A Novel Ensemble Learning Approach for the Classification of Eye Movements While Reading Using Tree-Based Models

Aditya Vasantharao
Sameer Gabbita
Thomas Jefferson High School for Science & Technology
Dr. Selma Yilmaz
October 22, 2021

Table of Contents

1. Project Goal
2. Description of Dataset
3. Pre-Processing
4. Attribute Selection Algorithms
5. Classification Algorithms
6. Results and Analysis
7. Conclusion
8. How To Reproduce Our Model
9. Team Member Tasks
10. Sources

1. Project Goal

We want to determine if there is a way to predict how different eye movements indicate what information is relevant to a person when reading a text and retrieving information (answering questions) from that text.

2. Description of Dataset

We collected our dataset from OpenML. We loaded and processed the dataset using Python's Pandas library. The dataset had a total of 10,936 instances, with each instance consisting of 27 attributes. The class variable, the column named "label" in the CSV file, had 3 different discrete values: 0 (irrelevant), 1 (relevant), 2 (correct).

	lineNo	assgNo	fixcount	firstPassCnt	P1stFixation	P2stFixation	prevFixDur	firstfixDur	firstPassFixDur	nextFixDur	...	regressLen	nextWordRegress	regressDur
0	1	1	1	1	1	0	0	100	100	99	...	0	0	0
1	2	1	1	1	1	0	99	278	278	159	...	0	0	0
2	3	1	1	1	1	0	278	159	159	159	...	0	0	0
3	4	1	1	1	1	0	159	159	159	139	...	0	0	0
4	5	1	1	1	1	0	159	139	139	239	...	0	0	0

pupilDiamMax	pupilDiamLag	timePrtctg	nWordsInTitle	titleNo	wordNo	label
0.0095	0.145	0.0131	7	4	3	0
0.0095	0.183	0.0363	7	1	1	0
0.0370	0.183	0.0208	7	1	3	0
0.0370	0.183	0.0208	7	1	5	0
0.0390	0.183	0.0182	7	1	6	0

Figure 1: First 5 columns of the eye movements dataset

	lineNo	assgNo	fixcount	firstPassCnt	P1stFixation	P2stFixation	prevFixDur	firstfixDur	firstPassFixDur	nextFixDur	...
count	10936.000000	10936.000000	10936.000000	10936.000000	10936.000000	10936.000000	10936.000000	10936.000000	10936.000000	10936.000000	...
mean	5468.500000	167.904353	1.214338	1.200165	0.805596	0.402432	156.423830	166.871982	195.853877	167.873903	...
std	3157.095606	97.304614	0.523573	0.494028	0.395759	0.490411	79.743864	74.807895	107.390201	77.748977	...
min	1.000000	1.000000	1.000000	1.000000	0.000000	0.000000	0.000000	20.000000	20.000000	0.000000	...
25%	2734.750000	83.000000	1.000000	1.000000	1.000000	0.000000	99.000000	119.000000	119.000000	119.000000	...
50%	5468.500000	164.000000	1.000000	1.000000	1.000000	0.000000	139.000000	159.000000	179.000000	159.000000	...
75%	8202.250000	256.000000	1.000000	1.000000	1.000000	1.000000	199.000000	199.000000	239.000000	199.000000	...
max	10936.000000	336.000000	12.000000	8.000000	1.000000	1.000000	1036.000000	1036.000000	1628.000000	1133.000000	...

regressLen	nextWordRegress	regressDur	pupilDiamMax	pupilDiamLag	timePrtctg	nWordsInTitle	titleNo	wordNo	label
10936.000000	10936.000000	10936.000000	10936.000000	10936.000000	10936.000000	10936.000000	10936.000000	10936.000000	10936.000000
531.124817	0.219184	300.377195	0.218395	0.329608	0.030666	5.900421	4.889539	2.924104	0.914594
1935.852983	0.413713	723.222017	0.371121	0.455328	0.026749	1.543177	2.883194	1.878208	0.776556
0.000000	0.000000	0.000000	-4.325500	-4.013600	0.000500	2.000000	1.000000	1.000000	0.000000
0.000000	0.000000	0.000000	0.070800	0.123450	0.015300	5.000000	2.000000	1.000000	0.000000
0.000000	0.000000	0.000000	0.145700	0.217300	0.023400	6.000000	4.000000	2.000000	1.000000
219.000000	0.000000	298.000000	0.253625	0.353600	0.036800	7.000000	7.000000	4.000000	2.000000
35448.000000	1.000000	11140.000000	4.491700	4.491700	0.362300	10.000000	10.000000	10.000000	2.000000

Figure 2: Brief description of each of the attributes in the dataset

The dataset consisted of both discrete and continuous data, as shown in figure 2 with a large difference between the 25% and 75% percentiles for each continuous attribute.

3. Pre-processing

We first wanted to look into how much information was missing. If data was missing from our 27 attributes, the missing data would be replaced by the average of the remaining data if the attribute was continuous or we would fill the missing data with the most common value for an attribute if it was discrete. However, none of the data was missing so data imputation wasn't necessary. We then wanted to clean up some of the data by removing attributes that did not have any predictive value—attributes that consisted of only index values, and, thus, had solely unique values for each instance.

```
{'lineNo': 10936,
 'assgNo': 336,
 'fixcount': 8,
 'firstPassCnt': 7,
 'P1stFixation': 2,
 'P2stFixation': 2,
 'prevFixDur': 61,
 'firstfixDur': 63,
 'firstPassFixDur': 111,
 'nextFixDur': 68,
 'firstSaccLen': 9548,
 'lastSaccLen': 9350,
 'prevFixPos': 7866,
 'landingPos': 6847,
 'leavingPos': 6900,
 'totalFixDur': 149,
 'meanFixDur': 254,
 'nRegressFrom': 6,
 'regressLen': 572,
 'nextWordRegress': 2,
 'regressDur': 381,
 'pupilDiamMax': 3810,
 'pupilDiamLag': 2517,
 'timePrtctg': 1065,
 'nWordsInTitle': 9,
 'titleNo': 10,
 'wordNo': 10,
 'label': 3}
```

Figure 3: Number of unique values for each attribute

As demonstrated in figure 3, the line number column has a unique value for every instance, and, upon further examination, we determined that it was an index column. Thus, we dropped it for our training and testing sets. Although the title number and word

number only had 10 unique values, we also dropped those columns since their purposes were also for indexing.

Once we took out attributes from the original dataset, we split the data into a training set (80% of the data) and a testing set (20% of the data).

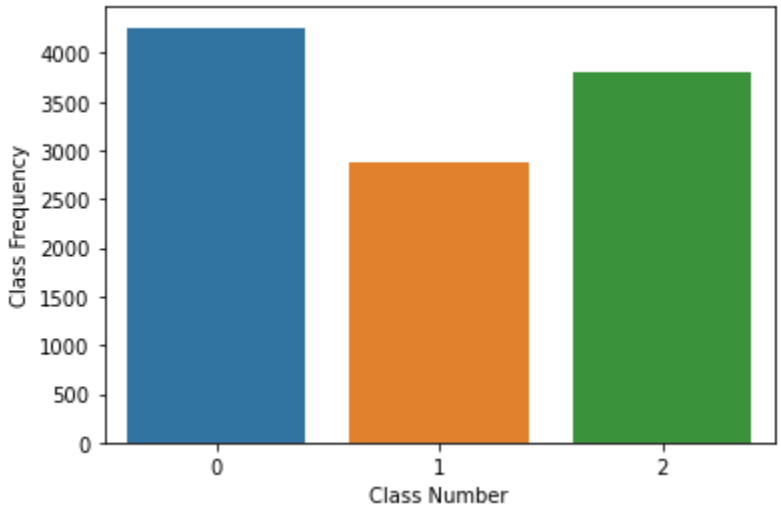

Figure 4: Class distribution in the original dataset

Roughly 35% of the data had 0 as the class variable, 38% of the data had 1 as the class variable, and the remaining 26% of the data had 2 as the class variable. We wanted to maintain the same class distribution from our original dataset in our training and testing sets. To do this, we used Scikit-learn's "train_test_split" method to split 80% of the data (8748 instances) into the training set and 20% of the data into the testing set. Furthermore, we passed in the original data's class column as a parameter for stratification while splitting the data to ensure that the class distribution was maintained.

4. Attribute Selection

For attribute selection, we used 5 algorithms: Pearson Correlation Coefficient, Recursive Feature Elimination with Cross-Validation (RFECV) with RandomForests, LassoCV, RFECV with GradientBoosting, and a LightGBM Selector. For the Pearson Correlation Coefficient, we checked the correlation coefficients for all values with respect to the class variable, and then selected those with a high enough correlation. We then checked to make sure that none of the variables were highly correlated with each other and removed those that were. Unfortunately, our variables were not well correlated with the class variable; the highest correlation we found had a value of 0.228493.

$$r = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum (x_i - \bar{x})^2 \sum (y_i - \bar{y})^2}}$$

Figure 5: Pearson Correlation Coefficient formula

The RFECV algorithms selected about 10-15 attributes each, running through the RandomForests and GradientBoosting algorithms and selecting the attributes that performed the best. Finally, the LightGBM selector used a LightGBM classifier to run through the dataset and evaluate the performance of each attribute. This algorithm also selected 10 attributes to proceed to the classification stage.

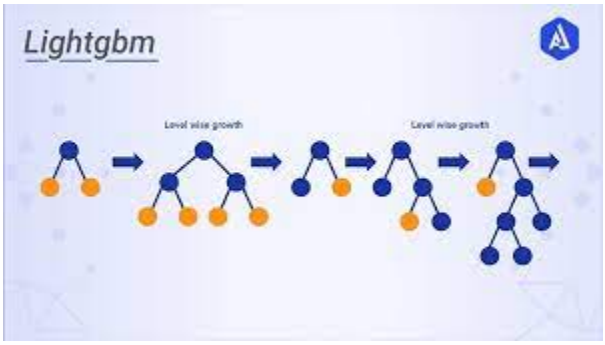

Figure 6: LightGBM classifier structure (Source)

5. Classification Algorithms

Initially, we experimented with many supervised learning algorithms, including DecisionTrees, K-Nearest Neighbors, Support Vector Machines, Neural Networks, and Naive Bayes. Preliminary analysis showed that, out of all of these, tree-based ensemble learning models worked the best, so we mainly explored variations of these algorithms. Namely, we experimented with RandomForests, ExtraTrees, Bagging, AdaBoost, and GradientBoosting ensemble classifiers. We utilized Scikit-learn's implementations for each of our classifiers.

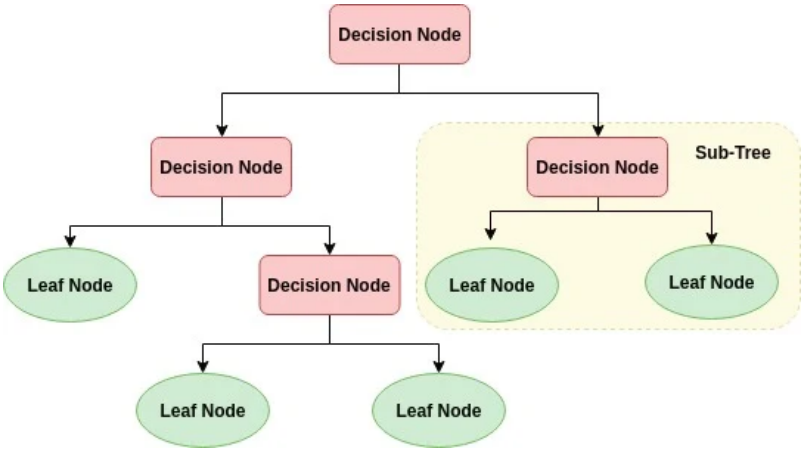

Figure 7: General Decision Tree structure, modified to make tree-based ensemble learning algorithms (Source)

6. Results & Analysis

The accuracy and confusion matrix of a model on the testing set were the primary metrics used to compare algorithms. We utilized accuracy since we did not face a class imbalance problem; however, if there was a class imbalance, we likely would have used sensitivity, specificity, and F1 score. Our best performance was the model that utilized RFECV with RandomForests for attribute selection combined with the ExtraTrees classifier, which resulted in an accuracy of 67.3%. As shown in the confusion matrix, this model was able to predict "Correct Answer" correctly most of the time, but had trouble distinguishing between "Relevant" and "Irrelevant."

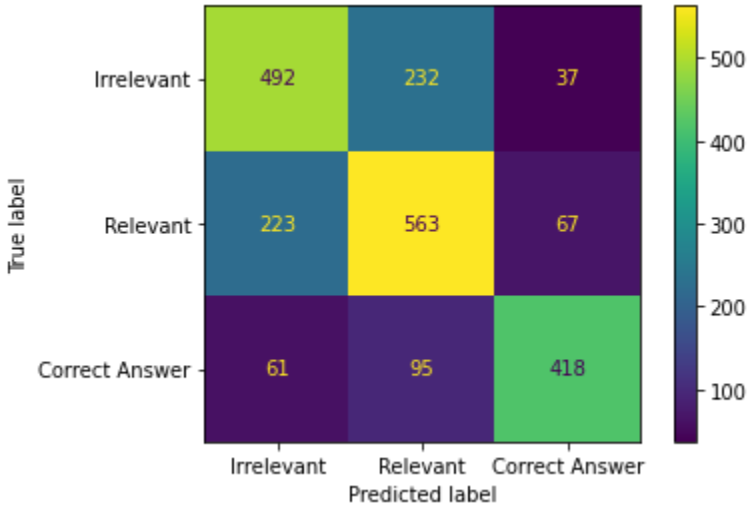

Figure 8: Confusion matrix for RFECV with Random Forests Attribute Selector and ExtraTrees Classifier model

In general, the attribute selection methods that performed the best were the RFECV algorithms, followed by the LightGBM Selection algorithm, the LassoCV algorithm, and the Pearson correlation coefficient selection. This is likely because our dataset was very uncorrelated in one dimension; LassoCV and Pearson correlation coefficient both depend on a linear association between attributes, and a low correlation between attributes would result in these models performing poorly. Furthermore, the tree-based models outperformed neural networks (as we determined in our preliminary analysis) because our dataset has nearly twice as many discrete features as continuous features and consists of ultra-clean, tabular data—tree-based models are known to work extremely well in this type of environment.

```
Classifier: RandomForestClassifier; Feature Selection: Pearson correlation coefficient [5 features] -> 43.3%
Classifier: RandomForestClassifier; Feature Selection: RFECV w/ RandomForests [22 features] -> 63.0%
Classifier: RandomForestClassifier; Feature Selection: LassoCV  [12 features] -> 50.7%
Classifier: RandomForestClassifier; Feature Selection: RFECV w/ GradientBoost [22 features] -> 61.0%
Classifier: RandomForestClassifier; Feature Selection: LightGBM Selection [10 features] -> 54.6%

Classifier: ExtraTreesClassifier; Feature Selection: Pearson correlation coefficient [5 features] -> 42.8%
Classifier: ExtraTreesClassifier; Feature Selection: RFECV w/ RandomForests [22 features] -> 67.3%
Classifier: ExtraTreesClassifier; Feature Selection: LassoCV  [12 features] -> 50.4%
Classifier: ExtraTreesClassifier; Feature Selection: RFECV w/ GradientBoost [22 features] -> 65.6%
Classifier: ExtraTreesClassifier; Feature Selection: LightGBM Selection [10 features] -> 56.7%

Classifier: BaggingClassifier; Feature Selection: Pearson correlation coefficient [5 features] -> 43.4%
Classifier: BaggingClassifier; Feature Selection: RFECV w/ RandomForests [22 features] -> 58.4%
Classifier: BaggingClassifier; Feature Selection: LassoCV  [12 features] -> 51.0%
Classifier: BaggingClassifier; Feature Selection: RFECV w/ GradientBoost [22 features] -> 59.2%
Classifier: BaggingClassifier; Feature Selection: LightGBM Selection [10 features] -> 53.0%

Classifier: AdaBoostClassifier; Feature Selection: Pearson correlation coefficient [5 features] -> 44.1%
Classifier: AdaBoostClassifier; Feature Selection: RFECV w/ RandomForests [22 features] -> 53.2%
Classifier: AdaBoostClassifier; Feature Selection: LassoCV  [12 features] -> 48.2%
Classifier: AdaBoostClassifier; Feature Selection: RFECV w/ GradientBoost [22 features] -> 53.2%
Classifier: AdaBoostClassifier; Feature Selection: LightGBM Selection [10 features] -> 49.4%

Classifier: GradientBoostingClassifier; Feature Selection: Pearson correlation coefficient [5 features] -> 46.9%
Classifier: GradientBoostingClassifier; Feature Selection: RFECV w/ RandomForests [22 features] -> 57.8%
Classifier: GradientBoostingClassifier; Feature Selection: LassoCV  [12 features] -> 51.0%
Classifier: GradientBoostingClassifier; Feature Selection: RFECV w/ GradientBoost [22 features] -> 58.6%
Classifier: GradientBoostingClassifier; Feature Selection: LightGBM Selection [10 features] -> 51.6%
```

Figure 8: Accuracies for each model combination used

7. Conclusion

In conclusion, using an ExtraTrees classifier with an RFECV attribute selector, we were able to successfully predict, with a 67.3% accuracy, how different eye movements indicate what information is relevant to a person when reading a text and retrieving information (answering questions) from that text. Despite having a relatively low accuracy, this model can be used for lie detection, in conjunction with other methods; it can be used to determine if a person who is being interrogated is telling the truth about a written statement or a piece of evidence after reading it, for example. When used in conjunction with other methods, this model can be used as a supplement for lie detection, or, possibly, as a preliminary method to determine whether the person being interrogated believes

that what they are reading is completely correct or merely tangential, since our model was able to predict the “Correct Answer” label quite well.

Throughout this project, we learned a lot about the theory behind various machine learning algorithms as well as data analysis techniques. We realized that most of our time involves exploring the data (data preprocessing) and using that knowledge to perform attribute selection rather than coding the learning algorithms since libraries such as Scikit-Learn, TensorFlow, and WEKA have abstracted these concepts.

8. How To Reproduce Our Model

To reproduce our model, simply clone our GitHub repository and run the `full_model.ipynb` Jupyter Notebook. Running this file can be done by opening it in VSCode and downloading any necessary extensions to run .ipynb files, and then clicking the run button to run it.

This file, along with our data, modeling experiments, and the rest of our code for the project, can be found in our GitHub repository.

9. Team Member Tasks

We split the work roughly evenly. Sameer worked on preprocessing, exploratory data analysis to better understand the data, writing code for the supervised learning algorithms including Decision Trees, K-Nearest Neighbors, and Naive Bayes, and experimenting with different models and conducting preliminary analysis on which type of classifiers would be the most effective. Aditya mainly focused on implementing various attribute selection algorithms and writing code for the tree-based classification algorithms. We also helped each other whenever we were stuck.

10. Sources

References

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., and Liu, T.-Y. (2017). LightGBM: a highly efficient gradient boosting decision tree. *31st International Conference on Neural Information Processing Systems*, 3149–3157.

Salojarvi, J., Puolamaki, K., Simola, J., Kovanen, L., Kojo, I., Kaski, S. (2005). *Inferring relevance from eye movements: Feature extraction*. *Publications in Computer and Information Science*, *A82*. http://www.cis.hut.fi/eyechallenge2005/

Pedregosa F. (2011). *Scikit-learn: Machine Learning in Python*, *JMLR, 12*, 2825-2830.

Ye, A. (2020, September 9). *When and why tree-based models (often) outperform neural networks*. Medium. Retrieved October 23, 2021, from https://towardsdatascience.com/when-and-why-tree-based-models-often-outperform-neural-networks-ceba9ecd0fd8.