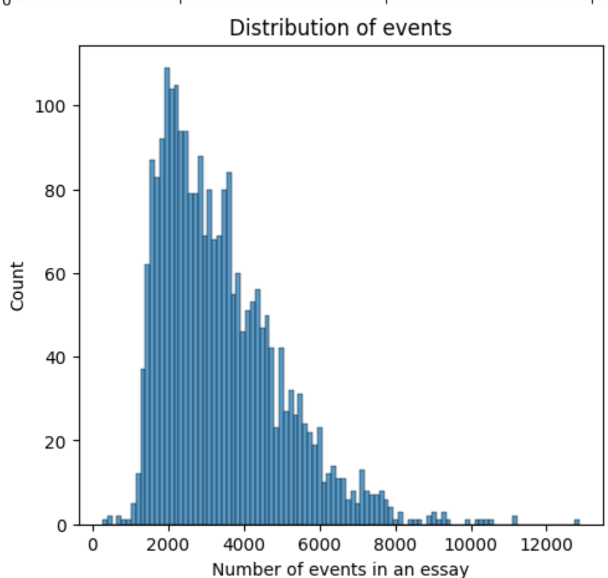
RangeIndex: **8405898 entries, 0 to 8405897**
Data columns (total of 11 columns):

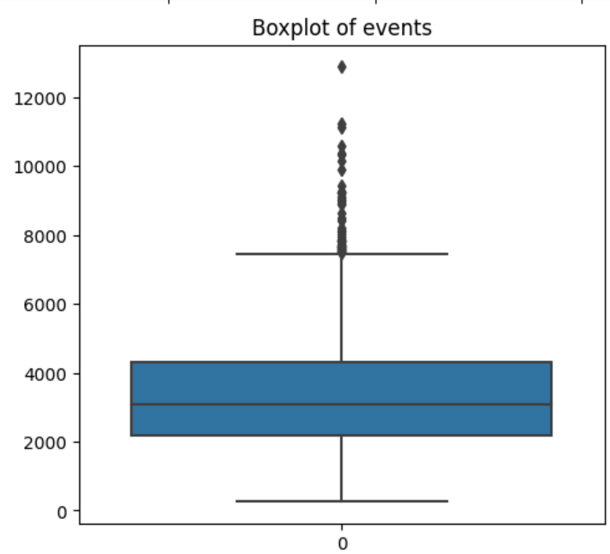
```
#   Column          Dtype
---  ------          -----
0   id              object
1   event_id        int64
2   down_time       int64
3   up_time         int64
4   action_time     int64
5   activity        object
6   down_event      object
7   up_event        object
8   text_change     object
9   cursor_position  int64
10  word_count      int64
dtypes: int64(6), object(5)
```

	event_id	down_time	up_time	action_time	cursor_position	word_count
count	8.405898e+06	8.405898e+06	8.405898e+06	8.405898e+06	8.405898e+06	8.405898e+06
mean	2.067649e+03	7.935603e+05	7.936584e+05	9.808498e+01	1.222964e+03	2.314687e+02
std	1.588284e+03	5.149451e+05	5.149428e+05	2.533985e+02	9.485242e+02	1.759088e+02
min	1.000000e+00	1.060000e+02	2.520000e+02	0.000000e+00	0.000000e+00	0.000000e+00
25%	8.520000e+02	3.731842e+05	3.732820e+05	6.600000e+01	4.990000e+02	9.600000e+01
50%	1.726000e+03	7.208860e+05	7.209800e+05	9.300000e+01	1.043000e+03	2.000000e+02
75%	2.926000e+03	1.163042e+06	1.163141e+06	1.220000e+02	1.706000e+03	3.270000e+02
max	1.287600e+04	8.313630e+06	8.313707e+06	4.474700e+05	7.802000e+03	1.326000e+03

- Total 2471 Unique Essay Id's

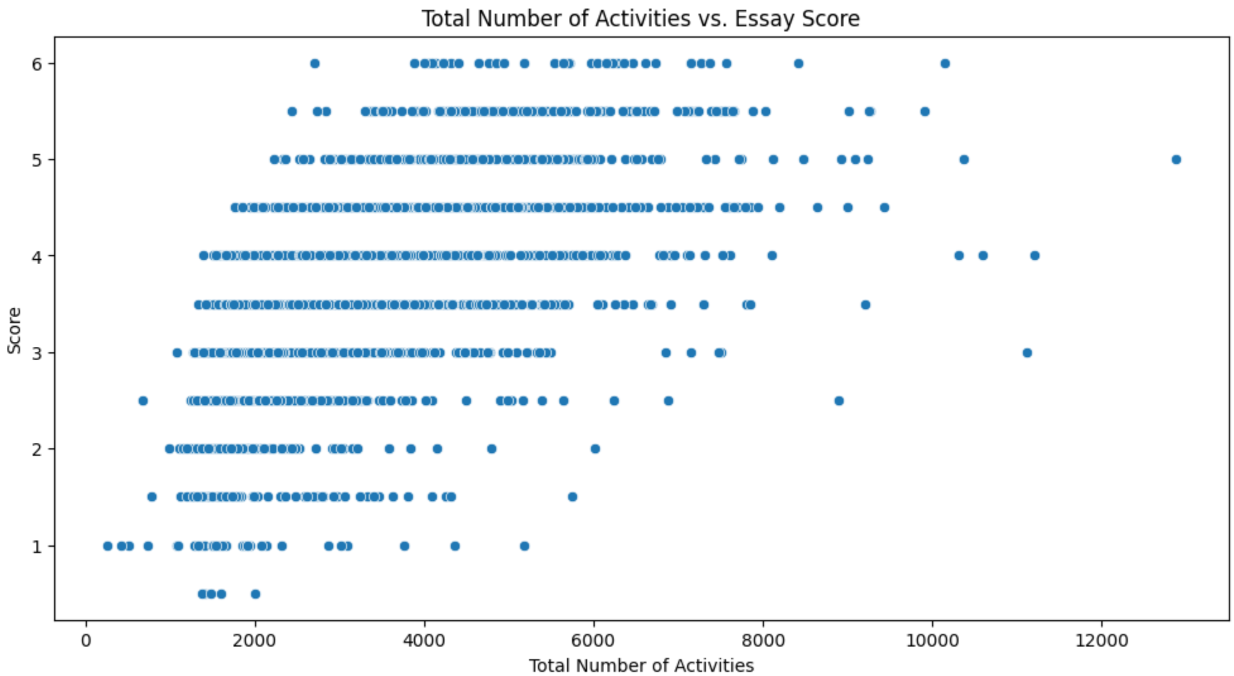


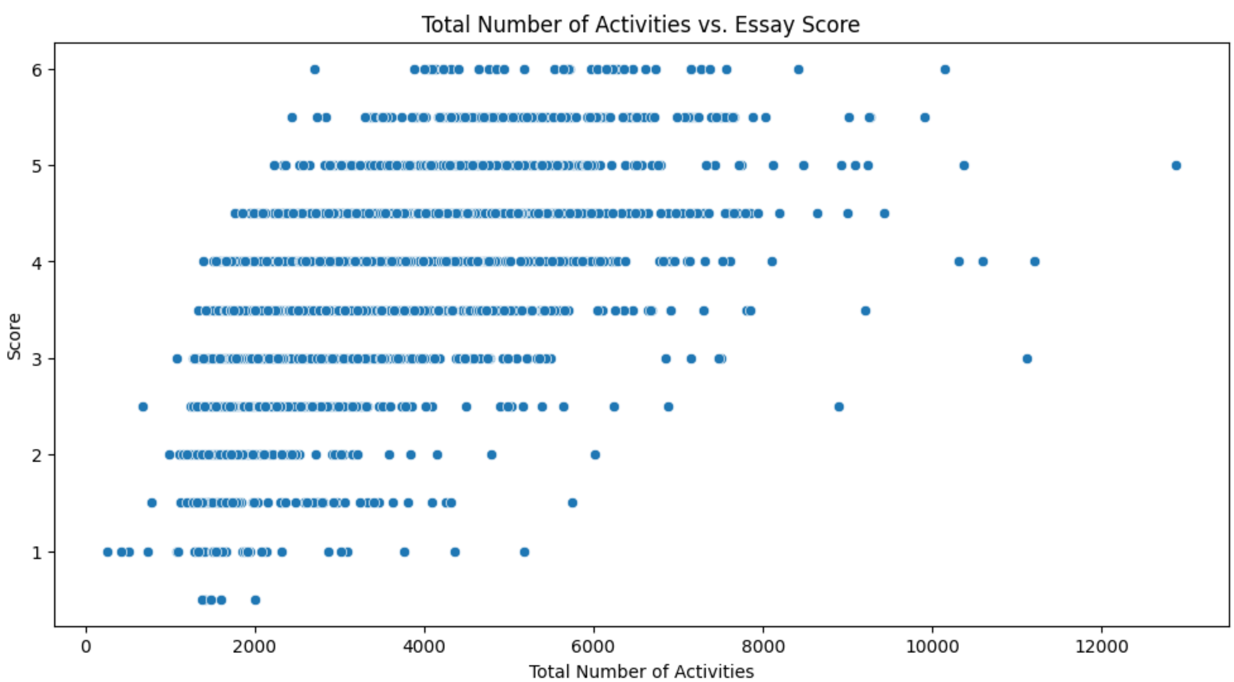




Scatter Plots for visualization

- **Scatter plots to visualize the relationship between typing behavior features and essay**





- **The dataset is already Pre-Processed.**

Features Extracted for evaluation

```
    action_time_mean=pd.NamedAgg(column="action_time", aggfunc="mean"),
    action_time_sum=pd.NamedAgg(column="action_time", aggfunc="sum"),
    action_time_min=pd.NamedAgg(column="action_time", aggfunc="min"),
    action_time_max=pd.NamedAgg(column="action_time", aggfunc="max"),
    word_count_max=pd.NamedAgg(column="word_count", aggfunc="max"),
    cursor_position_max=pd.NamedAgg(column="cursor_position",
aggfunc="max"),
    cursor_position_mean=pd.NamedAgg(column="cursor_position",
aggfunc="mean"),
    activity_count=pd.NamedAgg(column="activity", aggfunc="count"),
    text_change_counts = pd.NamedAgg(column="text_change",
aggfunc="count"),
    down_event_counts = pd.NamedAgg(column="down_event", aggfunc="count"),
    up_event_counts = pd.NamedAgg(column="up_event", aggfunc="count"),
    score = pd.NamedAgg(column="score", aggfunc="max")
```

Model Building:

Data Preparation

The dataset, represented by df_agg_new, is initially explored using the shape attribute to understand its dimensions. The target variable (y) is defined as the 'score' column, and the feature matrix (X) is created by excluding the 'id' and 'score' columns.

Data Splitting

The dataset is split into training and testing sets using the train_test_split function from scikit-learn. Approximately 67% of the data is used for training (X_train and y_train), while the remaining 33% is reserved for testing (X_test and y_test).

Model Selection and Cross-Validation

Three different regression models are chosen for evaluation: Linear Regression, Random Forest Regressor, and Extreme Gradient Boosting (XGBoost). Cross-validation is performed to assess the models' performance using metrics such as Mean Absolute Error (MAE), Mean Squared Error (MSE), R^2, and Accuracy. The cv_comparison function is created to streamline the cross-validation process.

Model Evaluation:

Cross-Validation Results
The cross-validation results (comp) provide a comparative overview of the three models' average performance across multiple metrics. These metrics help evaluate how well each model generalizes to unseen data.

Mean Absolute Error Comparison
The Mean Absolute Error (MAE) for each model is further examined across different folds, providing insights into the consistency of model performance. The average MAE and a comparison table (maes_comp) are presented.

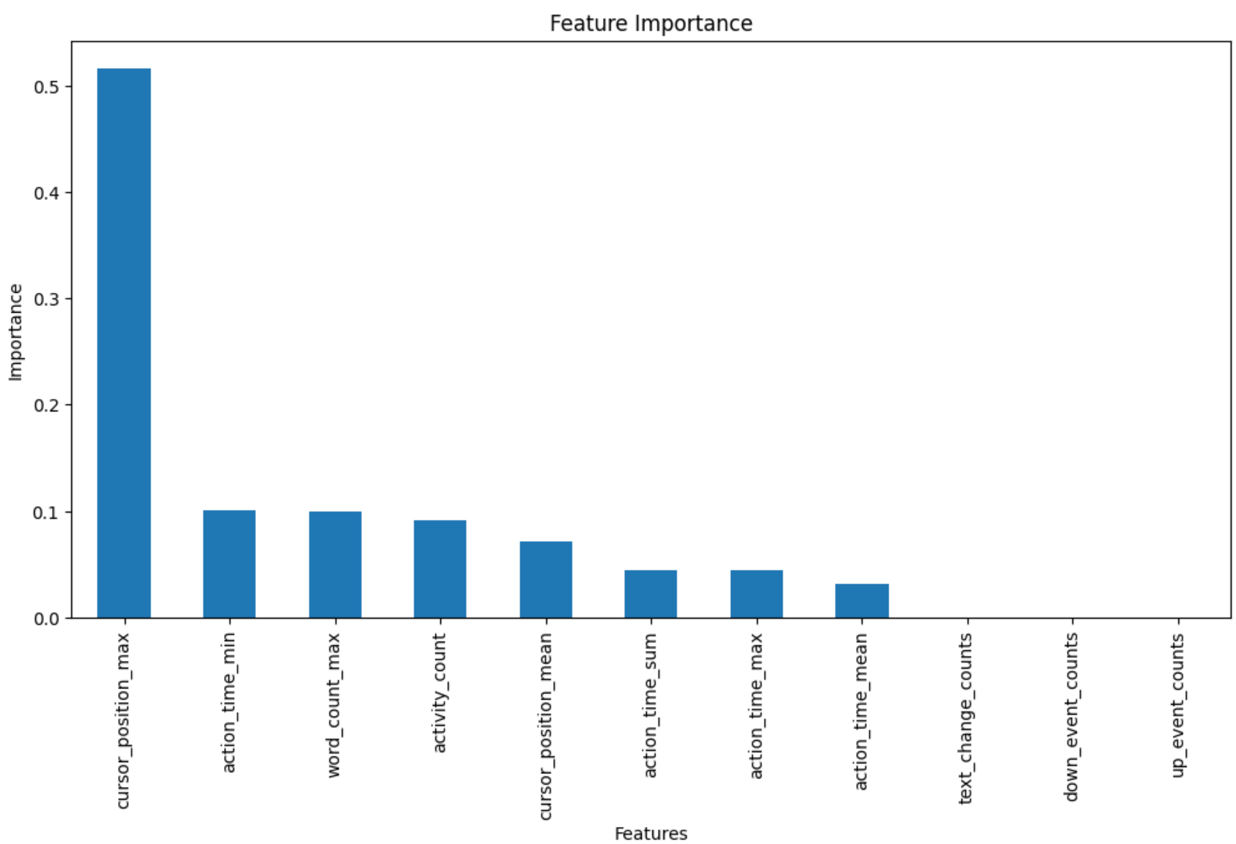
Hyperparameter Tuning (Random Search)
Randomized search is conducted to identify optimal hyperparameters for both the Random Forest and XGBoost models. This step aims to enhance the models' performance by fine-tuning parameters such as the number of estimators, maximum depth, and learning rate.

Model Training with Best Parameters
The final step involves training the models with the identified optimal hyperparameters. This includes creating instances of Linear Regression, Random Forest, and XGBoost models, fitting them to the training data (X_train and y_train), and preparing them for the subsequent evaluation on the test set.

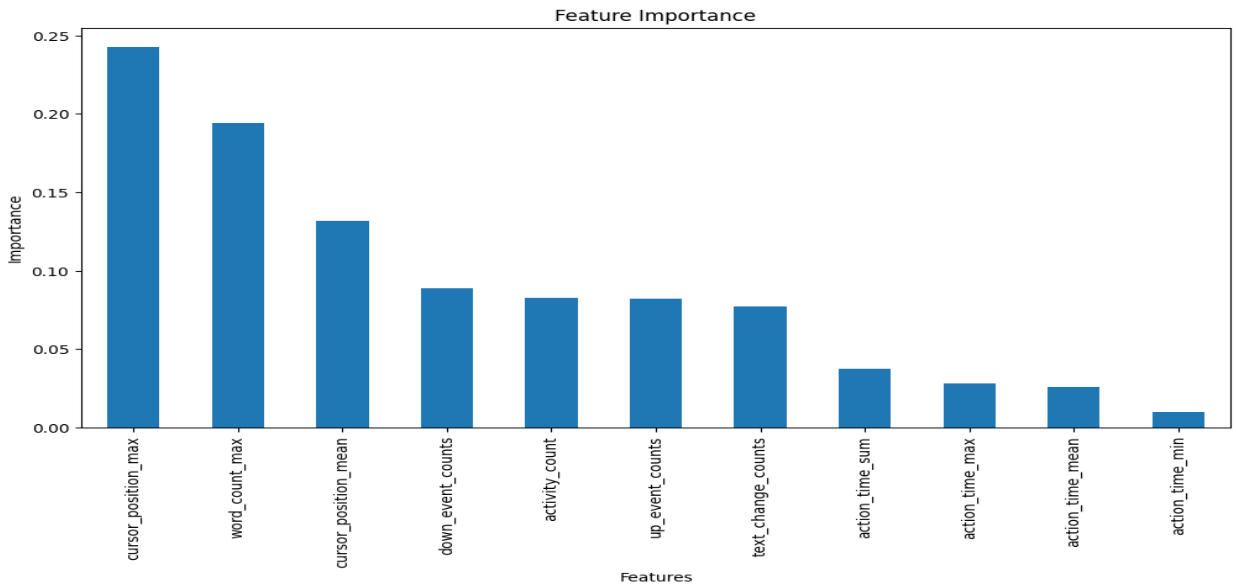
Results Interpretation:

Feature Importance Visualization
Feature importance is visualized for both the Random Forest and XGBoost models. This visualization highlights the most influential features contributing to the models' predictions. The importance is presented in bar charts, aiding in feature selection and understanding the models' decision-making processes.

ForXGBRegressor Important Feature are :



For RandomForestRegressor Important Feature are :



Final Model Comparison on Test Set

The three final models (Linear Regression, Random Forest, and XGBoost) are evaluated on the previously untouched test set (X_test and y_test). Performance metrics, including Mean Absolute Error, Mean Squared Error, R^2, and Accuracy, are calculated and presented in a comparison table (final_scores). This final assessment provides valuable insights into how well each model performs on new, unseen data. And the best model is

XGBoost Regressor give the best accuracy 82%