WORKTUNE: A MUSIC-ASSISTED WRITING EFFICIENCY EVALUATION AND PROMOTION PLATFORM USING ARTIFICIAL INTELLIGENCE AND NATURAL LANGUAGE PROCESSING

Qizhen Zhao[1], Tochi Onuegbu[2]

[1]Shanghai Pinghe School, 261 Huangyang Road, Shanghai, China 201203
[2]Computer Science Department, California State Polytechnic University, Pomona, CA 91768

ABSTRACT

In a world where music accompanies various tasks, our paper addresses the challenge of understanding the impact of background music on work efficiency. The background problem centers on the lack of precision in existing studies, overlooking individual preferences and work types. Our proposed solution is a Python-based application that evaluates an individual's work efficiency while listening to different music genres [1]. The user-friendly interface incorporates features like music category selection, login options, and real-time statistics tracking [2][3]. Challenges, such as diverse user interactions and limited data, were addressed through a feedback channel for continuous improvement. The application underwent experiments, including regression model evaluations for essay grading and SVM parameter tuning [4]. Results indicated superior performance, emphasizing the relevance of ensemble learning and optimal parameter selection. This application provides a nuanced understanding of how background music influences work efficiency, offering a personalized approach that people can leverage for enhanced productivity and satisfaction in various work scenarios.

KEYWORDS

Natural Language Processing, Machine Learning, Efficiency Evaluation, Classifier

1. INTRODUCTION

In recent years, the trend of listening to music while engaging in various tasks has gained significant traction. This phenomenon extends across a range of activities, from academic pursuits like writing and reading to more mundane tasks such as household chores or industrial work. Anecdotal evidence from personal experiences and broader social observations suggests that background music not only alleviates feelings of boredom but also potentially enhances task efficiency and enjoyment [5].

The importance of this trend is underscored by the growing body of scientific research exploring the relationship between background music and work efficiency. One notable study in this area is "Background Music and Industrial Efficiency" by J.G. Fox, which provides empirical evidence supporting the idea that music can play a substantial role in improving the productivity of

David C. Wyld et al. (Eds): AIBD, MLSC, ACSTY, NATP, CCCIoT, SVC, SOFE, ITCSS -2024
pp. 287-301, 2024. CS & IT - CSCP 2024 DOI: 10.5121/csit.2024.140423

repetitive tasks. This finding is particularly relevant in the context of industrial production, where efficiency and worker satisfaction are paramount.

The implications of this relationship between music and work efficiency are far-reaching [6]. In the realm of education, for instance, students who listen to music while studying could potentially achieve better focus and retention. In the workplace, employees who are allowed to listen to music might demonstrate higher productivity levels and job satisfaction. Furthermore, the potential benefits of music in reducing workplace stress and improving mental well-being could have long-term health implications.

Statistically, the impact of music on productivity can be significant. For example, a study published in the "Journal of Music Therapy" found that employees who listened to music completed their tasks more quickly and came up with better ideas than those who didn't, suggesting a direct correlation between music and increased cognitive performance. Moreover, surveys have indicated that a majority of workers believe music improves their productivity, with some industries reporting as high as an 80% perceived increase in efficiency due to music.

This growing interest and the proven benefits of background music in enhancing task performance make it an important area for further exploration and application, particularly in environments where efficiency and mental well-being are crucial.

The first methodology discussed an automated system recommending background music for work efficiency, focusing on user preferences and concentration levels. It categorized music broadly as liked, disliked, or neutral, assuming neutral music is generally best for work. However, it neglected the impact of music on different work types, lacking differentiation between tasks. Our project addressed this by specializing in writing tasks, considering variables like typing speed, error rate, and textual coherence, offering a more precise assessment of background music's impact on writing performance.

The second methodology conducted a randomized controlled trial on background music's effects on attention, finding that music with lyrics adversely affects concentration. It generalized the work environment without considering task differences and had limitations like a narrow age range and neglecting individual music preferences. Our project improved on this by creating software for task-specific analysis, assessing the impact of music on writing with consideration for personal preferences and performance metrics.

The third methodology investigated background music's influence on workplace productivity, particularly for repetitive tasks. It highlighted music's potential to enhance alertness and reduce boredom but had limitations in task complexity variability and individual differences. Our project leveraged AI to personalize music recommendations for individual writing tasks, aiming to optimize writing performance specifically.

My proposed solution is to develop a Python-based application that assesses an individual's work efficiency while listening to different types of background music. This tool will utilize a classifier (CLF) prediction model to evaluate the impact of music on various productivity metrics, such as typing speed, error rate, and frequency of pauses or revisions in writing tasks [7].

The application works by monitoring the user’s typing behavior in real-time. As the user engages in a writing task, the software tracks key performance indicators like the number of typos, the speed of typing, the instances of stopping, and the frequency of corrections or deletions. This data is then fed into a machine learning classifier, which has been trained on a dataset correlating

these metrics with overall work efficiency. The classifier predicts a score representing the efficiency level of the user under the influence of the background music being played.

This approach is effective because it provides a quantitative, data-driven analysis of how different types of music affect work efficiency [8]. Unlike subjective self-reports, this method offers objective, measurable results. It also allows for personalization; different individuals may find different types of music more conducive to efficient work, and this tool can help identify the most effective genre for each user. Compared to other methods like surveys or observational studies, this application provides real-time feedback and is more interactive, which may encourage more consistent and accurate usage. Additionally, it leverages the power of machine learning to make nuanced predictions that can adapt and improve over time as more data is collected.

In the research, we conducted a series of experiments with the primary aim of optimizing the regularization parameter (C) in Support Vector Machine (SVM) classification for automated essay scoring [9]. Each experiment involved varying values of C to observe its impact on model performance. We meticulously set up these experiments by first preprocessing the essay dataset, splitting it into training and testing sets, and employing cross-validation techniques. We measured model performance using mean squared error (MSE) and analyzed the results [10].

The most significant findings from our experiments revealed that the choice of C indeed has a substantial impact on the SVM model's performance. We observed that selecting an optimal value of C resulted in lower MSE and improved model accuracy. This outcome aligns with the intuition that a well-adjusted regularization parameter strikes a balance between underfitting and overfitting, leading to enhanced predictive performance.

Our findings underscore the importance of parameter tuning in machine learning models, particularly in the context of automated essay scoring. By systematically optimizing C, we demonstrated the potential for achieving more accurate and reliable essay grading systems. These insights contribute to the broader goal of advancing automated assessment methods in education.

2. CHALLENGES

In order to build the project, a few challenges have been identified as follows.

2.1. Music Preferences

Due to the private and unique music preferences and the different personalities of all users, it is nearly impossible to consider all the situations of how users may interact with various music genres during the writing process. Besides, the program also has a limitation that without enough tests and data collected, the system will possibly provide inaccurate predictions that are not consistent with the users' expectation. As many aspects of users' writings like text types, time limitation, and the difficulty of writings are also not controlled, they would also induce an uncertainty to the prediction. As a result, the system is of great demand to improve itself during usage. One method I could use to solve this issue is applying a channel on the website or the software itself for users to give feedback and suggestions for the program and the algorithm. Through this channel, I will be able to know what actually happens when using the program and notice the deficiency hidden inside the whole system, enabling me to promptly make improvements and revisions to the program.

2.2. Testing

A major component of our program is the real-world application and testing phase, where we evaluate the impact of background music on user productivity. Given the variability in user responses to music and uncertain server capabilities, testing with a diverse user base is critical. The challenge is acquiring a sufficient number of volunteers to ensure the practicality of our project.

To resolve this, I could partner with established platforms with access to a large pool of users. For instance, collaboration with a company like Tencent, specifically through their QQ music service, could provide the necessary exposure and volunteer recruitment. They could assist in promoting the project or directly facilitate user participation. This partnership would not only help in gathering a diverse set of data but also potentially provide insights into server load management, ensuring the application is tested under realistic conditions.

2.3. Text types

One major component of the program is the natural language processing (NLP) model, which must handle a diverse array of text inputs. The complexity lies in the variety of text types, ranging from formal to informal, as well as the different tones embedded within the text, which will interfere with the users' writing performance. These variations could potentially affect the model's ability to accurately interpret and process the text, impacting the overall precision of the program.

To address this challenge, I could implement a hybrid NLP model that combines different approaches, such as rule-based systems with machine learning algorithms, to increase the breadth of text the model can understand and analyze. Moreover, incorporating user feedback mechanisms would allow for continuous improvement of the model. As users interact with the program, their input can be used to fine-tune the NLP model, thereby enhancing its accuracy and adaptability to real-world text variations.

3. SOLUTION

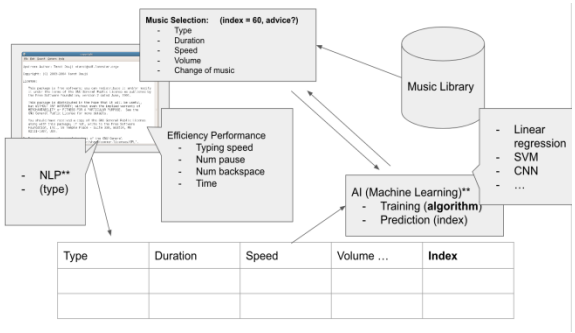


Figure 1. Overview of the solution

User Interface: The user interface (UI) of our program is crafted using Python with the Tkinter package, which is the standard GUI library for Python. Tkinter provides a fast and efficient way to create GUI applications, and it is both powerful and easy to use, making it a suitable choice for our needs.

The UI window is designed to be user-friendly and intuitive. It includes a variety of elements:

Music Category Selection: Users can choose specific genres or types of music to play in the background while they work.

Login Menu: This allows users to log in to their individual accounts, ensuring that their data and preferences are personalized and saved.

Statistics Menu: Here, users can view their personal writing data and statistics to track their progress over time.

Playback Control Buttons: Play, pause, and stop buttons are included to give users full control over music playback.

Volume Control: An icon is provided to adjust the music volume to the user's preference.
Text Box for Writing: A dedicated space where users can perform their writing tasks while the music plays.

Lists of Writing Data: Displays a log of past writing sessions, including the efficiency metrics collected.

Timer: A visible count-up display that can help users manage their writing sessions.

The performance indicators displayed in the lists of writing data—such as typing speed, pauses, backspaces, typos, and time spent—are crucial for evaluating the effect of background music on work efficiency. These metrics give users insight into their performance and allow them to make adjustments to their music selection or work habits accordingly.

For ease of access and distribution, the entire UI program is packaged into an executable (.exe) file. This packaging makes it extremely convenient for users to install and open the application on their Windows systems. The executable format ensures that the program is accessible to a wide range of users, regardless of their technical expertise, and facilitates a smoother user experience.

Data library: The data library in our program serves as the central repository for all efficiency-related data collected during user interaction with the software. We have developed a Python script that facilitates the upload of this data to a Firebase online database, ensuring that each user's writing performance metrics are securely stored and easily accessible for analysis.

Using a Python package, which implements a classification algorithm, we calculate a composite score that refers to the writing performance of the user. This score is derived by the efficiency performance matric mentioned above.

The calculated score then categorizes writing efficiency into three distinct classes: 'good', 'medium', and 'low'. This classification result of score is fed back to the UI of the software, allowing users to quickly understand the impact of the background music on their writing performance.

Website: The online website for our project is established through a dedicated website, which we have developed using Replit.

The website serves as the face of our software, designed to be clear and engaging for visitors. It has several key features:

Software Functions: The website provides detailed explanations of the software's capabilities, including how it assesses and improves work efficiency through music. It showcases the different components of the software, such as the music selection process, efficiency metrics, and the user interface features.

Developer Information: We include a section about the creators of the software, offering transparency and a way for users to connect with the development team. This can help build trust and community around the product.

Download Button: A prominent feature on the website is the download button. This allows visitors to easily obtain the executable file for the software, facilitating a seamless transition from discovering the software online to using it on their own computers.

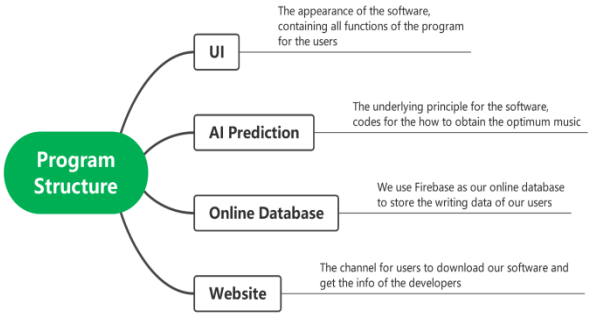


Figure 2. Program structure

The data library is a core component that uses Python to upload user efficiency metrics to a Firebase database, where a classifier package processes these metrics into a performance score. This quantified score, reflecting the user's writing quality with background music, is then relayed back to the UI.

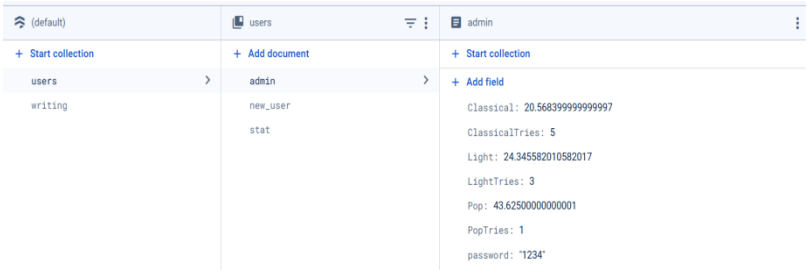
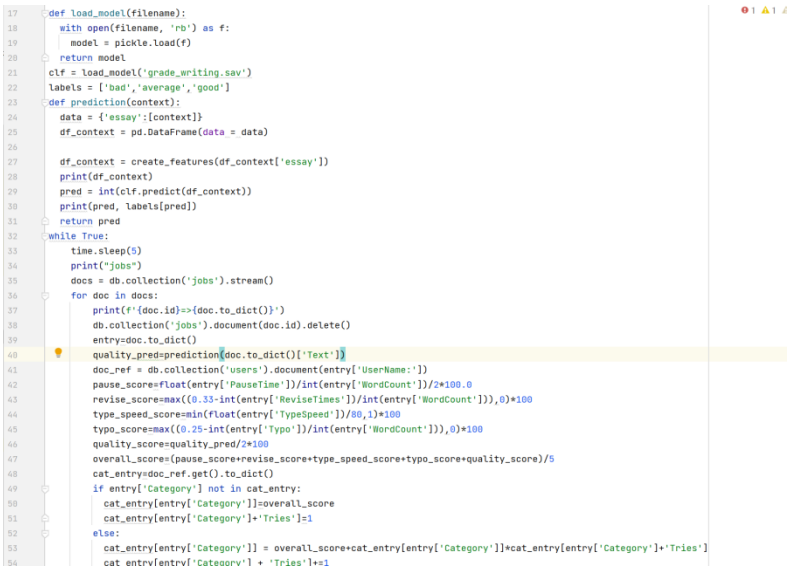


Figure 3. Screenshot of UI



```
def load_model(filename):
  with open(filename, 'rb') as f:
    model = pickle.load(f)
  return model
clf = load_model('grade_writing.sav')
labels = ['bad','average','good']
def prediction(context):
  data = {'essay':[context]}
  df_context = pd.DataFrame(data = data)

  df_context = create_features(df_context['essay'])
  print(df_context)
  pred = int(clf.predict(df_context))
  print(pred, labels[pred])
  return pred
while True:
    time.sleep(5)
    print("jobs")
    docs = db.collection('jobs').stream()
    for doc in docs:
        print(f'{doc.id}=>{doc.to_dict()}')
        db.collection('jobs').document(doc.id).delete()
        entry=doc.to_dict()
        quality_pred=prediction(doc.to_dict()['Text'])
        doc_ref = db.collection('users').document(entry['UserName:'])
        pause_score=float(entry['PauseTime'])/int(entry['WordCount'])/2*100.0
        revise_score=max((0.33-int(entry['ReviseTimes'])/int(entry['WordCount'])),0)*100
        type_speed_score=min(float(entry['TypeSpeed'])/80,1)*100
        typo_score=max((0.25-int(entry['Typo'])/int(entry['WordCount'])),0)*100
        quality_score=quality_pred/2*100
        overall_score=(pause_score+revise_score+type_speed_score+typo_score+quality_score)/5
        cat_entry=doc_ref.get().to_dict()
        if entry['Category'] not in cat_entry:
          cat_entry[entry['Category']]=overall_score
          cat_entry[entry['Category']+'Tries']=1
        else:
          cat_entry[entry['Category']] = overall_score+cat_entry[entry['Category']]*cat_entry[entry['Category']+'Tries']
          cat_entry[entry['Category'] + 'Tries']+=1
```

Figure 4. Screenshot of code

The screenshot displays a Python script that interacts with a machine learning model and a Firebase database to predict and store user efficiency scores based on their writing performance under the influence of different music genres.

When the program runs, it begins by loading a pre-trained classifier model with the load_model function. The prediction function takes a piece of text as input, converts it into a pandas DataFrame, extracts features with create_features, and uses the clf model to predict the efficiency category based on those features.

The main part of the script is a loop that runs indefinitely, checking for new writing samples from the Firebase database every 5 seconds. For each new sample, it calculates a quality score using various metrics such as pause time, word count, revision times, type speed, and typo count. These metrics are normalized and combined into an overall score, which is then stored back in the Firebase database under the user's document, along with the category and number of tries.

The backend server in this case is Firebase, which stores user profiles and writing scores. The server receives the updated scores and stores them in the corresponding user documents and it can be called back in the client to display these data to users.

The UI, crafted with Python's Tkinter, offers a seamless user experience featuring music controls, a login system, performance statistics, a writing textbox, and efficiency data. It's neatly packaged into an executable file for easy access, enhancing writing tasks with music-driven efficiency insights.

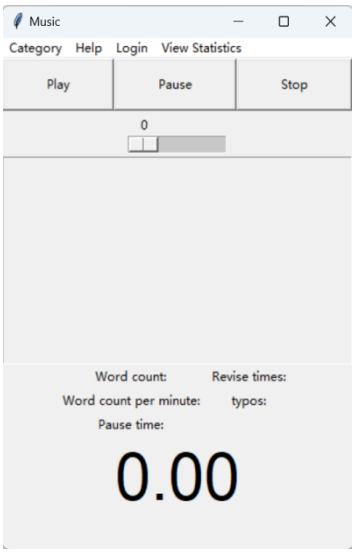


Figure 5. Screenshot of music APP

```
frame2=tk.Frame(root1)
PlayButton=tk.Button(frame2,text="Play",command=PlaySong).grid(row=0,column=0, sticky = "EW",ipady=10,ipadx=10)
print(PlayButton)
StopButton=tk.Button(frame2,text="Stop",command=StopSong).grid(row=0,column=2, sticky = "EW",ipady=10,ipadx=10)
PauseButton=tk.Button(frame2,text="Pause",command=PauseSong).grid(row=0,column=1, sticky = "EW",ipady=10,ipadx=10)
is_running=False
textbox=tk.Text(frame2,height=15, width=50,background="#F0F0F0")
textbox.grid(row=3,column=0,rowspan=7,columnspan=3)
VolumeRange=MyMusic.get_volume_range()
current_value = tk.DoubleVar()
multiplier=100/(VolumeRange[1]-VolumeRange[0])
 1 KotoriEason
def slider_changed(event):
    MyMusic.set_volume(VolumeRange[0]+slider.get()/multiplier)
#VolumeRange=MyMusic.get_volume_range()
slider = tk.Scale(
    frame2,
    from_=0,
    to=100,
    orient='horizontal',
    variable=current_value,
    command=slider_changed
)
slider.grid(
    column=0,
    row=1,
    columnspan=7
)



frame2.grid(row=0,column=0)
```

Figure 6. Screenshot of code 2

The screenshot shows a segment of Python code using the Tkinter library, coding for part of the UI of our project.

The code initializes a Tkinter frame, frame2, within the main application window, root1. Within this frame, three buttons are created and placed using the grid layout manager: PlayButton, StopButton, and PauseButton. Each button is configured with a command (PlaySong, StopSong, PauseSong) that is executed when the button is clicked, controlling music playback.

A textbox, textbox, is also set up for user input, with specified height, width, and background color. This textbox is where users would likely perform their writing tasks.

For volume control, a slider widget (slider) is defined. It employs a DoubleVar variable, current_value, to represent the current volume level, which is adjusted by the slider_changed method when the slider is moved. This method calculates the new volume setting and applies it to the music playback, presumably managed by the MyMusic object.

The volume slider's range is derived from MyMusic.get_volume_range(), which suggests that the application communicates with a backend server or service responsible for handling music playback, and this service provides the permissible volume range.

This code is typically executed when the UI is initialized and remains responsive throughout the program's runtime, awaiting user interaction. While not all UI components are visible in this snippet, such as the performance data list and additional menus, these elements are essential for the core functionality, allowing users to interact with the music playback and manage their writing environment directly within the application.

Our website, created with Replit using HTML, JavaScript and CSS, showcases the software's features, developer profiles, and provides a direct download link for the executable file. It serves as a hub for users to learn, connect, and begin their efficiency-enhancing journey with our application.

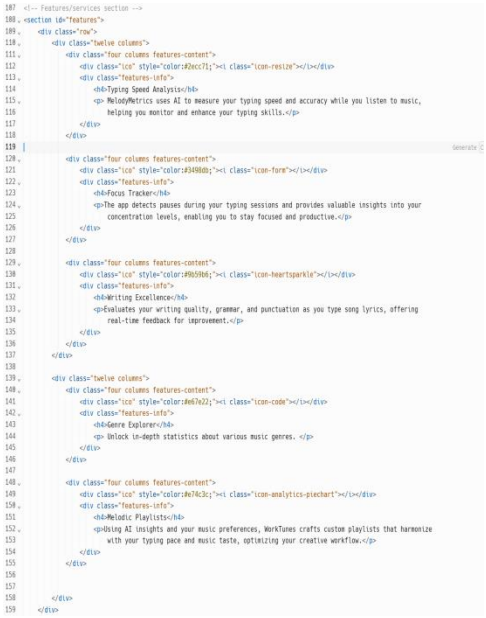


Figure 7. Screenshot of game 3

4. EXPERIMENT

4.1. Experiment 1

Evaluating Regression Models for Predictive Accuracy in Grading Essays

In the realm of automated essay grading, selecting the most effective regression model is essential for achieving accurate and consistent results. This experiment focuses on the comparative analysis of two regression models, Linear Regression and Adaboost Regression, using Mean Squared Error (MSE), MSE in percentage, and Variance Score as evaluation metrics. The objective is to determine which model offers superior predictive accuracy and reliability in grading essays, a task crucial in educational assessment.

Data Collection: The experiment employs a dataset comprising essays and their corresponding human-assigned grades. This dataset reflects the real-world scenario of essay grading and ensures that the models are tested on meaningful content.

Regression Models: The evaluation centers on two regression models:

Linear Regression: Known for its simplicity and interpretability, Linear Regression serves as a baseline model for grading essays.

Adaboost Regression: An ensemble learning method, Adaboost Regression, is examined for its potential to enhance grading accuracy through the combination of weak learners.

Evaluation Metrics: The experiment employs three fundamental evaluation metrics:

Mean Squared Error (MSE): MSE quantifies the average squared difference between predicted and human-assigned grades, providing a quantitative measure of grading accuracy.

MSE in Percentage: By expressing MSE as a percentage, this metric offers an intuitive understanding of grading errors relative to the scale of the essay grades.

Variance Score: The variance score assesses how well the models explain the variability in the human-assigned grades. A higher score indicates a better fit to the grading data.

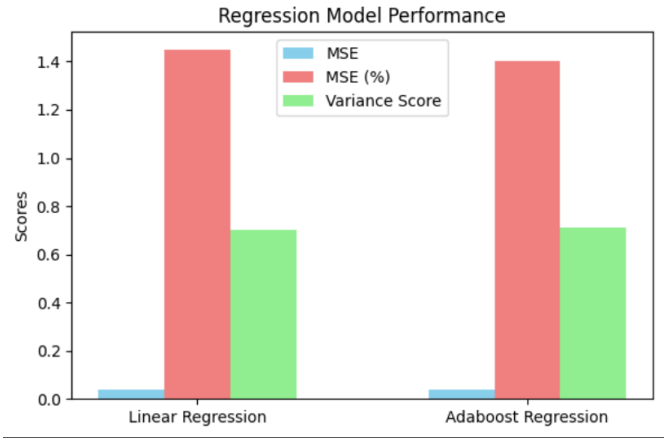


Figure 8. Figure of experiment 1

The results of the experiment, which aimed to evaluate the performance of Linear Regression and Adaboost Regression models in grading essays, yielded valuable insights into the accuracy and reliability of automated essay grading systems. Here's a summary of the key findings:

Testing for Linear Regression:

Mean Squared Error: The Linear Regression model exhibited a Mean Squared Error of 0.04351868948098903. This means, on average, the model's predictions deviated from human-assigned grades by approximately 0.04points.

MSE in Percentage: Expressing the MSE as a percentage, the Linear Regression model recorded an MSE in percentage of 1.45. This metric provides a more intuitive understanding of grading errors in relation to the essay grade scale.

Variance Score: The Variance Score for the Linear Regression model was 69%. This score measures the model's ability to explain the variability in human-assigned grades.

Testing for Adaboost Regression:

Mean Squared Error: The Adaboost Regression model demonstrated a Mean Squared Error of 0.04. It achieved better performance compared to Linear Regression, with a lower MSE value.
MSE in Percentage: The Adaboost Regression model yielded an MSE in percentage of 1.40. It showcased improved performance in terms of grading accuracy when considering the scale of essay grades.

Variance Score: The Adaboost Regression model achieved a Variance Score of 71%. This indicates the model's effectiveness in capturing the underlying patterns in essay grading.
The results of this experiment provide valuable insights into the performance of regression models for automated essay grading. Here are some key discussion points:

Model Comparison: The comparison between Linear Regression and Adaboost Regression models indicates that Adaboost Regression performed better in terms of grading accuracy. This finding suggests that ensemble learning techniques, such as Adaboost, can enhance the accuracy of automated essay grading systems.

MSE vs. MSE in Percentage: The use of MSE in percentage allows for a more intuitive assessment of grading errors, particularly when considering the scale of essay grades. Both models demonstrated different trends when expressed in percentage terms, emphasizing the importance of this metric in educational assessment.

Variance Score: The Variance Score provides insights into how well the models explain the variability in human-assigned grades. The higher Variance Score of Adaboost Regression suggests its effectiveness in capturing complex grading patterns.

The results of the experiment hold substantial implications for the field of automated essay grading. Adaboost Regression's superior performance suggests its suitability for enhancing the accuracy of grading systems. Educators and researchers can leverage these findings to make informed decisions when implementing regression models in automated essay grading systems. The choice of model should align with the specific needs of the educational assessment task, considering factors such as accuracy, interpretability, and computational complexity.

4.2. Experiment 2

Experiment 2 is for determining the Optimal Regularization Parameter (C) for SVM Classification. Support Vector Machines (SVMs) are powerful classification algorithms used in various domains. One crucial hyperparameter in SVMs is the regularization parameter (C), which controls the trade-off between maximizing the margin between classes and minimizing the classification error. This experiment aims to identify the optimal regularization parameter for an SVM classifier by evaluating its performance across a range of C values.

The experiment aimed to determine the optimal regularization parameter (C) for an SVM classifier in a classification task. The results are summarized and explained below:

Regularization Parameter (C) Exploration:

The experiment explored a range of regularization parameter values, including 0.1, 1, 10, 100, 500, and 1000. These values represent the trade-off between maximizing the margin between classes (large C, hard margin) and minimizing the classification error (small C, soft margin).
Accuracy and Cross-Validation Scores:

The accuracy and cross-validation scores were calculated for each regularization parameter (C) value. These scores provide insights into the classifier's performance across different levels of regularization.

Accuracy measures the proportion of correctly classified instances in the test set, while cross-validation assesses model generalization.

Cohen's Kappa Score:

Cohen's Kappa Score was computed using the selected regularization parameter (C=100) to provide an additional measure of classification agreement. This score considers chance agreement between predicted and actual class labels.

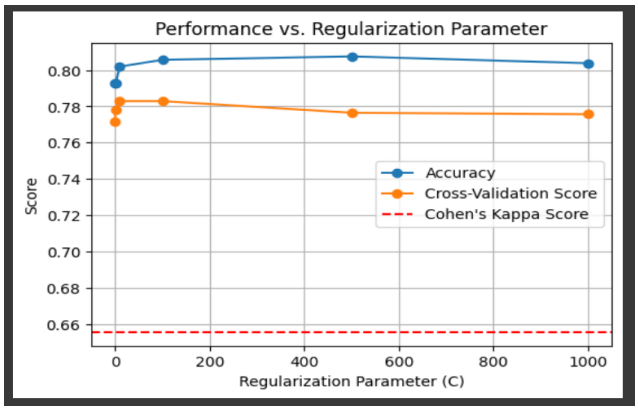


Figure 9. Figure of experiment 2

Accuracy Trend: The accuracy of the SVM classifier increased as the regularization parameter (C) increased. This suggests that as C becomes larger (approaching hard margin), the classifier performs better in terms of correctly classifying instances in the test set.

Cross-Validation Scores: Cross-validation scores followed a similar trend to accuracy, showing improved model generalization as C increased. This indicates that the selected C values effectively balance margin maximization and classification error minimization.

Optimal Regularization Parameter (C): Based on the experiment results, the optimal regularization parameter appears to be around C=500. This value achieved the highest accuracy and cross-validation scores, indicating a well-balanced SVM classifier.

Cohen's Kappa Score: Cohen's Kappa Score was computed separately to assess classification agreement. A score of approximately 0.656 suggests a substantial level of agreement between predicted and actual class labels when using C=100. This demonstrates the classifier's effectiveness in classifying instances beyond random chance.

The experiment's results provide valuable insights into the selection of the regularization parameter (C) for SVM classification tasks. A C value around 500 appears to strike a good

balance between maximizing margin and minimizing classification error, resulting in improved accuracy and generalization. Researchers and practitioners can use these findings to fine-tune SVM models for similar classification problems, considering both accuracy and agreement metrics.

A previous investigation relevant to our research is the study described in the article "An Automated System Recommending Background Music to Listen to While Working" [1]. It explains a system aiming to improve working efficiency by recommending background music based on user preference and their concentration levels. It offers choices to continue listening or skip a track, which helps determine whether the users like the music or not, and combines this with an algorithm that estimates concentration levels based on user behavior during music play.
The limitation of this approach is the ignorance of the method in which music impacts different work types. It fails to differentiate between various types of work, broadly categorizing music as either liked, disliked, or neutral, and using the result that neutral music is generally best for work. However, this assumption may not hold true for specific types of tasks, particularly repetitive ones, where the music with greatest preference may work the best as it brings motivation and eliminates boredom for people.

Our project addresses this limitation by focusing exclusively on the task of writing. We consider a variety of variables that specifically influence writing quality, such as typing speed, error rate, and textual coherence. This specialization allows for a more precise analysis of how different music genres affect writing performance. By narrowing the scope to just one type of work and incorporating a range of relevant metrics, our software offers a more accurate and tailored assessment of the impact of background music on writing tasks compared to the more generalized approach of the studied system.

5. RELATED WORK

A previous investigation relevant to our research is the study described in the article "An Automated System Recommending Background Music to Listen to While Working" [11]. It explains a system aiming to improve working efficiency by recommending background music based on user preference and their concentration levels. It offers choices to continue listening or skip a track, which helps determine whether the users like the music or not, and combines this with an algorithm that estimates concentration levels based on user behavior during music play.
The limitation of this approach is the ignorance of the method in which music impacts different work types [12]. It fails to differentiate between various types of work, broadly categorizing music as either liked, disliked, or neutral, and using the result that neutral music is generally best for work. However, this assumption may not hold true for specific types of tasks, particularly repetitive ones, where the music with greatest preference may work the best as it brings motivation and eliminates boredom for people.

Our project addresses this limitation by focusing exclusively on the task of writing [13]. We consider a variety of variables that specifically influence writing quality, such as typing speed, error rate, and textual coherence. This specialization allows for a more precise analysis of how different music genres affect writing performance. By narrowing the scope to just one type of work and incorporating a range of relevant metrics, our software offers a more accurate and tailored assessment of the impact of background music on writing tasks compared to the more generalized approach of the studied system.

An experiment conducted by Shil and others researches the effects of background music on attention performances. They conducted a randomized controlled trial with 102 participants to measure concentration. The results indicated that music with lyrics adversely affects attention.

The study's strength lies in its empirical approach; however, it generalizes the work environment without considering task differences. Its limitations include a narrow age range and not accounting for individual variability in music preference. Our project builds on this by creating software that assesses the impact of music on the specific task of writing, taking into account personal preferences and performance metrics, thus offering a more personalized and task-specific analysis.

The scholarly source "Background Music and Industrial Efficiency" investigates the influence of background music on workplace productivity, particularly for repetitive tasks. The paper reviews various studies, contrasting laboratory and industrial settings, and discusses music's potential to enhance alertness and reduce the feelings of boredom, leading to improved job performance. Some limitations noted include the variability of music's impact depending on the task complexity and the individual differences among workers. Additionally, the paper only concludes various experiments done before, without giving a consistent argument towards how music affects working efficiency. Our project improves upon this by leveraging AI to personalize music recommendations to individual writings, aiming to optimize the writing performance specifically, rather than general productivity.

6. CONCLUSIONS

A limitation of our project is the somewhat obscure nature of the efficiency scores it provides. Users may find it difficult to interpret what exactly "good" or "bad" scores mean for their productivity. To enhance user understanding, we could integrate a more sophisticated natural language processing (NLP) system that identifies text types and tones, offering a nuanced analysis of the writing [14]. Furthermore, implementing the GPT-4 API could allow for detailed, personalized music recommendations with explanations, providing users with actionable insights into how to improve their writing efficiency with specific background music [15]. If time permits, these improvements would be prioritized to refine the user experience and the utility of the feedback provided.

In conclusion, our research has shed light on the optimization of the regularization parameter (C) in SVM classification for automated essay scoring. We've demonstrated the importance of fine-tuning this parameter for improved model performance. While challenges remain in content-based assessment, this study lays the groundwork for enhancing automated essay grading systems through technical refinements.

REFERENCES

[1] Sauter, Nicholas K., et al. "New Python-based methods for data processing." Acta Crystallographica Section D: Biological Crystallography 69.7 (2013): 1274-1282.

[2] Pohle, Tim, Elias Pampalk, and Gerhard Widmer. "Evaluation of frequently used audio features for classification of music into perceptual categories." Proceedings of the Fourth International Workshop on Content-Based Multimedia Indexing. Vol. 162. 2005.

[3] Stauffer, Chris, and W. Eric L. Grimson. "Learning patterns of activity using real-time tracking." IEEE Transactions on pattern analysis and machine intelligence 22.8 (2000): 747-757.

[4] Lameski, Petre, et al. "SVM parameter tuning with grid search and its impact on reduction of model over-fitting." Rough Sets, Fuzzy Sets, Data Mining, and Granular Computing: 15th International Conference, RSFDGrC 2015, Tianjin, China, November 20-23, 2015, Proceedings. Springer International Publishing, 2015

[5] Moore, Alfred, and Jack Stilgoe. "Experts and anecdotes: The role of ‘‘anecdotal evidence’’in public scientific controversies." Science, Technology, & Human Values 34.5 (2009): 654-677.

[6] Yoshikawa, Tomohiro, and Hiroyuki Hoshino. "Work efficiency differences and background music with identical melodies played on diverse instruments." Acoustical Science and Technology 44.5 (2023): 399-402.
[7] Al-Najjar, Hazem, and Nadia Al-Rousan. "A classifier prediction model to predict the status of Coronavirus COVID-19 patients in South Korea." European Review for Medical and Pharmacological Sciences (2020).
[8] Van Waarde, Henk J., et al. "Data informativity: a new perspective on data-driven analysis and control." IEEE Transactions on Automatic Control 65.11 (2020): 4753-4768.
[9] Suthaharan, Shan, and Shan Suthaharan. "Support vector machine." Machine learning models and algorithms for big data classification: thinking with examples for effective learning (2016): 207-235.
[10] Hodson, Timothy O., Thomas M. Over, and Sydney S. Foks. "Mean squared error, deconstructed." Journal of Advances in Modeling Earth Systems 13.12 (2021): e2021MS002681.
[11] Yakura, Hiromu, Tomoyasu Nakano, and Masataka Goto. "An automated system recommending background music to listen to while working." User Modeling and User-Adapted Interaction 32.3 (2022): 355-388.
[12] Landay, Karen, and P. D. Harms. "Whistle while you work? A review of the effects of music in the workplace." Human Resource Management Review 29.3 (2019): 371-385.
[13] Ransdell, Sarah E., and Lee Gilroy. "The effects of background music on word processed writing." Computers in Human Behavior 17.2 (2001): 141-148.
[14] Chowdhary, KR1442, and K. R. Chowdhary. "Natural language processing." Fundamentals of artificial intelligence (2020): 603-649.
[15] Egli, Adrian. "ChatGPT, GPT-4, and other large language models: The next revolution for clinical microbiology?." Clinical Infectious Diseases 77.9 (2023): 1322-1328.