**Assessing readability in annual reports of companies using Natural Language Processing and Machine Learning with Python**

*Yanko Mirov, MSci Computer Science*

I certify that the material contained in this dissertation is my own work and does not contain unreferenced or unacknowledged material. I also warrant that the above statement applies to the implementation of the project and all associated documentation. Regarding the electronically submitted version of this submitted work, I consent to this being stored electronically and copied for assessment purposes, including the School’s use of plagiarism detection systems in order to check the integrity of assessed work. I agree to my dissertation being placed in the public domain, with my name explicitly included as the author of the work.

Date: March 24th, 2020

Signed: Yanko Mirov

**Abstract**

The aim of this project is to assess annual reports produced by companies and produce an output which is comprised of the readability score. This score will be used to evaluate how *easy* it is to read the report and what level of education will be required to understand the contents.

**Acknowledgements**

Thanks to El Haj Mahmoud who is my project supervisor for providing me with guidance and thorough feedback.

**Table of contents**

1. **Introduction**
   1. **Project goal**
   2. **Readability analysis overview**
   3. **Motivation**
   4. **Report Overview**
2. **Background Research**
   1. **Measurement of text readability in general texts**
   2. **Measurement of text readability in financial disclosures**
   3. **Fraud detection in the finance industry and how it is relevant to readability**
   4. **Relevant machine learning and natural language processing algorithms and models**
   5. **Supervised learning vs. unsupervised learning**
   6. **Machine learning models**
   7. **Summary of background research**
3. **System design**
   1. **Methodology**
   2. **Software architecture**
   3. **User Interface**
4. **Implementation**
   1. **Dependencies management**
   2. **Dataset and feature selection**
   3. **Feature normalization**
   4. **Splitting data into train and test sets**
   5. **Evaluation of trained models**
   6. **Results and discussion**
   7. **Baseline score**
5. **Conclusion**
   1. **Review of aims**
   2. **Learning outcomes**
   3. **Future work**
   4. **Closing paragraph**

**References**

**Appendix**

**Chapter I. Introduction**

**Project goal**

Assessing readability of texts has been a challenge over the years as critics have used a variety of different methods each having their own drawback. What many, if not most, of the methods have in common is that they rely on static formulas alone. However, as results have shown, accurate readability analysis requires more complex evaluation. This project aims to:

* produce a readability score in the financial domain.
  + The financial reports are in the English language and they are focused on the UK market.
* apply the fine-tuned mathematical formulas that have been used over the years in combination with machine learning to achieve that level of complex analysis required to produce more accurate readability scores.

There is a wide variety of factors taken into consideration when performing readability analysis such as word count, average sentence length, average word length, syllable count, familiarity of words, complexity of sentences, and many more. Thus, it can be a challenge to determine the best combination of factors.

This project will attempt to come up with readability parameters that when fed to a chosen machine learning model, would produce highly accurate readability score.

**Readability analysis overview**

Producing a readability score means an accurate indication of the probability of extracting the correct information from the annual report by the reader [1]. In other words, this accurate extraction would indicate the required education to read the given text. For example, the Texas Department of Insurance has a requirement that all insurance documents have a Flesh Reading Ease score of 40 or higher which would convert to the reading level of a first-year undergraduate student [2].

Since the aim of this project is to produce a readability score, it is important to distinguish that this score is an approximation because a large portion of it is not a qualitative judgement. In the past, it was thought that superficial text features such as sentence length, number of syllables per word, word count, et cetera were factors that could be taken alone and produce a very accurate readability score. However, over the years, it has been established that static readability measurements alone are easy to compute but are “shown to be poorly specified in financial applications” [3].

Throughout the years of readability analysis research, there have been numerous attempts at creating a unique formula which takes different factors into account from the others. Some of the most famous formulas are Flesch reading ease score, Flesch-Kincaid grade level and Gunning fog index [4]. There are small differences between these formulas. The main difference is the parameters they are taking to measure readability and the format of the output – whether it will showcase the grade level, or simply list an arbitrary score.

A low-grade readability score is usually preferred to a high-grade level [5]. Even though judging by the score alone is not enough to determine whether the piece of text has good content or not, it is generally more accessible if the grade level score is lower. In fact, the reason for many of the cases where the score is high-grade level is simply because the author expresses a complex thought that requires long sentences and often, long words [5]. However, that does not necessarily make the piece of text harder to read. Not all long words are hard to read, in fact many of them are very common and known by most education levels. Examples for such words are international, development, technology, organization, opportunity, represent, movement, significant, individual, et cetera. On the other hand, there are also short words that are not very common and not known by many readers. Examples for such words are cavil, descry, pithy, onus, jibe, cabal, et cetera. Hence, how long a word is can often be a misleading metric and should be treated carefully. Therefore, there are many other factors that should be combined when evaluating text readability. Such suggested factors from the literature are purpose of author, genre of text (for example science, therefore long words and sentences likely will be included and should be considered when evaluating the readability), topic, reader’s literacy and more [2][7].

**Motivation**

By producing a highly accurate readability score for annual reports in the financial domain, it would improve productivity by a huge margin, and it would decrease the error rate in analyzing them [1].

In the financial domain, readability affects the communications with shareholders [3]. The financial analysts are the experts who can understand the annual reports. However, if the report is hard to read, it would take a long time to understand the contents. Thus, shareholders will struggle to understand those reports and increase the chances of miscommunication [6]. However, if they knew the readability score beforehand, they would know how to approach the report beforehand. If the readability score is very low (hard to read), it would most likely mean that the financial disclosures are buried in legal jargon and difficult language.

**Report** **Overview**

The structure of this report is split in the following chapters:

* Background Research
  + Discusses the similarities and the differences between commonly used readability tools and formulas. In addition, the subsection also covers commonly used machine learning models, when and how they are useful, supervised vs. unsupervised learning, fraud detection in finance industry and how it is relevant to readability analysis and a summary of the background research.
* Design
  + Provides an overview of the code architecture, e.g. which are the major methods, what do they do, what are the created classes, their relationships, etc. In addition, it briefly discusses what the User Interface of the project is.
* Implementation
  + It is divided into sub-chapters. Each sub-chapter discusses different details about the implementation of the project. It mentions the major libraries and dependencies that the project relies on, including the programming language that has been used and all the libraries, and external tools. Afterwards, there is a sub-chapter dedicated to the dataset this project uses, how it is structured, and which parts of it are used and which are not. The chapter also outlines the baseline method that is used and gives pointers as to what is a base method, why it is used, and why was the particular base method chosen.
* Conclusion
  + The final chapter of the report summarizes the information and points out relevant revisions, the learning outcomes and relevant future work that could extend on the current implementation of the project.

**Chapter II. Background**

The demand for research in text readability has been increasing rapidly. Today there are many applications that can measure readability of general texts. Examples are readable.com, datayze.com and the most popular one – Grammarly. In addition, most of the research on the topic has been carried out on texts that contain mainly words and phrases that are encountered daily and do not contain necessarily legal and financial jargon. In fact, there aren’t any major and/or well-known tools related to measuring readability in financial texts.

Despite Grammarly (a very popular tool used for readability analysis) not being a financial readability tool, it is still of great relevance to this project because it uses machine learning and natural language processing – tools that this project will attempt to adapt. The tool effectively evaluates a given text and produces a variety of useful outputs – alongside which is readability – and gives indication on how to correct them using algorithms and trained models [10]. Grammarly, just like this project, uses Flesch score combined with machine learning to evaluate texts’ readability [11]. The difference is that this project focuses solely on financial disclosures, namely annual reports, whereas Grammarly focuses on more general texts [10].

However, with the amazing progress that has been established on general text readability, many researchers have started exploring applying text readability in more difficult domains – for example, the financial domain – the one that this project is focused on [3][6]. A large portion of the relevant research does not use machine learning and deep learning models but instead relies on static, parameter-tuned formulas, such as the Fog Index [1]. However, the minority of papers that have discussed machine learning in the text readability domain conclude that it is an invaluable tool that offers a great deal of advantages such as extremely accurate and fully automated predictions, reduced data collection costs, increased statistical power via large sample sizes, generalizability, and improved objectivity and replicability [12]. Moreover, the benefits offered from automated text readability analysis far overweigh the benefits offered from static, fine-tuned readability analysis that does not use any machine learning [1].

The background literature for this project is derived from the following areas of research:

* Measurement of text readability in general texts
* Measurement of text readability in financial disclosures
* Fraud detection in the finance industry and how is it relevant to readability
* Relevant machine learning and natural language processing algorithms and models
* Supervised learning vs. unsupervised learning
* Machine learning models

**Measuring text readability in general texts versus measuring readability in financial disclosures**

Through comparing numerous papers [3][4][5][14] that are related to general and financial text readability, it can be deducted that many of the techniques/features that are used in the measurement of the readability of general/simple texts [25], are not effective in the measurement of the readability of financial texts. An example feature can be word complexity – this can be proven useful to general words (e.g. the word *Capitalization* has 6 syllables (therefore hard to read) whereas dog has only 1 syllabus (therefore easy to read)) [6]. However, in the financial domain, that may not be necessarily a correct approach because there are many words in the domain that have many syllables but are easy to read. Therefore, features extraction needs to be very carefully considered for this project because of the sensitivity of the domain.

**Supervised versus unsupervised learning**

In machine learning, there are two types of algorithms – supervised and unsupervised. They are used in different use cases and in every machine learning project, it needs to be carefully considered which type the learning algorithm is going to adapt. There are numerous factors that need to be considered in choosing the correct type of learning. Some of them are [13]:

* Application
* Data
* Preprocessing
* Computational complexity
* Accuracy

The factors above need to be carefully considered to understand the type of machine learning algorithm the project is going to be more suited for. For example, if the Application factor (the first bullet point) is a relation between input and output variables, then Supervised learning should be adapted. However, if the underlying structure is hidden, then Unsupervised Learning would be the preferable option [13].

Having reviewed all the factors, this project will adapt the Supervised type of machine learning because its goal is to predict new samples (annual reports) based on previous data. In addition, during training, both input and output variables are provided to the model. Thus, the algorithms that will be tested are decision tree, support vector machines, logistic regression, and random forests.

**Fraud detection in the finance industry using natural language processing**

At first, fraud detection and financial readability measurement might seem as irrelevant topics. However, there are some practices and analytical concepts that are present in both. In this subsection, these commonalities will be discussed and evaluated.

It is suggested that the management of companies that are not performing well tend to hide bad news in order to delay stock price reduction (this malpractice is named Management Obfuscation Theory) by decreasing the readability of financial reports [14]. A previous paper [14] has shown that there are numerous factors that go into defrauding financial reports. For example, many of the cases, it has been found that it is the management who defraud the reports which makes it even harder for external auditors due to collusion and concealment. Another paper discovered that ratio analysis (which is a common analytical procedure), has had limited ability in the detection of fraud. Most fraudulent reports contain legal jargon which is hard to understand (low readability score). A pilot study was conducted to examine the association between text readability and fraud, and it recommends that the application of the Coh Metrix 2.0 version-Flesch Reading Ease Score formula as measurement tool for text complexity or readability for detecting fraud. Thus, the Flesch score will be evaluated in this project as a text readability metric and will represent the label to be predicted.

**Machine learning models**

There are many different regression and classification machine learning models to choose from. Like evaluation and data preprocessing techniques, each model has its own advantages and disadvantages. There are some models that are more common than others. Some of the most important factors to take into consideration when choosing a machine learning model are the following [15][21]:

* Linearity
  + The data points can be graphically represented as a linear line.
  + Machine learning models that work well with this concept are linear regression, logistic regression, and support vector machines.
* Number of features and parameters
  + If the number of the features and parameters is overwhelming, the training time can be impracticable. However, there is a machine learning model that is designed for this situation – support vector machines.
* Training time
  + Without counting accuracy and the data set size, the complexity of the machine learning algorithm has a very big influence on the training time. For example, decision tree might take significantly more time to train compared to naïve bayes.
* Problem type
  + Supervised or unsupervised
  + Every machine learning algorithm has been designed to achieve a goal. It depends on the goal as to what the problem type is – e.g. classification or regression. Thus, there are algorithms that are good at classifying data and there are other machine learning algorithms that are good at prediction continuous data. The most common classification and regression algorithms [22] are decision tree, naïve bayes, k-nearest neighbors, support vector machine, logistic regression, multivariate regression, and linear regression.

When choosing what machine learning models to train for this project, all characteristics above will be considered.

**Summary of background research**

Text readability analysis has been an area of research for a long time now. Throughout the years, many different and difficult to address challenges have been discovered. Such challenges might be topic of text, reader literacy, et cetera. In itself, readability is a very vague area which is why it is very difficult to correctly predict for different texts. Formulas and techniques that have been used have had a significant increase in the prediction scores but are not perfect. The main challenge is that it is very difficult to take the context of sentences as a parameter and formulas have to mainly rely on basic static features such as words length, sentences length, characters length, etc. A popular readability evaluation technique is Automated Readability Index (or in short ARI) [24].

**Chapter III. Design**

This section of the report outlines the design choices for the project that have been abandoned and/or pursued. These design choices will be mainly concerned with the internal design – in other words the implementation details (such as model selection) and partly with the software architecture of the project.

**Methodology**

There have been multiple directions that this project could have taken. They are machine learning, coming up with own static formula that could build on existing ones, or deep learning. After carefully exploring the above options, and attempting to implement basic variations of them, I came with a conclusion that pursuing the machine learning direction is the best choice. The reason for that is because machine learning definitely fit perfectly with the goal of this project – being able to predict readability scores for different annual reports with extremely good accuracy while eliminating the manual practice to do so. The other two methods were not as fitting due to variety of reasons. For example, coming up with a formula that does better than all the other existent ones, was a struggle, even when all the foundation for the others is available. I attempted to tune different parameters, add or remove ones but still got either the same score or worse, never better. After numerous attempts, this method was no pursued further as it was unlikely to be successful. Deep learning has also been hard to fit within the boundaries of the project. One reason for that is because it seemed excessive. Deep learning is considered to be a subset of machine learning and is largely influenced by *artificial neural networks*. These complex data structures comprise of a magnitude of input, output and “hidden” layers. The whole purpose of neural networks is to make it possible for a machine to learn via its own data processing [16]. As powerful neural networks can be and deep learning as a whole, machine learning can carry out the same results outlined in this project and uses less computational power. Deep learning in general is performed or larger amount of training data and takes a lot more time to predict the value [16].

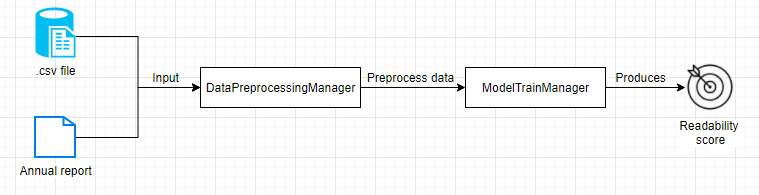
To summarize, machine learning is the method that was adapted and pursued. It was combined with natural language processing. Natural Language Processing (or NLP) attempts to understand the content and the sentiment of a document (in this project’s case – of annual reports) [17]. There have been many natural language processing techniques that have been attempted to synchronize with machine learning. The major ones have been tokenization, stemming. lemmatization, bag of words and removing stop words. Out of these five techniques, only bag of words, stop words removal and tokenization have been used. After evaluating the trained models multiple times, it was clear that lemmatization and stemming did not improve the performance of the model. They did not worsen it either, but they were simply not needed and only slowed the process of training the model. Stemming and lemmatization are very similar. Stemming removes all the suffixes from a word. For example, the word “going” would be converted to “go” or “reported” to “report”. Similarly, lemmatization reduces a word to its base form, called lemma (hence the name of the method). An example of lemmatization would be the word “am” would be converted to “be” [18].

In the following sub-chapter, the code architecture will be discussed. It will shed light on what classes have been created, what internal methods/functions have been written and description for every major one.

**Software architecture**

The programming language that has been used for the project is Python. In the beginning of the next chapter, it is discussed why Python is a great choice when working with machine learning and why it is especially useful for this project.

The architecture of the project is summarized under the following architecture diagram:



*(the diagram was created using draw.io)*

The project architecture consists of multiple classes – each class is responsible for the handling of a single part of the process (for example model training). These classes have been separated in different .py files. These are the following files and classes that are necessary for the full functioning of the code:

* DataPreprocessingManager.py
  + This python file is responsible for managing the data preprocessing stage of the project. It makes sure that the extracted data is in reasonable and easy-to-access format and to be able to feed it into numerical representation so that the machine learning models can understand it and be trained on it.
  + This python file contains two classes:
    - CorpusManager and FileExtractor
    - CorpusManager is responsible for extracting all the numerical information for the annual reports that are stored in the .csv file as well as deciding which columns are relevant and which one should be used for a label (Y).
    - FileExtractor’s purpose is to extract all the textual data from the annual reports and store it in dictionary to be able to easily access it. FileExtractor has been deprecated as the CorpusManager uses the information that is defined in there, but it also utilizes the csv file. However, if for some reason the .csv file will not be required/used and only the textual contents of the reports will be required, this file can be used.
* ModelTrainManager.py
  + This python file is responsible for the training of the model. It handles all the required steps such as splitting the data set into train and test sets [9], training the model, evaluating the prediction, etc.
  + This python file contains only 1 class:
    - Training
    - This class has 3 main functions which are essential for training machine learning models
      * remove\_nans()
        + Removes all the values that are not numbers using the built-in numpy *isnan()* function. All invalid numbers need to be removed or replaced with 0’s so the model can read them and ultimately learn from. For this implementation, the NaN’s (Not a Number) have been replaced with 0s and not removed.
      * feature\_scale()
        + Performs feature scaling on the features and labels of the data set.
      * init\_model(model)
        + Initializes the machine learning model that is passed as a parameter. For example, if the user would want to evaluate how accurate Naïve Bayes is, then he they would call it init\_model(naïve\_bayes\_model).
      * evaluate\_model()
        + Evaluates the trained model by using the predict() function. The output will be R2 score, confusion matrix and the accuracy score.
* Utils.py
  + This file contains the definitions for functions and variables that are used more than once in the other python files.
    - It stores the definitions of ARI table (Automated Readability Index), Flesch score table, and the baseline() function. It also stores other constant variables.
* main.py
  + This python file is responsible for managing all the other classes.

**User interface**

The project does not have GUI interface. Instead, it uses the terminal as UI. Every step of the way, the program notifies the user what it is doing. All major notifications that the user will get are “Loading csv file”, “Extracting features”, “Extracting textual data”, “Initializing model”, “Evaluating model”, and “Accuracy score: “. The reason these notifications exist is because in order for the project to finish predicting the score, it takes relatively long time (considering all the computations the machine will have to carry out). Having these notifications in place, lets the user know at which stage they are and how many more steps there are left.

**Chapter IV. Implementation**

Since the focus of this project is to explore how much more significance machine learning models would have on predicting readability, the programming language that will be used is Python due to the rich selection of machine learning and natural language processing libraries. In this chapter of the report, the decisions and implementations of the design will be reviewed and thoroughly discussed.

**Dependencies management**

Since the usage of machine learning is one of the main building blocks, there are many third-party libraries that have been used. The required libraries were established in the proposal of the project. However, not all listed libraries there were needed, and some were omitted. This was concluded when a given functionality was needed and was either not there or was not required.

The most significant libraries are:

* csv – stands for Comma Separated Values and is used in the project to read pre-formatted annual reports that are in a tabular format (Microsoft Excel).
* NumPy – an open-source numerical library that allows performing mathematical operations on arrays, hence offering a great productivity and flexibility boost.
* matplotlib – enormous libraries that allows producing production-quality graphic with just a couple of lines of code. In the project, it is used to showcase the results of each trained machine learning model and compare it to the others visually.
* pandas – is a data analysis library and provides different tools for managing data (this includes reading and writing) for different formats (for example csv files which is the required format for this project) [29]. pandas offers reshaping of arrays and is highly optimizable. These two characteristics are essential for this project due to the frequent need to reshape arrays (the different sets – X\_train, y\_train, X\_test, y\_test, see below) and the part of the dataset being stored in a csv (comma separated values) file.
* sklearn – this open-source library provides essential tools from training the machine learning models to applying feature scaling and/or normalization. The tool has been used to split the data into train and test sets [9], apply feature scaling on the pre-processed data, and more.

All the above-mentioned libraries are managed using *pip,* a Python tool that allows installing and updating libraries in an easy manner.

**Dataset and feature selection**

There are two parts to the dataset that is utilized by this project – an excel file containing different parameters about each report and the textual data from the reports.

The Excel file which contains all the numerical data extracted from the annual reports using the CFIE-FRSE tool includes the following columns:

* document id, document name, header text, header id, header type, chair CEO flag, header page, actual page, end page, report year, word count, Flesch, fog, page number, front\_1, rear\_1, front\_2, rear\_2, file name flag, number of pages, performance flag, strategy flag, forward looking, negativity, positivity, henry neg 2006, henry neg 2008, henry pos 2008, uncertainty, causal, causal\_martin, performance, user keywords, and booklet flag

Some of these columns are more important than others and contribute much more to assessing the readability than others. There are also columns that have been excluded from training the model – document name, header text, header id, report year, Flesch, Fog and page number. All these columns would not contribute to readability analysis and therefore were not used. However, Flesch and Fog were excluded from the training process not because they would not contribute to the assessment, but because they are the columns the model needs to *predict*.

In addition, the *front* and *rear* columns help us determine which sections of the reports contribute to the analysis of readability. The rationale behind it is that the front sections (front\_2=1) are considered narrative, or in other words – they contain more text than numbers, as opposed to rear sections which are full of numbers. This distinction not only allowed for identification of the useful sections, but also sped up the performance of the data extraction phase because all the rear sections were skipped.

The numerical representation of the annual reports in the csv file proved to be insufficient enough for high result predictions since some of the columns/features are not descriptive and do not contribute to the readability analysis. Therefore, applying natural language processing techniques onto the annual reports directly was necessary. By using the textual data alongside the csv file, there was enough data to feed into the machine learning models for them to produce predictions with higher accuracy. The annual reports were extracted from a directory using the os Python library and using the nltk tool, PorterStemmer, which allowed removing English stop words from the raw annual report texts. Stop words are words that do not have a significant value and therefore there is no need to process them. Examples of such words can be: ‘the’, ‘is’, ‘are’, etc. Applying this technique resulted in much faster processing of the annual reports as they contained far less and much more meaningful words. After the extracted information has been completed, the structure of the variable holding the data is a dictionary for each annual report that holds texts for each of the section titles stored in another dictionary. Example:

* {‘Annual report title’: {‘Section title’: ‘contents of section’}}

The reason for the above structure is because it makes it easy to separate the annual reports from the section titles while still grouping them together accordingly.

**Feature Normalization**

Feature normalization transforms the feature columns in given range in order to make them be in similar scale [19]. The goal of this machine learning transformation is to improve the overall performance and training stability of the model. There are different normalization techniques which are used for different use cases. Some examples of normalization are linear scaling, clipping and log scaling. For example, if the data contains extreme data points, Clipping should be used. On the other hand, if the data points are uniformly distributed, then Linear Scaling should be used. In the scope of this project, the features are represented by a bag of words model and most of the vectors contain data points that are distributed evenly.

It is also important to note that for the training of the models, feature normalization was not used. After testing with different models, with and without normalization, results have summarized that normalization does not improve the performance of the model.

**Splitting the data into training and test sets**

This sub-chapter of the implementation is concerned with creating two separate sets – training set and test set.

The training set will be used to train the machine learning model on existing observations. The test set on the other hand will be used to evaluate the performance of the model on new observations. It is important to distinguish that all the “unseen” observations in the test set will have the same features as the observations in the training set and any possible future data that the model will interact with.

In this project, if feature scaling were to be applied, it should be done after the split of the data to avoid *information leakage* which could occur when accessing information from the test set before the training has completed [9].

In general, there is not one set optimal split percentage. It differs and is based on the objectives of the projects. The points to consider might include:

* Computational cost in training the model
  + If the computational cost is too high, then a large percentage dedicated to the train set might not be optimal
* Computational cost in evaluating the model
* Training set representativeness
  + If the representativeness of the training set is large, then a smaller percentage might be dedicated to the training set and larger to the test set
* Test set representativeness

Most common split percentage include (but are not limited to):

* 80%/20% – Train/Test
* 67%/30% – Train/Test
* 50%/50% - Train/Test

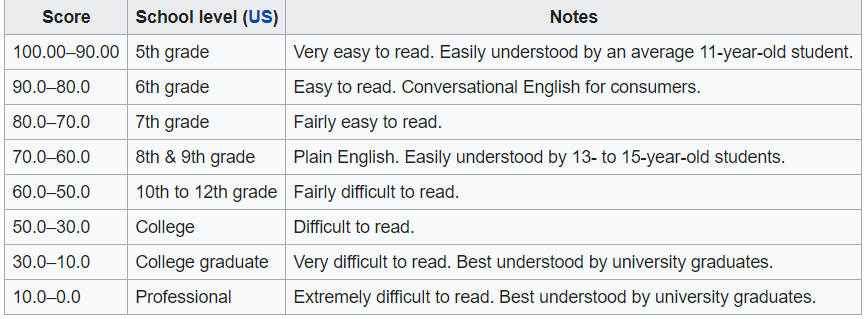
Using the information above, the set split proportion has been selected to be 80%/20% (but it works well with 85%/15%) because the computational cost for training the model is not huge and the training set representativeness is acceptable.

The train and test sets are split into four variables – X\_train, y\_train, and X\_test and y\_test. The following bullet points represent the breakdown of the contents of each of the variables:

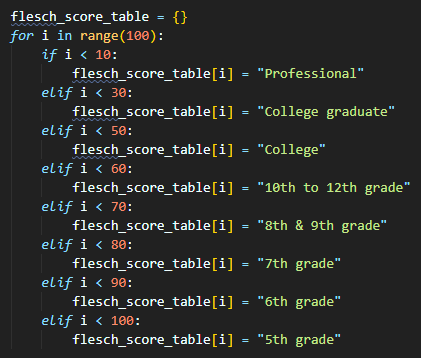
* X\_train
  + The features that the model will be trained on. The machine learning model will learn from these features and try to detect the data from the y variable.
* y\_train
  + The label (readability score) that the model will be trained on. Each feature vector from X\_train will have a corresponding label. When training, the model will learn why the label belongs to the given X\_train row.
* X\_test
  + Subset of the data set that stores all the features that the model will be tested on (previously unknown).
* y\_test
  + Subset of the data set that stores all the labels to all rows in X\_test.

X\_train and y\_train will be used to train the model and X\_test and y\_test will be used to test the model so we can evaluate how effective the machine learning model is in predicting the readability scores.

**Evaluation of trained models**

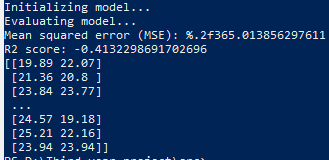
When evaluating machine learning models, there are many different metrics that can be used. Some of them are more suitable than others in different situations. Other metrics contain more detailed information about the predictions than others which many times is required in order to assess the performance of the machine learning model. Some of the more popular classification evaluation metrics are confusion matrix, accuracy score, precision, recall, F1 score (precision and recall combined). However, this project will not use any predetermined evaluation metrics. The reason for that is because Flesch score does not need to be predicted 100 percent accurately, but instead it needs to be in the same range. The table below shows a summary of all ranges (from 0.0 to 100.0): *source: Wikipedia (https://en.wikipedia.org/wiki/Flesch%E2%80%93Kincaid\_readability\_tests)*

For example, if the test value is 23.94 and the predicted value of the model is 20 or 25, it will still be considered an accurate prediction and would contribute positively to the overall evaluation score. Both the predicted and the test values are in the same range (10-30 in the table) which means that both values correspond to the same readability level – college graduate.

The following screenshot shows how the table above has been constructed in Python: 

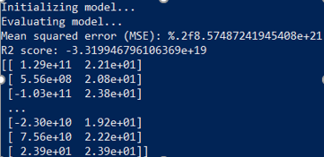
Flesch\_score\_table stores all the score ranges going from 0 to 100. It is achieved using dictionary because of the convinience of having the scores correspond to different levels (profeissional, college graduate, etc.) and easy access.

The screenshot below showcases an output from the project which illustrates the predicted value on the left-hand side and the actual value on the right-hand side, e.g. [19.89 22.07] where the 19.89 is the predicted value and 22.07 is the actual value:



All 6 visible rows’ (pointing to the 2D array) predicted values are accurate as they are in the same range. The model that outputs this is Decision tree regressor. It is important to note the Mean squared error and R2 score showed in the screenshot. As discussed before, they will not be used to evaluate the performance, because they show good results only and only if the predicted and the actual test value are complete match (not if they are both in the same range). Hence why there needed to be adopted a custom evaluation score. To evaluate the accuracy score of the tested model, each predicted, and test value were compared. However, as stated above, it is not compared if they are exactly equal, but whether they are in the same range in the Flesch table shown above. If both values are in the same range (e.g. if both values are in the 10-30 range, that means both of them correspond to College graduate), then 100 is added to an array. However, if they are not in the same range, that means the model predicted the value 100% wrong, therefore adding a 0 to the list. After all values have been added (each value being either 100 or 0), the overall accuracy score is set to the average value of the list above. For example, if there are 4 values to be predicted, and 3 of them were in the same range and 1 is not, the list would be [100,100,0,100]. Then to get the average, the formula is the sum of the array divided by its number of elements, so: 300/4 which is 75. This means that the model predicted the given values with 75% success. Of course, in the actual dataset that this project is tested on, it will have many more elements in the array, thus the score will be more realistic.

Label normalization has also been a tested concept, but it can be useful only in handful of scenarios. It has been tested in this project and the performance is somewhat the same. However, it takes time to perform label normalization and since it does not improve the performance, it has not been used. Not only that, but the labels (the readability score) has been altered and the range is not as easily distinguishable. The following screenshot shows label normalization output:

*Linear Regression*

It is not clear what the range is now because the values have shifted.

By carefully choosing evaluation methods, it is ensured that overfitting and other related problems are not omitted and are carefully addressed by reconfiguring the machine learning model.

**Results and discussion**

This section will discuss the performance of all machine learning models that have been tested. It also puts each of them in a category of how “fast” the model takes to output a score. The 3 categories are “Very slow”, “Manageable”, “Fast”.

The machine learning models have not been trained using a GPU but training models using a modern GPU could speed up the process tremendously.

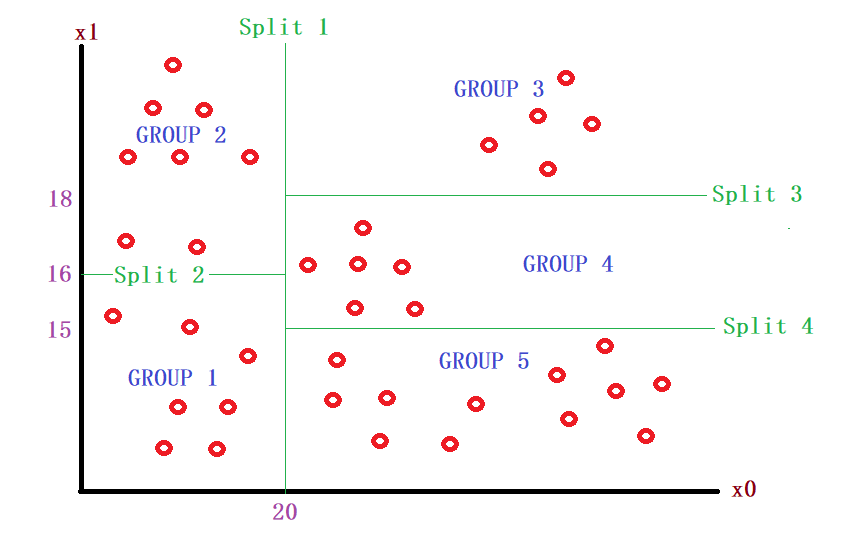
Each machine learning model has different algorithm to try and predict given values. Therefore, each model will have a different accuracy. The purpose of this sub-section is to simply point out the accuracy scores that have been produced by every tested machine learning model. Note that there will not be a huge gap between the accuracy score since most of the tested machine learning models work well with continuous data and are the most popular in the field. This section will also mention how long each model/algorithm takes from start to finish. However, since each machine has its own configuration, it might run faster on other computers than on mine, and vice versa. Thus, the exact time will not be discussed, but rather it will be discussed how fast it takes relative to the other algorithms.

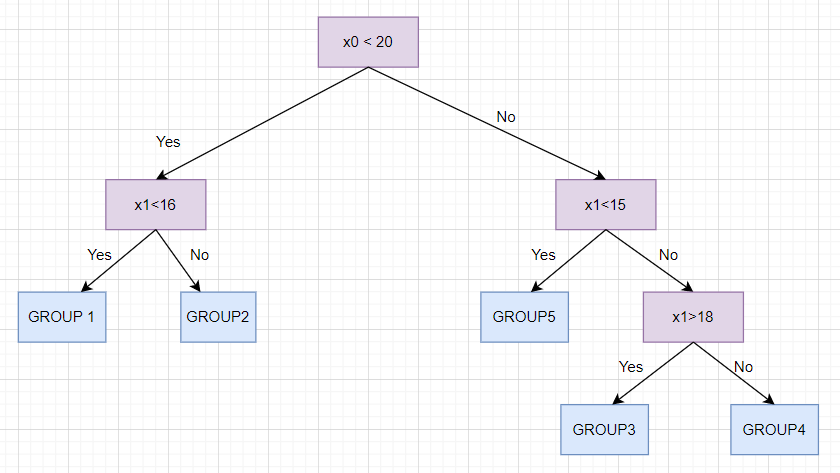
The time scale will have 3 possible values:

1. Very slow
   1. If a model is in this category, the predicted accuracy score might be higher due to the increased number of computation but if time is a concern, then should be avoided.
2. Manageable
   1. A model in this category runs in a comfortable amount of time. The accuracy score is likely to be promising and if time is a concern, the model should be considered.
3. Fast
   1. A model that is in this category will finish predicting the readability score much faster than models in the previous categories, but it is possible the accuracy score to be slightly worse.

**Decision Tree Regressor**

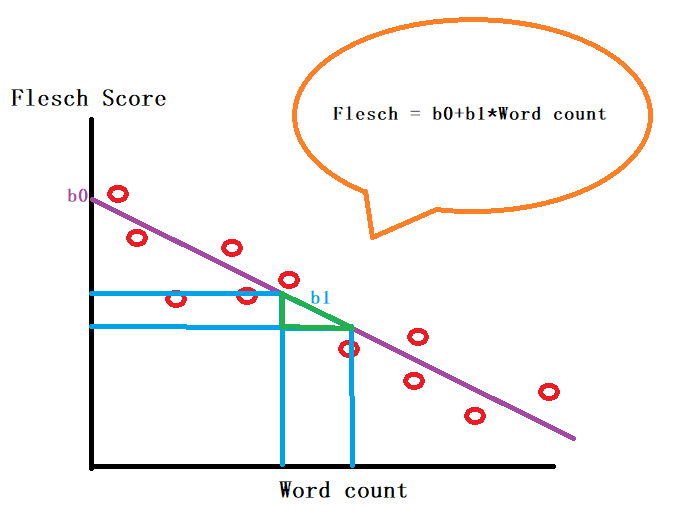
One of the machine learning models that was trained was Decision Tree Regressor. Decision tree in general is a decision-making tool that is in the form of a tree where each parent represents a condition/decision and their children representing the possible results/answers [23][24]. Decision tree regression works with continuous data. On the time scale, it has been placed at number 2, Manageable**. The produced accuracy is ~88%.** It has also been tested with modifying the input parameters. One modification was not removing the stop words and not filtering which annual reports to be scanned based on the csv columns (see sub-chapter Feature Extraction and Feature Selection). The effect of this alteration on the prediction score was not significant. From ~88% accuracy, it fell to ~82%. While 6% accuracy may seem a lot, it improved the training time substantially (thought it still stays in the Manageable speed category). To be able to represent a decision tree using numerical data (the Flesch scores), the entry points of the data set are grouped into segments/splits like the following:

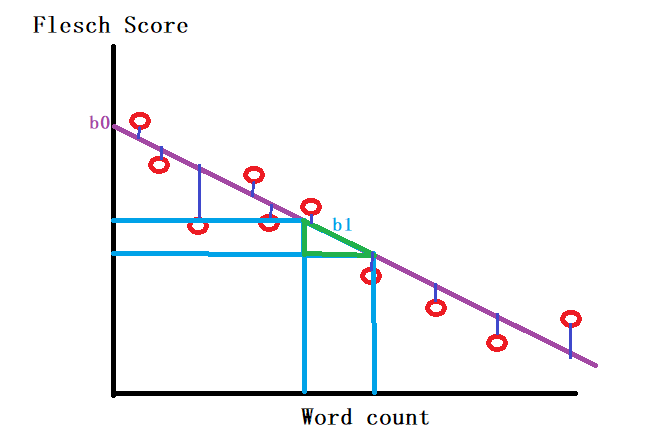
*Decision Tree Illustration*

The above screenshot does not reflect that actual dataset that was used for this project but is used to demonstrate how to algorithm manages the data points. The two numbers ‘15’, ‘16’, ‘18’, and ’20 represent the coordinates on the axis where the splits have occurred. This is important information because they will be the building points for building the decision tree itself (see the decision tree diagram below). x0 and x1 represent the features. For example, in text readability context, they could represent the word count and average sentence length (or any other parameters). For this project, there will be many more features that are taken into consideration but are not drawn because it would be very difficult to visualize. Additionally, it is important to point out that the splits are entirely curcumstantial – meaning they do not have to be the same as in the screenshot above. In fact, ideally, there are fewer splits (depending on how large the dataset is) [27]. The rationale behind having fewer splits between the data points is to avoid *overfitting*. To determine where the splits should be, the decision tree algorithm looks at the *information entropy* or uses *gini impurity index*. By default, the scikit-learn model (which is the one this project adapts) uses the gini impurity index by default [28]. In simple terms, information entropy and gini impurity answer the question “if a split were to be perfmored between these two clusters/groups, would it add any information value”. Using all this information, this is how the above diagram would be represented in the form of a tree: 

X0 and x1 are used for every decision needed to be made until the leafs (or in this case, the groups) have been reached. Decision trees work nicely for this problem because, despite having many more features (not just two (x0 and x1)), it scales nicely and the prediction score (see above) confirms that.

**Simple Linear Regression**

Another model that was trained was Linear Regression. Linear regression uses the simple formula y = b0 + b1\*x1 which represents the line. b0 is a constant which represents the starting coordinate of the line, b1 is the slope of the line and xi is a feature [26], e.g. words count. The left part of the equation, *y*, is the dependent variable that is attempted to be predicted, in this project’s scope that would be the Flesch reading score extracted from the csv file. Linear regression attempts to understand by what margin exactly, does the feature x1 affect the predicted value y [26]. For example, let y be the Flesch score the machine learning model tries to predict and x1 be word count. Simple linear regression would be represented like this:  *Simple linear regression*

From the diagram above, the purple line represent the linear line that goes through all the data points (the red circles) (this is not the best-fitting line, this is just drawn for illustration purposes) and b0 represents where the line starts [30]. The reason the line points downards is because as the word count increases, so should the Flesch score. Of course, as discussed in this project, simply predicting flesch score by the word count is not feasible. It will be shown how linear regression adjusts for higher magnitude of features (xi). B1 reprsents the slope, or the rate at which the linear line increases or decreases. For example, b1 can be measured between two points (the green lines represent the slope between two points). The vertical green line represents the change in Flesch score and the horizontal green line represents the change in word count. So in short, it shows by how much the Flesch score decreased when word count increased. But how exactly can this linear line be utilized? It shows where exactly each data point should be. The following illustration shows where the points should have been and where they actually are: 

The dark blue lines connecting the linear line and the data points show where the red circles should have been according to the model. To get the best fitting line, the linear regression model will use the following formula:

Best fitting line = *SUM(actual y – predicted y)^2 [30]*

As mentioned above, this type of linear regression does not work for this project because to predict the Flesch score, it has been established that there will be needed a lot of features, not just one. The above type of linear regression is called Simple due to its ability to only consider a single feature.

**Multiple linear regression**

The type of linear regression that can handle multiple features is called Multiple Linear Regression. Just like simple LR, multiple LR tries to determine the best fitting line by taking into consideration all the features and their coefficients (b1). The formula for the linear line for multiple LR is [31]:

*Y = b0 + b1\*x1 + b2\*x2 + b3\*x3 + bi\*xi*

where each xi represents a feature.

The multiple linear regression model will try to generate linear lines across the data points and then will get the best-fitting line out of all by using the best fitting line formula (see above). Because of all the computation involved (largely related to creating linear lines and computing the best-fitting line), this model is not one of the fastest but is also not extremely slow.

Therefore, it falls under the Manageable category in terms of speed. After testing the model, **the produced accuracy is ~95%** which beats Decision Tree Regressor in terms of performance.

**Random Forest Regression**

Another machine learning model that was trained was Random Forest Regression. This machine learning model is extremely popular for predicting continuous data (such as Flesch reading scores). The reason why it is widely implemented is due to its high accuracy and algorithm simplicity [26].

This model’s name contains the word “forest” because it is made up of many decision trees (see above) [28]. This type of regression’s purpose is to reduce the risk of overfitting (because of the wider variety of trees) and accuracy [26][28]. The process of performing random forest regression involves selecting *K* data points from the training set, construct the decision tree with the K data points, then the model chooses a number of trees that it thinks should be created and then repeats the previous two stages [26][28].

Put simply, this model is a set of Decision trees (explained above) that are performed on random data points. Interesting enough, this model performed just as good as a single Decision tree. **The accuracy of this model is ~88%.** However, despite getting the same accuracy score and taking more time to train, random forest regression should be the preferred method due to the fact that it reduces the risk of overfitting. It runs slightly slower than Decision tree regressor, but the time it takes to train and evaluate the model is not significantly slow. In the category defined above, it is placed in the **Manageable** category.

**Baseline score**

Even though the score from the trained machine learning models that is shown and described above, we cannot determine exactly how accurate it is. The rationale behind is that it is possible to achieve the same, or at least close enough, prediction score just using simpler methods without machine learning. If that were the case, there would be no point in using machine learning as it requires more resources, for example time. To understand how accurate the above prediction scores are, a baseline score needs to be tested.

Baseline score is the score that is produced by using simplistic features. The purpose of having baseline score is to know the *base* score for predicting readability. That is, when more features are being added and other tools/techniques are being used, its accuracy score can be compared to the baseline score. For example, if the accuracy of the baseline score is 60% and the accuracy of a modified formula is 70%, it can be seen that by adding these additional features, there is a 10% increase in the accuracy score. Consequently, if the new method of training the model scored 50%, there is a 10% decrease which means it underperforms the baseline and should not be used in most cases.

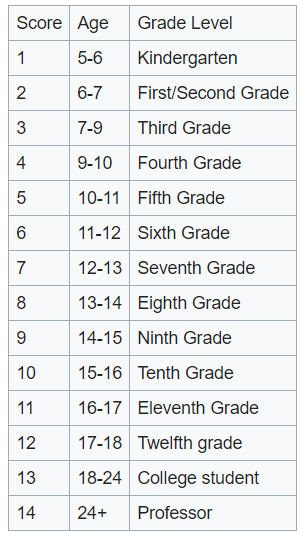
As discussed in this report (refer to the Implementation and Background chapters), the main method for predicting and evaluating readability uses the Flesch score of each report which is provided in the csv file. To predict the Flesch score, the current method is using textual data of the annual reports grouped into a CountVectorizer (refer to Implementation) and removing all the stop words for more accurate prediction.

The chosen baseline method represents Automated Readability Index (ARI) which is another readability metric table, similar to the Flesch score table. Just like Flesch, it produces an approximation in the form of US grade level that is required to comprehend the textual contents [27]. However, it is important to note that this baseline score does not utilize machine learning, instead it is a simple formula.

The formula used for calculating the ARI is [27]:

*4.71 x (characters / words) + 0.5 x (words / sentences) – 21.43*

This equation scales the result to be between 1 and 14 but some exception might occur that can go beyond 14 (in which case they would still be considered as the latest grade level in the table, in this case - professor). The table rows and columns are showed in the following screenshot:

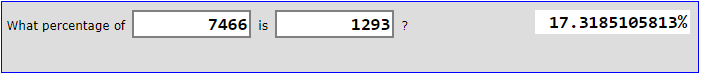


*source: Wikipedia (https://en.wikipedia.org/wiki/Automated\_readability\_index)*

The reason for choosing ARI as a baseline is because it uses only the most basic features of the text (the number characters, words and sentences) and does not take into account the context of the sentences nor does it use any machine learning. However, it is important to point out that ARI can be used to evaluate other languages as well, unlike Flesch score which can be used to evaluated English only. So, one thing that the baseline score is better is its scalability. However, for the purposes of this project, language selection is not a concern and language-scalability is not essential.

There was a problem when I tried to compare the baseline score to the main implementation’s score. The issue was that one of the main method uses machine learning, and as discussed above, the data has been split amongst 4 sets – train set for features, train set for labels, test set for features and test set for labels. This means that the test set is randomized and is only 15% of the entire dataset. Whereas the baseline method, calculates the ARI score for *every single* report and simply comparing 15% of the data to 100% of the data will not give accurate comparison. However, to evaluate the base method’s performance, it is not necessary to compare it with the same values that the main method was tested on. What matters is that they are tested on the same dataset entries. For perspective, if there are 100 reports in the dataset, then the main implementation will be evaluated after comparing it to 15 random reports. Whereas the baseline score will be evaluated after comparing every ARI grade level to every Flesch grade level. For example, if the produced ARI score is 13 (which corresponds to College student grade level) and the Flesch score is between 10 and 30 (which also corresponds to College student grave level), then there is a match and the ARI method correctly predicted the grade level of the text.

After calculating the baseline score for each report by analyzing the excel spreadsheet, the score is:



*Source: http://www.alcula.com/calculators/finance/percentage-calculator/#gsc.tab=0*

Only 1293 annual reports’ grade levels were predicted correctly out of 7466. In percentage, it is 17% which is way less than what the focused implementation of this project predicts.

To conclude, using simplistic formulas that do not utilize machine learning, such as Automated Readability Score, and only use basic features such as word count, sentence count, etc., perform significantly worse than feeding this data into machine learning models.

**Chapter V. Conclusion**

**Review of Aims**

The central goal of this project was to create a readability tool that utilizes the modern advances of Machine Learning. The target texts to be analyzed by this project are Annual Reports that are produced by companies. Unlike many of the available tools that are concerned with measuring readability in the financial domain, this tool combines the static approaches that have been implemented and thoroughly tested, with state-of-the-art modern machine learning models. The rationale behind this combination is that these modern algorithms can take variety of different parameters, apply complex computations, and automate the entire process. As a result, less time is spent on carrying out the calculations by the user, it is more scalable, and very flexible. However, with all these benefits offered by this tool, there is a downside. More specifically, it is the time that is required to predict the readability. It can vary from one hour to more than 10 hours. It all depends on the number of parameters that are being processed, the type of machine learning algorithm, and most importantly – the size of the dataset the model to be trained and tested on. The long time for producing results is a trade-off for automating the process, producing high-accurate results and the ability to carry out highly complex computations with ease. Moreover, even though the required time is long, it can be produced once per model and then store the result in a spreadsheet. That way, the user can track which models performed better and will not need to run the tool every single time.

**Learning Outcomes**

I learned a wide variety of skills thanks to this project. Firstly, it exposed me to the machine learning field and natural language processing and having worked on this project for a long time, it made me realize how much I enjoy working in these ever-developing fields. After completing the project, I know what a machine learning model is, how to train it, how to evaluate it and finally, how to choose between the array of models that are available. I also learned how to combine natural language processing with machine learning and yield very accurate results. Due to the freedom granted by my supervisor, I had the opportunity to explore many ways of achieving solutions, which in turn exposed me to gaining knowledge with multiple tools. I was introduced to a new way of testing the integrity of the methods used by adapting a baseline score which is just a similar method but with more simplistic design. Apart from the computer science related learnings, I also learned a lot about computational linguistics. This project allowed me to understand how to analyze the English language using linguistics techniques.

Apart from my academic learning, I have also learned many transferrable skills. I can now claim that I have experience in technical writing, significantly improved my communication skills, both verbal and written, and gained vast knowledge in Python and its libraries. Since this project has had a long time period for completion, I had to carefully plan every week and every month and keep up to date with the assigned milestones. This has made me more aware of how important organization is and why it should be a must for every project.

**Suggested Revisions**

While all the set milestones have been completed and the main aim has been achieved, there are some parts that could be added or extended if there was more time to further develop.

An example of such improvement would be the user interface. It would be ideal if a GUI was developed where the user has the freedom of choosing what model to train, to select the parameters that are offered by the project and still get continuous feedback while the model is being trained. However, due to the challenges in implementing everything else, this optional goal has not been met.

Another aspect where the project could improve is by further modifying or coming up with alternative baseline methods. While Automated Readability index score is a viable option and could be used as a comparator, there could be better options. For example, there could be an option that requires even simpler parameters than Automated Readability Index and act as a better *base* case. However, overall ARI proved to be a fantastic baseline and showcased how the Flesch score and the other factors and parameters that have been tested, performed.

Similarly, more machine learning and deep learning models could have been experimented with. While many models have been tested, there have been others who have not. There could be models that have the potential of performing better than the ones mentioned in the Implementation chapter. However, every model testing requires a large amount of time due to all the calculations and testing all the available models could not be achieved in the allotted time frame.

To conclude, the areas that could benefit from more work are the UI, and further baseline and machine learning models experimentation.

**Future Work**

While all the main objectives have been met, that is producing readability using machine learning models and common parameters used in previous research (see Background), there is a potential for many more optional goals and improvements.

In the future, the project can be expanded to analyze readability of different financial texts. Currently, the focus of readability evaluation was on Annual Reports produced by companies. Given how the current project is structured, it would not be too difficult to be able to produce readability scores from other forms of financial disclosures. What would need to be modified is the parameters that are fed into the machine learning model. The current parameters are too specific for Annual Reports. So, coming up with more generalized parameters that can be used as an input for a machine learning model, would allow predicting readability score for a wider variety of literature.

**Closing Paragraph**

TODO

**References**

1. Automated readability prediction - <https://medium.com/voice-tech-podcast/automated-readability-prediction-71d40b677563> - Accessed 3rd February 2021
2. What is readability and why does it matter - <https://devstephen.medium.com/what-is-readability-and-why-does-it-matter-7773d018e4b9> - Accessed 19th November, 2020
3. Loughran, T. and McDonald, B., 2014. Measuring readability in financial disclosures. *the Journal of Finance*, *69*(4), pp.1643-1671.Van Der Walt, S., Colbert, S.C. and Varoquaux, G., 2011. The NumPy array: a structure for efficient numerical computation. *Computing in science & engineering*, *13*(2), pp.22-30, Accessed 17th February 2021
4. 1. Kincaid, J.P., Fishburne, R.P., Rogers, R.L., & Chissom, B.S. (1975). Derivation of new readability formulas (automated readability index, fog count, and flesch reading ease formula) for Navy enlisted personnel. Research Branch Report 8–75. Chief of Naval Technical Training: Naval Air Station Memphis – Accessed 16th February 2021
5. Klare, George R. “Assessing Readability.” *Reading Research Quarterly*, vol. 10, no. 1, 1974, pp. 62–102. *JSTOR*, www.jstor.org/stable/747086. Accessed 17th February 2021
6. Bonsall IV, S.B., Leone, A.J., Miller, B.P. and Rennekamp, K., 2017. A plain English measure of financial reporting readability. *Journal of Accounting and Economics*, *63*(2-3), pp.329-357., Accessed 18th February 2021
7. Scott A. Crossley, Stephen Skalicky, Mihai Dascalu, Danielle S. McNamara & Kristopher Kyle (2017) Predicting Text Comprehension, Processing, and Familiarity in Adult Readers: New Approaches to Readability Formulas, Discourse Processes, 54:5-6, 340-359, DOI: [10.1080/0163853X.2017.1296264](https://doi.org/10.1080/0163853X.2017.1296264), Accessed 15th February 2021
8. Train-test split for evaluating machine learning algorithms - <https://machinelearningmastery.com/train-test-split-for-evaluating-machine-learning-algorithms/> - Accessed 24th November, 2020
9. How does Grammarly work? - <https://support.grammarly.com/hc/en-us/articles/115000090871-How-does-Grammarly-work-> - Accessed 27th November, 2020
10. What is readability - <https://www.grammarly.com/blog/readability/>, Accessed 1st of February 2021
11. Automatic Readability Assessment - <https://academicworks.cuny.edu/cgi/viewcontent.cgi?article=2964&context=gc_etds> – Accessed 15th February 2021
12. Alloghani M., Al-Jumeily D., Mustafina J., Hussain A., Aljaaf A.J. (2020) A Systematic Review on Supervised and Unsupervised Machine Learning Algorithms for Data Science. In: Berry M., Mohamed A., Yap B. (eds) Supervised and Unsupervised Learning for Data Science. Unsupervised and Semi-Supervised Learning. Springer, Cham. <https://doi.org/10.1007/978-3-030-22475-2_1>, Accessed 19th February 2021
13. I. W. Othman et al., "Text readability and fraud detection," 2012 IEEE Symposium on Business, Engineering and Industrial Applications, Bandung, Indonesia, 2012, pp. 296-301, doi: 10.1109/ISBEIA.2012.6422890.January, 2021, Accessed 21st February 2021
14. How to evaluate regression models? - <https://medium.com/acing-ai/how-to-evaluate-regression-models-d183b4f5853d> - Accessed 20th February 2021
15. Deep learning vs machine learning - <https://docs.microsoft.com/en-us/azure/machine-learning/concept-deep-learning-vs-machine-learning>, Accessed 21st February 2021
16. Chowdhury, G.G., 2003. Natural language processing. *Annual review of information science and technology*, *37*(1), pp.51-89. – Accessed 21st February 2021
17. Balakrishnan, V. and Lloyd-Yemoh, E., 2014. Stemming and lemmatization: A comparison of retrieval performances. – Accessed 21st February 2021
18. Normalization - <https://developers.google.com/machine-learning/data-prep/transform/normalization> - Accessed 21st February 2021
19. Choosing a suitable Machine Learning algorithm -https://www.geeksforgeeks.org/choosing-a-suitable-machine-learning-algorithm/ - Accessed 17th February 2021
20. The most common machine learning classification algorithms for data science and their code - <https://medium.com/swlh/13-machine-learning-classification-algorithms-for-data-science-and-their-code-e185e8fca507> - Accessed 17th February 2021
21. Min Xu, Pakorn Watanachaturaporn, Pramod K. Varshney, Manoj K. Arora, Decision tree regression for soft classification of remote sensing data, Remote Sensing of Environment, Volume 97, Issue 3, 2005, Pages 322-336, ISSN 0034-4257, Accessed 29th February 2021
22. Myles, A.J., Feudale, R.N., Liu, Y., Woody, N.A. and Brown, S.D., 2004. An introduction to decision tree modeling. *Journal of Chemometrics: A Journal of the Chemometrics Society*, *18*(6), pp.275-285. Accessed 29th of February 2021
23. Danielle S. McNamara, Max M. Louwerse, Philip M. McCarthy & Arthur C. Graesser (2010) Coh-Metrix: Capturing Linguistic Features of Cohesion, Discourse Processes, 47:4, 292-330, DOI: [10.1080/01638530902959943](https://doi.org/10.1080/01638530902959943), Accessed 15th February 2021
24. Groß, J., 2012. *Linear regression* (Vol. 175). Springer Science & Business Media., Accessed 17th February 2021
25. Senter, R.J. and Smith, E.A., 1967. *Automated readability index*. CINCINNATI UNIV OH., Accessed 19th February 2021
26. Random forest regression - <https://www.keboola.com/blog/random-forest-regression> - Accessed 27th February 2021
27. Segal, M.R., 2004. Machine learning benchmarks and random forest regression. – Accessed 28th February 2021
28. Decision trees and random forest explained - <https://towardsdatascience.com/decision-tree-and-random-forest-explained-8d20ddabc9dd> - Accessed 28th February 2021
29. About pandas - <https://pandas.pydata.org/about/index.html> - Accessed 1st March 2021
30. Line of best fit in linear regression - <https://towardsdatascience.com/line-of-best-fit-in-linear-regression-13658266fbc8> - Accessed 1st March 2021
31. Tranmer, M. and Elliot, M., 2008. Multiple linear regression. *The Cathie Marsh Centre for Census and Survey Research (CCSR)*, *5*(5), pp.1-5. – Accessed 1st March 2021