
A study of Machine Learning Algorithms for Automated Essay Grading

By

MD. SADDAM HOSAIN



Department of Computer Science and Engineering
MANARAT INTERNATIONAL UNIVERSITY

A thesis titled "A study of Machine Learning Algorithms for Automated Essay Grading " submitted to the Manarat International University in accordance with the requirements of the degree of BACHELOR OF SCIENCE IN COMPUTER SCIENCE AND ENGINEERING

DECEMBER 2017

ABSTRACT

Automated essay grading is the task of predicting the score of an essay so that the score might seem like it has come from a human reader. Aim of this paper is to examine the possibility of developing such automated essay grader. To this end different natural language processing techniques and machine learning algorithms have been studied. As essay data set 1800 essays from Hewlett foundation available in Kaggle, have been used. At first different natural language processing techniques and tools have been used to process those essays. Then noticeable tf-idf features are extracted to train different supervised machine learning algorithms to build an artificial system that could score further user given essays.

DEDICATION AND ACKNOWLEDGEMENT

At first I would like to thank my thesis supervisor Sohaib Abdullah, Assistant Professor of Department of Computer Science and Engineering for allowing me to work on this thesis under his supervision and for his inspiration, ideas and suggestions to improve this work. In many stages of my work I have found the support and help from my supervisor.

AUTHOR'S DECLARATION

It is hereby declared that this thesis or any part of it has not been submitted else where for the award of any degree or diploma.

SIGNED: DATE:

TABLE OF CONTENTS

	Page
List of Tables	vii
List of Figures	viii
1 Introduction	1
1.1 Motivation	2
1.2 Limitation	2
2 Problem definition	3
2.1 Usefulness	3
2.2 Goal	3
3 Literature Review	5
4 Machine Learning	7
4.1 Supervised Learning	8
4.2 Unsupervised Learning	8
4.3 Semi-Supervised Learning	8
4.4 Python	8
4.5 scikit-learn	9
4.6 Natural Language Processing	9
4.7 Natural Language Toolkit (NLTK) & NLP	9
4.8 Numpy	10
4.9 pandas	10
5 Machine Learning Algorithms	11
5.1 Supervised Machine Learning	11
5.2 Unsupervised Machine Learning	11
5.3 Support Vector Machine	12
5.4 SGDRegressor	12
5.5 Random Forest Algorithm	13

5.6	K Nearest Neighbors	14
5.7	Naive Bayes classifier	15
6	Methodology	16
6.1	Thesis Workflow	16
6.2	Data collection	17
6.2.1	Code	19
6.3	Text Preprocessing	19
6.3.1	Word tokenization	19
6.3.2	Removing Stop Words	19
6.3.3	Stemming	19
6.3.4	Code	20
6.4	Producing Tf-Idf Matrix	22
6.4.1	Code	22
6.5	Singular Value Decomposition (SVD)	23
6.5.1	Code for U,S,V	24
7	Regression and Classification model	25
7.1	Regression	25
7.1.1	SGDRegressor	25
7.1.2	RandomForestRegressor	27
7.2	Classification	29
7.2.1	BernoulliNB	29
7.2.2	KNeighborsClassifier	30
7.2.3	RandomForestClassifier	32
7.2.4	Support Vector Machine	33
8	Results and Comparison	35
8.1	R Squared Score	35
8.2	Variance	36
8.3	Accuracy score	36
8.4	Precision score	36
8.5	recall score	36
8.6	Mean squared error	37
9	Conclusion and Future work	39
9.1	Conclusion	39
9.2	Future work	39
10	Glossary	40

11 Reference	41
---------------------	-----------

LIST OF TABLES

TABLE	Page
7.1 Results of SGDRegressor	26
7.2 Results of RandomForestRegressor	27
8.1 R Squared Score	35
8.2 Variance Score	36
8.3 Accuracy Score	36
8.4 Precision score	36
8.5 recall_score	37
8.6 Classification Result	37
8.7 Mean Squared error	38

LIST OF FIGURES

FIGURE	Page
6.1 Thesis workflow	17
6.2 Data Set	18
6.3 Tokenized word.	20
6.4 The Formation of singular value decomposition	23
7.1 Graph of Actual and predicted value of SGDregression model	26
7.2 Actual and predicted value of SGDregression model	27
7.3 Graph of Actual and predicted value of RandomForestRegressor model	28
7.4 Actual and predicted value of RandomForestRegressor model	28
7.5 Graph of Actual and predicted value of BernoulliNB model	29
7.6 Actual and predicted value of BernoulliNB model	30
7.7 Graph of Actual and predicted value of KNeighborsClassifier model	31
7.8 Actual and predicted value of KNeighborsClassifier model	31
7.9 Graph of Actual and predicted value of RandomForestClassifier model	32
7.10 Actual and predicted value of RandomForestClassifier model	33
7.11 Graph of Actual and predicted value of svm model	34
7.12 Actual and predicted value of svm model	34

INTRODUCTION

Automated essay scoring (AES) has been in the research area of computer science since the early 1966. Predicting the score of an essay so that the score might seem like it has come from a human reader is a bit daunting task because there are numerous quantified features that have to be extracted from the essay as well as many unquantifiable properties like the perceptions of the writer while writing the essay and his thoughts that he is trying to inscribe on the paper. Therefore, the behavior of the essay inherently noisy, non-stationary and deterministically chaotic. The quantifiable data that we have extracted from the essay is relatively easy for the computer to process rather than processing the ideas or thoughts of the writer in the essay, which may or may not affect the scoring of an essay by a computer. Analyzing natural language, or free-form text used in everyday human-to-human communications, is a vast and complex problem for computers regardless of the medium chosen, be it verbal communications, writing, or reading. Ambiguities in language and the lack of one “correct” solution to any given communication task make grading, evaluating or scoring a challenging undertaking. In general, this is a perfect domain for the application of machine learning techniques with large feature spaces, and huge amounts of data containing interesting patterns. Automated essay scoring (AES) is the use of specialized computer programs to assign grades to essays written in an educational setting. It is a method of educational assessment and an application of natural language processing. Its objective is to classify a large set of textual entities into a small number of discrete categories, corresponding to the possible grades—for example, and the numbers 1 to 6.

My research focuses on the part where to make an artificial environment learn and predict scores from the essays that it will receive. Using a training dataset, the artificial environment can identify the patterns and tries to predict the next possible output.

1.1 Motivation

Many large-scale testing programs around the world include at least one essay writing item. Examples include the GMAT® test administered by the Graduate Management Admission Council®, the GRE ® revised General Test administered by ETS, as well as the Pearson® Test of English (PTE). The written responses to such items are far more complex than responses to multiple-choice items, and are traditionally scored by human judges. Human raters typically gauge an essay's quality aided by a scoring rubric that identifies the characteristics an essay must have to merit a certain score level. Automated scoring has the potential to provide solutions to some of the obvious shortcomings in human essay scoring. My research tends to build a system for computer-based scoring involving relevant aggregation of quantifiable text features in order to evaluate the quality of an essay.

1.2 Limitation

Finding the right sets of data for any research is really a challenge and for my research it is basically the same. The problem is that in order to make my environment learn and extract features from the essays I need to have a large collection of essays.

PROBLEM DEFINITION

Scoring students' writing is one of the most expensive and time consuming activity for educational assessment. Especially in language tests writing is considered as an essential part to assess students' language skill. In such tests students are usually require to write different essays and raters score their essays based on some given criteria. As such language tests become more and more popular and number of the testees become larger scoring essays become a huge labor intensive work.

2.1 Usefulness

Moreover human scoring methods will lead to an inaccuracy of grading . First of all scoring task is much more time consuming and expensive. It requires teacher recruiting, training, calibration and monitoring total system. Even human raters often subject to perception difference error, drift error and inconsistency error. Therefore a more accurate and fastest automated essay scoring method has been desired for a long time.

2.2 Goal

Automated essay scoring has been a very fascinating research area among the computer scientists since early nineties. As much as it is challenging it is also rewarding with its significant knowledge gain. Simply by making a computer understand to one of the human languages along with the complexities and dimensions of that language and on top of that rating a person's essay while suggesting some improvements is absolutely very demanding brainstorming thing as well as a solution for many real world scenarios. Scenarios where thousands of students around the world are writing some sort of essays for various reasons need to be checked by hundreds of qualified

people is really a time consuming and laborious task. This research has been a platform for us to learn about NLP, how it works and learning about the methods of machine learning along with many existing algorithms and combine them with NLP to make a stable AI that can help the teachers for grading the essays . I have also been trying to see how accurate the current prediction algorithm works and how this would help me to attain my goal. I have started working with the existing different machine learning algorithms to see which one predicts the better score of an essay with less prone for error. To begin this process I must carefully select a dataset where there are relatively large number of essays on a particular topic . I must find dataset and start extracting the features from those collection of essays before feeding them into the algorithm. As a result, I have turned our attention to python's Natural Language Toolkit (NLTK) for Natural Language Processing (NLP) to extract features like word count, in fact I have extracted tf-idf features from every single essay before running the machine learning algorithm. Therefore, this research is just a tip of iceberg compare to what we think is waiting for us in the future.

LITERATURE REVIEW

First generation automated essay scoring systems have been developed in the mid to late 20th century. Limited language processing methods and algorithms (e.g. word counts, grammar/spell checks) are used to extract and evaluate lexical and syntactic properties of essays. Regression analyses were then typically applied to generate an essay score. As early as 1966, Page developed Project Essay Grader (PEG) which was first developed which uses multiple regression technique to grade essays. On this system grading was done on the basis of writing quality, taking no account of content. Vector-space model was developed by Sparck Jones at 1972 which starts with co-occurrence term-document matrix formed from the essay. TF-IDF is used for weighting the elements of matrix and Cosine correlation is used for scoring. It is less successful at judging overall quality of essay. E-rater is an essay-scoring system developed in 1990 by Educational Testing Service. E-rater uses multiple regressions with NLP to extract writing features of essays. By comparing human and E-Rater grades essays with 87% accuracy. Bayesian Essay Test Scoring System (BETSYS) is based on Multivariate Bernoulli Model (MBM) and the Bernoulli Model (BM). BETSYS is a program that classifies text based on trained material and is being developed by Lawrence M. Rudner. Using BETSYS an accuracy of over 80% of Artificial Intelligence (AI), Natural Language Processing (NLP), and statistical technique. IntelliMetric process assists to examine the essay according to the main characteristics of standard written English.

Bin L. et al. designed an essay grading technique that used text categorization model which incorporates K-Nearest Neighbor (KNN) algorithm. Transforming the essays into the vector space model (VSM), TF-IDF and IG are applied for feature selection from the feature pool of words, phrases and arguments. After training for the KNN algorithm, a precision over 76% Intelligent Essay Assessor (IEA) developed by Hearst et al. based on the Latent Semantic Analysis (LSA)

technique was originally designed for indexing documents and text retrieval. Whittington et al. defined that LSA represents documents and their word content in a large two-dimensional matrix semantic space. Using a matrix decomposition technique known as Singular Value Decomposition (SVD), new relationships between words and documents are built and existing relationship are improved to more accurately represent their true significance. The correlation from 0.59 to 0.89 has been achieved between the IEA and human raters.

MACHINE LEARNING

Instead of writing programs that explain to computers how to perform specific task, in machine learning, programmers write algorithms that explain computers how to learn by themselves to perform tasks. Machine Learning algorithms is a computer program that teaches computers how to program themselves so that every time a human programmer does not have to explicitly describe a computer to perform the task that we want to achieve. The need to make a program that explains to computers how to perform each task is the greatest limitation faced by traditional computer science programming. It has prevented computers from further extending our intelligence to solve more complex tasks. To truly extend our intelligence, we need computers to accomplish tasks that we don't even know how to do. However, machines do not learn in the way humans do, humans have specialized pattern discovering abilities that allow humans to extract patterns but machines have no idea of what so ever a pattern is. Fortunately, there is something called feedback mechanism that in some ways mildly similar. Machines "learn" when they take a series of input data items and, based on some mathematical criteria, they correctly chose an algorithm (a pattern of sorts) to apply to that input so that the output is acceptable to the user. Being accepted or not accepted is important because that feedback information accumulates and feeds into the selection criteria used to select the algorithm to use. It's a closed feedback loop. The current machine learning algorithms that exist are designed to solve a single well defined problem, often better that humans. However, significant work normally needs to be done to get the data into a form that is algorithmically operable. Furthermore, these algorithms do not match people's expectation of human intelligence because they tend to learn on thing very precisely at a time but not more than one thing. Machine Learning has been categorized into 3 categories precisely:

4.1 Supervised Learning

In supervised learning the data has been divided into two subgroups. First group is called the training set and the later called the test set, with the training set a model has been prepared where a machine has to predicts outcome based on that model and corrected when there is a wrong prediction. The training process continues until a desired level of accuracy has been attained. After that, the model that has been prepared has to be tested with the test set to see how it actually works in the real life.

4.2 Unsupervised Learning

Unsupervised learning is where you only have input data (X) and no corresponding output variables.

The goal for unsupervised learning is to model the underlying structure or distribution in the data in order to learn more about the data.

These are called unsupervised learning because unlike supervised learning above there is no correct answers and there is no teacher. Algorithms are left to their own devises to discover and present the interesting structure in the data.

4.3 Semi-Supervised Learning

Input data is a mixture of labelled and unlabeled examples. There is a desired prediction problem but the model must learn the structures to organize the data as well as make predictions.

4.4 Python

Python is a widely used high-level programming language for general-purpose programming, created by Guido van Rossum and first released in 1991. An interpreted language, Python has a design philosophy that emphasizes code readability (notably using whitespace indentation to delimit code blocks rather than curly brackets or keywords), and a syntax that allows programmers to express concepts in fewer lines of code than might be used in languages such as C++ or Java. It provides constructs that enable clear programming on both small and large scales.

Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

Python interpreters are available for many operating systems. CPython, the reference implementation of Python, is open source software and has a community-based development model, as do nearly all of its variant implementations. CPython is managed by the non-profit Python Software Foundation.

4.5 scikit-learn

Scikit-learn (formerly scikits.learn) is a free software machine learning library for the Python programming language. It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, k-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy.

The scikit-learn project started as scikits.learn, a Google Summer of Code project by David Cournapeau. Its name stems from the notion that it is a "SciKit" (SciPy Toolkit), a separately-developed and distributed third-party extension to SciPy.[4] The original codebase was later rewritten by other developers. In 2010 Fabian Pedregosa, Gael Varoquaux, Alexandre Gramfort and Vincent Michel, all from INRIA took leadership of the project and made the first public release on February the 1st 2010. Of the various scikits, scikit-learn as well as scikit-image were described as "well-maintained and popular" in November 2012.

4.6 Natural Language Processing

Natural language processing is the combination of computer science, computational linguistics and artificial intelligence. This field of study is related to the human-computer interactions. Natural language processing is the study that enables computers to understand the meaning of natural languages. The language research with the help of machines started back in 1950's. Modern NLP algorithms are mainly based on Machine learning and Statistics. Some of the major tasks in natural language processing are automatic summarization, translation, morphological segmentation, named entity recognition, semantic

analysis, optical character recognition, parsing, question answering, speech analysis, information extraction, information retrieval etc.

4.7 Natural Language Toolkit (NLTK) & NLP

One of the most important ingredients for our research is python's package NLTK. We have used NLTK for the little bit of NLP that we have in our research to find out many features as possible from the test essays before putting those features into our python machine learning code. After getting the data set from Kaggle site our next step would be to come up with a NLP system. NLTK would provide us with most of the basic requirements. There is not much say about NLTK because this package is so well known that whenever anyone wants to do some twitching with natural language thing with a machine and that person is new in this sort of field then his best option to go for NLTK and start playing with it. However, even NLTK has go to a long way albeit it has come a long from where it started but in order to handle the complexities in language this package has lots of potential rooms for improvements.

4.8 Numpy

The python programming language was not originally designed for numerical computational jobs however, it soon began to capture the attention of many scientific/engineering communities early on. Therefore, people have started working onto something that would allow researchers to use python for heavy calculation. Finally, in the early 2005 Numpy was released as a standalone package that would be very easy to install and light weighted as well. So Numpy is an extension of python programming language, adding support for large, multidimensional arrays and matrices along with a very elaborate library of high- level mathematical functions to operate on these arrays. Nonetheless, Numpy arrays must be views on contiguous memory buffers. A replacement package called Blaze attempts to overcome this limitation. Algorithms that are not expressible as a vectored operation will typically run slowly because they must be implemented in "pure Python", while vectorization may increase memory complexity of some operations from constant to linear, because temporary arrays must be created that are as large as the inputs. Runtime compilation of numerical code has been implemented by several groups to avoid these problems; open source solutions that interoperate with NumPy include `scipy`, `weave`, `numexpr` and `Numba`. Cython is a static-compiling alternative to these.

4.9 pandas

One of the most versatile and powerful data analysis toolkit in python is pandas. It is a python package that provides fast, reliable and expressive data structures designed to make working with both relational and labelled data. . It aims to be the fundamental high-level building block for doing practical, real world data analysis in Python. Additionally, it has the broader goal of becoming the most powerful and flexible open source data analysis / manipulation tool available in any language. It is already well on its way toward this goal. I have come across pandas when we have to feed our xls file into our algorithm. Is has the Robust IO tools for loading data from flat files (CSV and delimited), Excel files, databases, and saving / loading data from the ultrafast.

MACHINE LEARNING ALGORITHMS

5.1 Supervised Machine Learning

There are many machine learning algorithms. Machine learning algorithms can be divided into categories. In my thesis I have used supervised learning. Supervised learning problems can be further grouped into regression and classification problems. Supervised learning is useful in cases where a property (label) is available for a certain dataset (training set), but is missing and needs to be predicted for other instances

- **Classification:** A classification problem is when the output variable is a category, such as “red” or “blue” or “disease” and “no disease”.
- **Regression:** A regression problem is when the output variable is a real value, such as “dollars” or “weight”.

Some common types of problems built on top of classification and regression include recommendation and time series prediction respectively. Some popular examples of supervised machine learning algorithms are:

- Linear regression for regression problems.
- Random forest for classification and regression problems.
- Support vector machines for classification problems.

5.2 Unsupervised Machine Learning

Unsupervised learning problems can be further grouped into clustering and association problems.

- Clustering: A clustering problem is where you want to discover the inherent groupings in the data, such as grouping customers by purchasing behavior.
- Association: An association rule learning problem is where you want to discover rules that describe large portions of your data, such as people that buy X also tend to buy Y.

Some popular examples of unsupervised learning algorithms are:

- k-means for clustering problems.
- Apriori algorithm for association rule learning problems.

5.3 Support Vector Machine

“Support Vector Machine” (SVM) is a supervised machine learning algorithm which can be used for both classification or regression challenges. However, it is mostly used in classification problems. In this algorithm, I plot each data item as a point in n-dimensional space (where n is number of features you have) with the value of each feature being the value of a particular coordinate. Then, I perform classification by finding the hyper-plane that differentiate the two classes very well .

An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall.

In addition to performing linear classification, SVMs can efficiently perform a non-linear classification using what is called the kernel trick, implicitly mapping their inputs into high-dimensional feature spaces.

When data are not labeled, supervised learning is not possible, and an unsupervised learning approach is required, which attempts to find natural clustering of the data to groups, and then map new data to these formed groups. The clustering algorithm which provides an improvement to the support vector machines is called support vector clustering and is often[citation needed] used in industrial applications either when data are not labeled or when only some data are labeled as a preprocessing for a classification pass.

5.4 SGDRegressor

Linear model fitted by minimizing a regularized empirical loss with SGD SGD stands for Stochastic Gradient Descent: the gradient of the loss is estimated each sample at a time and the model is updated along the way with a decreasing strength schedule (aka learning rate). The regularizer is a penalty added to the loss function that shrinks model parameters towards the zero vector

using either the squared euclidean norm L2 or the absolute norm L1 or a combination of both (Elastic Net). If the parameter update crosses the 0.0 value because of the regularizer, the update is truncated to 0.0 to allow for learning sparse models and achieve online feature selection. This implementation works with data represented as dense numpy arrays of floating point values for the features.

5.5 Random Forest Algorithm

In 1980 Leo Breiman and Adele Cutler came up with an idea of an algorithm then can do both classification and regression. Later on Ho Tin Kam, Dietterich, Amit and Geman joined them and initiated the early foundation of the Random Forest algorithm. In addition to classification this algorithm can also handle the missing values or other outliers in data set, or other data exploration work. Other top pros that invoked us to use this algorithm are the following: it can handle thousands of variables without deleting them, maintains accuracy when large proportion of the dataset is missing in fact it estimates the values for the missing data and many more. The insight Behind Random forest is that instead of training one single classifier it trains multiple weak classifier that classifies some particular attributes and the target variables. Later on Random forest collects the result from each of the weak classifier and comes up with the actual classification. In other words we could say each weak classifier votes for its defined attributes and then creates the ultimate result. In case of regression the weak regressors comes with some continuous values and then their average is taken as the predicted output. It is mainly based on the popular decision tree or we could say it is based on the popular bagging approach. The algorithm is as follows (for both classification and regression)

- If the number of cases in the training set is N , sample N cases at random - but with replacement, from the original data. This sample will be the training set for growing the tree.
- If there are M input variables, a number $m \ll M$ is specified such that at each node, m variables are selected at random out of the M and the best split on these m is used to split the node. The value of m is held constant during the forest growing.
- Each tree is grown to the largest extent possible. There is no pruning.

Now with Random forest algorithm there comes error that depends these following two things

The correlation between any two trees in the forest. Increasing the correlation increases the forest error rate. And The strength of each individual tree in the forest. A tree with a low error rate is a strong classifier. Increasing the strength of the individual trees decreases the forest error rate.

5.6 K Nearest Neighbors

K nearest neighbors is a simple algorithm that stores all available cases and classifies new cases based on a similarity measure (e.g., distance functions). KNN has been used in statistical estimation and pattern recognition already in the beginning of 1970's as a non-parametric technique. A case is classified by a majority vote of its neighbors, with the case being assigned to the class most common amongst its K nearest neighbors measured by a distance function. If $K = 1$, then the case is simply assigned to the class of its nearest neighbor.

Distance functions

Euclidean	$\sqrt{\sum_{i=1}^k (x_i - y_i)^2}$
Manhattan	$\sum_{i=1}^k x_i - y_i $
Minkowski	$\left(\sum_{i=1}^k (x_i - y_i)^q \right)^{1/q}$

It should also be noted that all three distance measures are only valid for continuous variables. In the instance of categorical variables the Hamming distance must be used. It also brings up the issue of standardization of the numerical variables between 0 and 1 when there is a mixture of numerical and categorical variables in the dataset.

Hamming Distance

$$D_H = \sum_{i=1}^k |x_i - y_i|$$

$$x = y \Rightarrow D = 0$$

$$x \neq y \Rightarrow D = 1$$

X	Y	Distance
Male	Male	0
Male	Female	1

Choosing the optimal value for K is best done by first inspecting the data. In general, a large K value is more precise as it reduces the overall noise but there is no guarantee. Cross-validation is another way to retrospectively determine a good K value by using an independent dataset to validate the K value. Historically, the optimal K for most datasets has been between 3-10. That produces much better results than 1NN.

5.7 Naive Bayes classifier

In machine learning, naive Bayes classifiers are a family of simple probabilistic classifiers based on applying Bayes' theorem with strong (naive) independence assumptions between the features.

Naive Bayes has been studied extensively since the 1950s. It was introduced under a different name into the text retrieval community in the early 1960s and remains a popular (baseline) method for text categorization, the problem of judging documents as belonging to one category or the other (such as spam or legitimate, sports or politics, etc.) with word frequencies as the features. With appropriate pre-processing, it is competitive in this domain with more advanced methods including support vector machines. It also finds application in automatic medical diagnosis.

Naive Bayes classifiers are highly scalable, requiring a number of parameters linear in the number of variables (features/predictors) in a learning problem. Maximum-likelihood training can be done by evaluating a closed-form expression, which takes linear time, rather than by expensive iterative approximation as used for many other types of classifiers.

In the statistics and computer science literature, naive Bayes models are known under a variety of names, including simple Bayes and independence Bayes. All these names reference the use of Bayes' theorem in the classifier's decision rule, but naive Bayes is not (necessarily) a Bayesian method.

METHODOLOGY

6.1 Thesis Workflow

In this section I will discuss the workflow that we have maintained throughout our thesis work. The following flowchart demonstrates the procedures or work research approach. At the very beginning right after finding the problem we would like to work on, the first challenge was to collect relevant data to train the system. In most the cases I find data that is full of so much noises that I have to abandon most of it. So finding a usable dataset is very is very important for the research. After much searching I finally got some usable format of data in Kaggle. Though I had to clean the data. In some places but still they helped us a lot. In next chapter I have discussed this in detail how I have collected all the data and cleaned them to extract noticeable indicators to predict the score. Later on in that I have discussed how with the help of natural language toolkit we extracted all the features needed for our research. After getting a desired dataset in proper format to implement Support vector machine Random forest for classification and stochastic gradient descent regression for each of these case. Right after the preprocess of each features we then used tf-idf features in our final algorithms .

Collect essay from users and predict the score that the essay may get by an experienced human grader. I have also demonstrated the results and errors that I have found and shown the result and compared them with each other. And finally I have concluded and represented my thoughts on the future work that I am passionate to achieve.

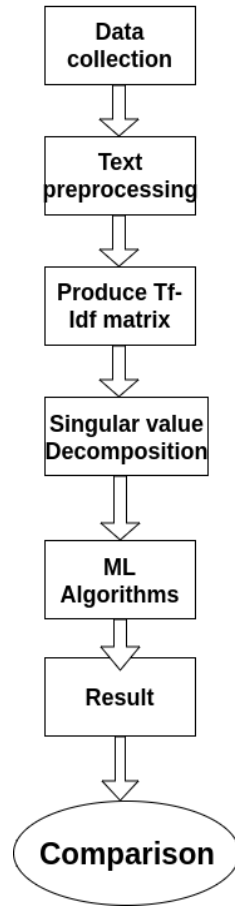
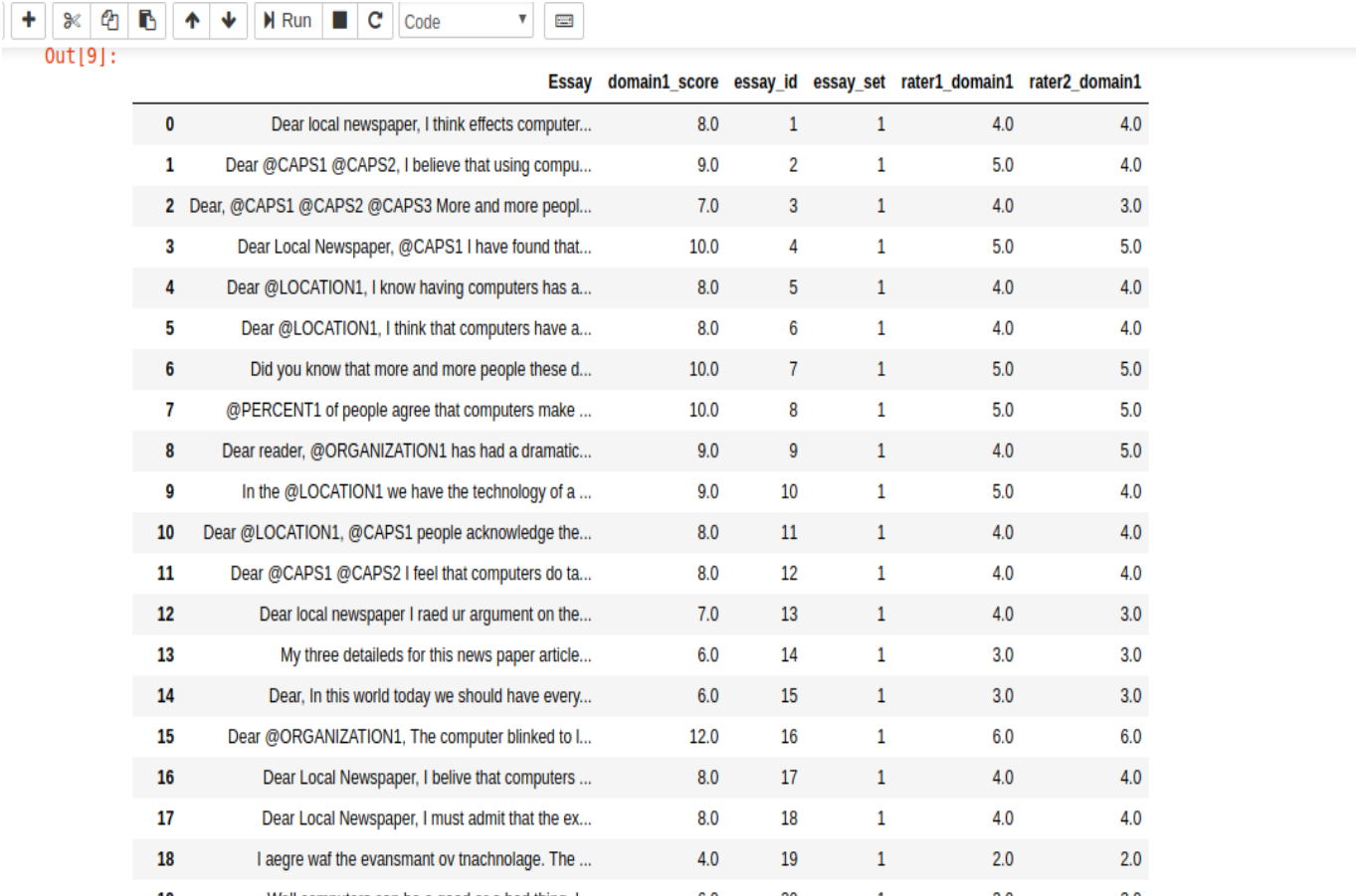


Figure 6.1: Thesis workflow .

6.2 Data collection

Supervised machine learning is totally dependent upon the condition of the data. A good and credible dataset can create the difference between good and bad machine learning agent. In our case finding a large dataset of essay that could be used in our research wasn't easy. After much hassle I have found the right usable dataset in Kaggle.com. Kaggle is a platform of the data science and predictive modeling. Numerous organizations put their valuable data in this platform to draw the attention of the researchers who could find a pattern and in the dataset or predict valuable output from that dataset. The dataset that I have used was provided by the William and Flora Hewlett foundation mostly known as the Hewlett foundation. They mostly sponsor data scientists and machine learning experts to solve social problems. Recently they have sponsored Automatic Essay Scoring and provided some quality data.

The dataset that I have used is a spreadsheet that contains nearly 17000 essays written by students of grade 7 to grade 10. The essay is divided into 8 sets. And these essays are marked by



Out[9]:

	Essay	domain1_score	essay_id	essay_set	rater1_domain1	rater2_domain1
0	Dear local newspaper, I think effects computer...	8.0	1	1	4.0	4.0
1	Dear @CAPS1 @CAPS2, I believe that using compu...	9.0	2	1	5.0	4.0
2	Dear, @CAPS1 @CAPS2 @CAPS3 More and more peopl...	7.0	3	1	4.0	3.0
3	Dear Local Newspaper, @CAPS1 I have found that...	10.0	4	1	5.0	5.0
4	Dear @LOCATION1, I know having computers has a...	8.0	5	1	4.0	4.0
5	Dear @LOCATION1, I think that computers have a...	8.0	6	1	4.0	4.0
6	Did you know that more and more people these d...	10.0	7	1	5.0	5.0
7	@PERCENT1 of people agree that computers make ...	10.0	8	1	5.0	5.0
8	Dear reader, @ORGANIZATION1 has had a dramatic...	9.0	9	1	4.0	5.0
9	In the @LOCATION1 we have the technology of a ...	9.0	10	1	5.0	4.0
10	Dear @LOCATION1, @CAPS1 people acknowledge the...	8.0	11	1	4.0	4.0
11	Dear @CAPS1 @CAPS2 I feel that computers do ta...	8.0	12	1	4.0	4.0
12	Dear local newspaper I raed ur argument on the...	7.0	13	1	4.0	3.0
13	My three detaileds for this news paper article...	6.0	14	1	3.0	3.0
14	Dear, In this world today we should have every...	6.0	15	1	3.0	3.0
15	Dear @ORGANIZATION1, The computer blinked to l...	12.0	16	1	6.0	6.0
16	Dear Local Newspaper, I belive that computers ...	8.0	17	1	4.0	4.0
17	Dear Local Newspaper, I must admit that the ex...	8.0	18	1	4.0	4.0
18	I aegre waf the evansmant ov tnachnolage. The ...	4.0	19	1	2.0	2.0
19	Well computers are a good as a bad thing. I	6.0	20	1	3.0	3.0

Figure 6.2: Data Set .

two human rater. The dataset contains the following parameters.

- Essay id: Its an unique identifier used to differentiate each students essay.
- Essay set : The essays varies in 8 different essay set number. Each set contains similar topic of the essay.
- Essay: this column contains the ASCII value of the essays or the essay texts written by the students.
- Rater1 domain1: the next column contains the essay score rated by a teacher
- Rater2 domain1: It contains the essay score rated by second teacher
- Domain1 score: It is the resolved score between the raters

A screenshot of the dataset is presented

6.2.1 Code

```
import pandas as pd

data = pd.read_excel('training_set_rel3.xls')
df = data[data['essay_set']==2]
x = df['essay']
Y = df['domain1_score']
```

6.3 Text Preprocessing

Text preprocessing is the first step of any data mining approach. Data preprocessing is needed to convert the raw unordered unusable data to structured usable format. Also dataset contains a lot of outliers and other noises that could affect the research in a negative manner. However the dataset that I have collected was very much in shape to be used in our research. Slightly modification was made to remove the outliers and other noises like some missing values removal and so on. Later on we have made the following modification to our dataset while using them so that I could use them in better way.

6.3.1 Word tokenization

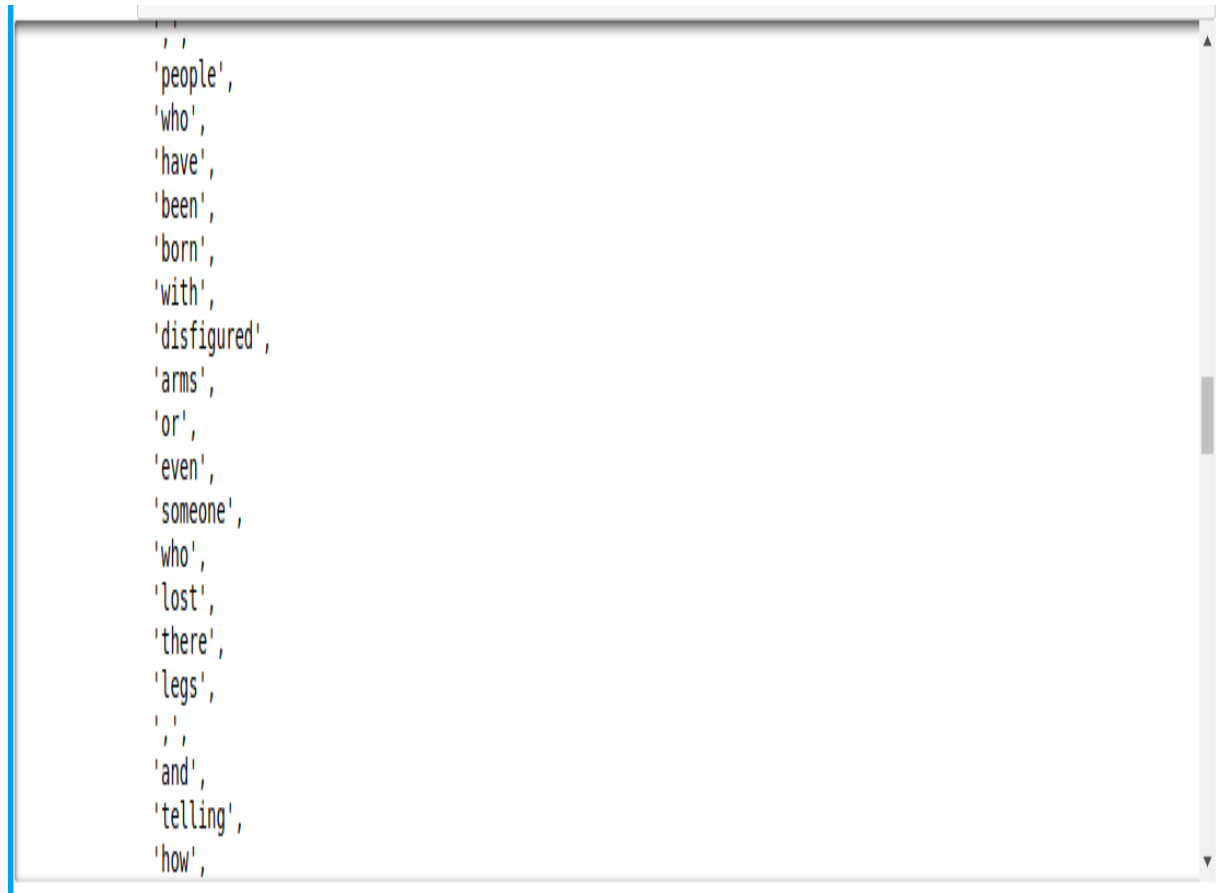
Tokenization is a processing of converting a large text into single pieces of tokens. In our case I have tokenized each essay before using them for feature extraction. I have mainly used python NLTK's word Tokenize and Sentence Tokenize module to tokenize of essay .

6.3.2 Removing Stop Words

In natural language processing Stop words is referred as a collection of words that is mostly present in each and every texts and that are mostly useless in natural language processing research. In fact removing those gives better result. These lists contains words like a, an, the etc. And also the punctuations like brackets and other symbols. In nltk there is a module named stop words that contains the list of words that could be removed from my essays. Though I need a slight modification to enrich the corpus of stop words as it does not include some of the most common words used today.

6.3.3 Stemming

Stemming is a process used in natural language processing that helps to transform the word to its base form or the stem form. For example “am, is, are” all of these can be transformed into base word “be”. Car, cars, car's, etc. can be traced down to car. For grammatical reason words



```
','  
'people',  
'who',  
'have',  
'been',  
'born',  
'with',  
'disfigured',  
'arms',  
'or',  
'even',  
'someone',  
'who',  
'lost',  
'there',  
'legs',  
,',  
'and',  
'telling',  
'how',  
,
```

Figure 6.3: Tokenized word.

are used differently in texts. If we can trace them down to their base form it will help us greatly in processing our data. Among many algorithms used to stem the words the porter algorithm is well appreciated. python nltk has a build in support for porter algorithm . We have used this algorithm and reduced the dimensionality of each essays.

6.3.4 Code

```
import nltk  
from nltk.stem.porter import PorterStemmer  
import string  
import re  
from nltk.stem.lancaster import LancasterStemmer  
from nltk.corpus import stopwords
```

```
num_regex = re.compile('^[+-]?[0-9]+\.[0-9]*$')

stemmer = LancasterStemmer()
def stem_tokens(tokens, stemmer):
    stemmed = []
    for item in tokens:
        stemmed.append(stemmer.stem(item))
    return stemmed

##### Stop Word Remove #####

stop_words = set(stopwords.words('english'))

#punctuation list
s = list(string.punctuation)

def tokenize(string):
    tokens = nltk.word_tokenize(string)
    for index, token in enumerate(tokens):
        if token == '@' and (index+1) < len(tokens):
            tokens[index+1] = '@' + re.sub('[0-9]+.*', '', tokens[index+1])
            tokens.pop(index)
    #remove english stop word from tokens
    filtered_sentence = [w for w in tokens if not w in stop_words]

    #remove punctuation
    tt = [i for i in filtered_sentence if not i in s]

    #Apply stemming
    stems_word = stem_tokens(tt, stemmer)

    return stems_word
```

6.4 Producing Tf-Idf Matrix

Tf-idf stands for term frequency-inverse document frequency, and the tf-idf weight is a weight often used in information retrieval and text mining. This weight is a statistical measure used to evaluate how important a word is to a document in a collection or corpus. The importance increases proportionally to the number of times a word appears in the document but is offset by the frequency of the word in the corpus. Variations of the tf-idf weighting scheme are often used by search engines as a central tool in scoring and ranking a document's relevance given a user query. One of the simplest ranking functions is computed by summing the tf-idf for each query term; many more sophisticated ranking functions are variants of this simple model. Tf-idf can be successfully used for stop-words filtering in various subject fields including text summarization and classification.

Typically, the tf-idf weight is composed by two terms: the first computes the normalized Term Frequency (TF), aka. the number of times a word appears in a document, divided by the total number of words in that document; the second term is the Inverse Document Frequency (IDF), computed as the logarithm of the number of the documents in the corpus divided by the number of documents where the specific term appears.

- **TF:** Term Frequency, which measures how frequently a term occurs in a document. Since every document is different in length, it is possible that a term would appear much more times in long documents than shorter ones. Thus, the term frequency is often divided by the document length (aka. the total number of terms in the document) as a way of normalization:

$$\text{TF}(t) = (\text{Number of times term } t \text{ appears in a document}) / (\text{Total number of terms in the document}).$$

- **IDF:** Inverse Document Frequency, which measures how important a term is. While computing TF, all terms are considered equally important. However it is known that certain terms, such as "is", "of", and "that", may appear a lot of times but have little importance. Thus we need to weigh down the frequent terms while scale up the rare ones, by computing the following:

$$\text{IDF}(t) = \log_e(\text{Total number of documents} / \text{Number of documents with term } t \text{ in it}).$$

I have used `TfidfVectorizer` from python `sklearn` to produce tf-idf matrix

6.4.1 Code

```
from sklearn.feature_extraction.text import TfidfVectorizer
```

```
tfidf = TfidfVectorizer(tokenizer=tokenize, min_df=1).fit(x)
X = tfidf.transform(x)
```

6.5 Singular Value Decomposition (SVD)

In linear algebra, the singular-value decomposition (SVD) is a factorization of a real or complex matrix. It is the generalization of the eigendecomposition of a positive semidefinite normal matrix (for example, a symmetric matrix with positive eigenvalues) to any $m \times n$ matrix via an extension of the polar decomposition. It has many useful applications in signal processing and statistics.

if r -rank matrix M . After apply svd

$$M = U \Sigma V^T$$

Where U and V are orthogonal matrices, and the elements in Σ are singular values those are in descending order. Specially, in natural language processing, maintaining only $k \ll r$ will produce a lower dimensionality and better approximation about the original matrix M . By removing the $(r - k)$ diagonal elements, $(r - k)$ columns in U and $(r - k)$ rows in V and $(r - k)$ elements in Σ where the elements with far too small values are considered to be noise and unnecessary, we can multiply the matrices and get the approximation to the original matrix:

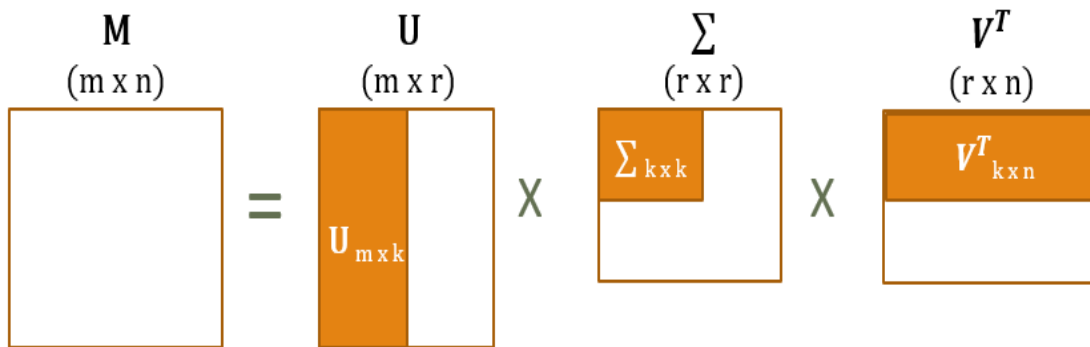


Figure 6.4: The Formation of singular value decomposition

The dimensionality reduction operation has been done by removing one or more smallest singular values from singular matrix S and also deleted the same number of columns and rows from U and V , respectively. The purpose of the dimensionality reduction is to reduce the noise and unimportant details in the data so that the underlying semantic structure can be used to compare the content of essays. The truncated SVD matrices have been used for making the training essay vectors. I have used python numpy for producing svd

6.5.1 Code for U,S,V

```
import numpy

M = X.toarray()
U, s, V = np.linalg.svd(M, full_matrices=False)
```

REGRESSION AND CLASSIFICATION MODEL

In this case I have taken result after dimensionality reduction by singular value decomposition and fit to the both regression and classification model .

7.1 Regression

In this case I have taken SGDRegressor and RandomForestRegressor form python scikit learn. After this I have tested with some essay for prediction

7.1.1 SGDRegressor

Stochastic Gradient Descent (SGD) is a simple yet very efficient approach to discriminative learning of linear classifiers under convex loss functions such as (linear) Support Vector Machines and Logistic Regression. Even though SGD has been around in the machine learning community for a long time, it has received a considerable amount of attention just recently in the context of large-scale learning. SGD has been successfully applied to large-scale and sparse machine learning problems often encountered in text classification and natural language processing. Given that the data is sparse, the classifiers in this module easily scale to problems with more than 100000 training examples and more than 100000 features.

mean squared log error	0.023011
mean absolute error	0.438274
median absolute error	0.363322
mean squared error	0.326005
r2_score	0.450000
variance	0.479017
min_max accuracy	0.874330

Table 7.1: Results of SGDRegressor

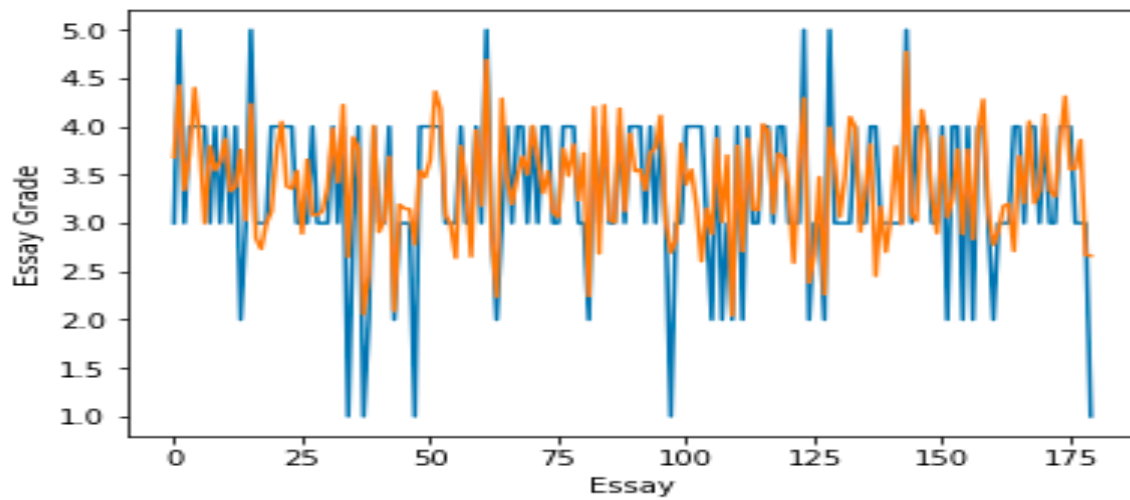


Figure 7.1: Graph of Actual and predicted value of SGDRegression model

Actual	Predicted
3.0	3.940235
5.0	4.581626
3.0	3.470368
4.0	3.603071
4.0	4.548583
4.0	3.794644
4.0	2.999502
3.0	3.717935
4.0	3.377777
3.0	3.497000
4.0	3.818070
3.0	3.454656
4.0	3.143791
2.0	3.667493
3.0	3.041063
5.0	4.316815
3.0	2.786516

Figure 7.2: Actual and predicted value of SGDregression model

7.1.2 RandomForestRegressor

A random forest is a meta estimator that fits a number of classifying decision trees on various sub-samples of the dataset and use averaging to improve the predictive accuracy and control over-fitting. The sub-sample size is always the same as the original input sample size but the samples are drawn with replacement if `bootstrap=True` (default).

mean squared log error	0.030639
mean absolute error	0.513528
median absolute error	0.473581
mean squared error	0.426575
r2_score	0.314432
variance	0.316639
min_max accuracy	0.856876

Table 7.2: Results of RandomForestRegressor

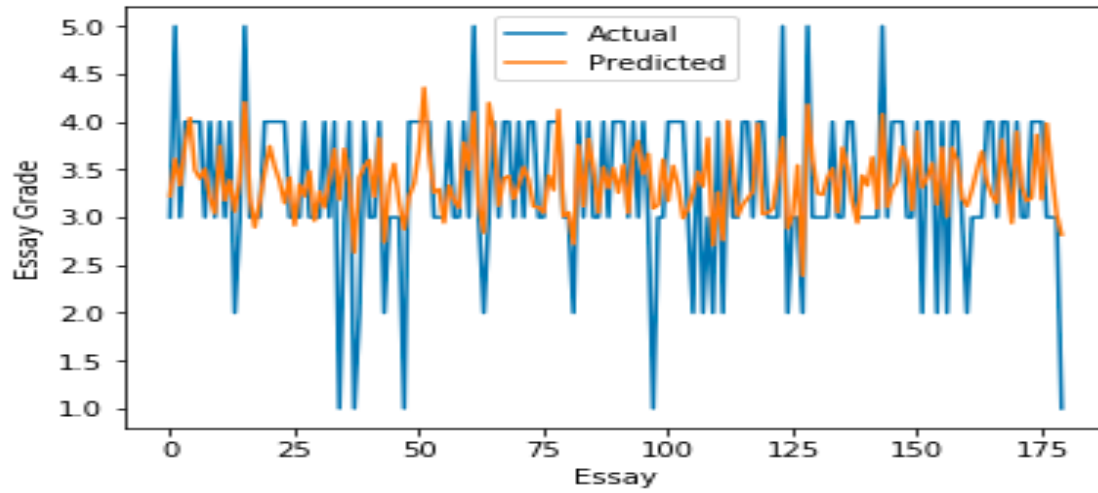


Figure 7.3: Graph of Actual and predicted value of RandomForestRegressor model

Actual	Predicted
3.0	3.222484
5.0	3.612401
3.0	3.334120
4.0	3.745250
4.0	4.040482
4.0	3.495385
4.0	3.405315
3.0	3.508833
4.0	3.204099
3.0	3.043552
4.0	3.746589
3.0	3.175400
4.0	3.386257
2.0	3.060246
3.0	3.398333
5.0	4.205212
3.0	3.217588

Figure 7.4: Actual and predicted value of RandomForestRegressor model

7.2 Classification

In this case I have taken BernoulliNB, KNeighborsClassifier, RandomForestClassifier and svm from python scikit learn . After this I have tested with some essay for prediction

7.2.1 BernoulliNB

Naive Bayes classifier for multivariate Bernoulli models. Like MultinomialNB, this classifier is suitable for discrete data. The difference is that while MultinomialNB works with occurrence counts, BernoulliNB is designed for binary/boolean features.

The mean accuracy is 63.33333333333333%

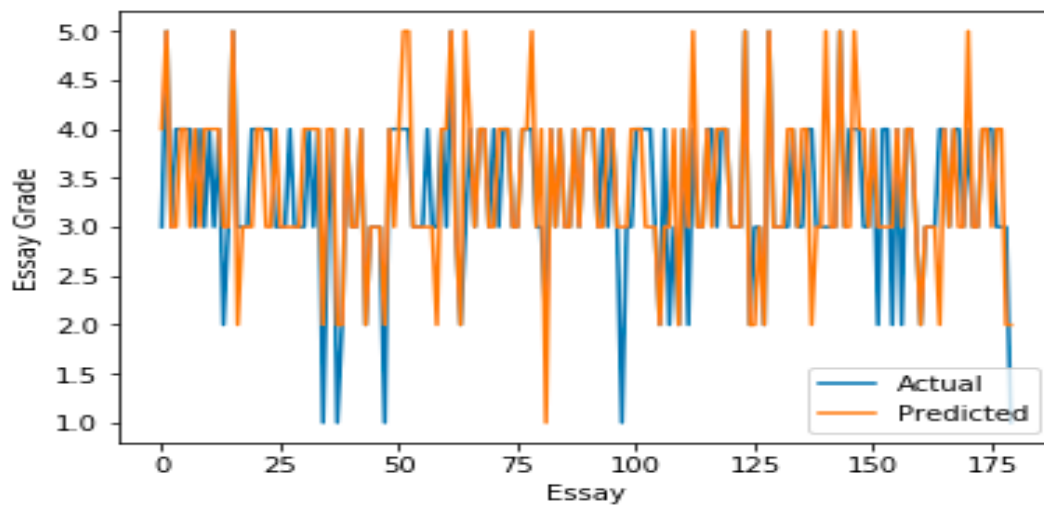


Figure 7.5: Graph of Actual and predicted value of BernoulliNB model

Actual	Predicted
3.0	4.0
5.0	5.0
3.0	3.0
4.0	3.0
4.0	4.0
4.0	4.0
4.0	3.0
3.0	4.0
4.0	3.0
3.0	4.0
4.0	4.0
3.0	4.0
4.0	4.0
2.0	3.0
3.0	3.0
5.0	5.0
3.0	2.0

Figure 7.6: Actual and predicted value of BernoulliNB model

7.2.2 KNeighborsClassifier

The k-neighbors classification in KNeighborsClassifier is the more commonly used of the two techniques. The optimal choice of the value k is highly data-dependent: in general a larger k suppresses the effects of noise, but makes the classification boundaries less distinct.

The mean accuracy is 49.444444444444446%

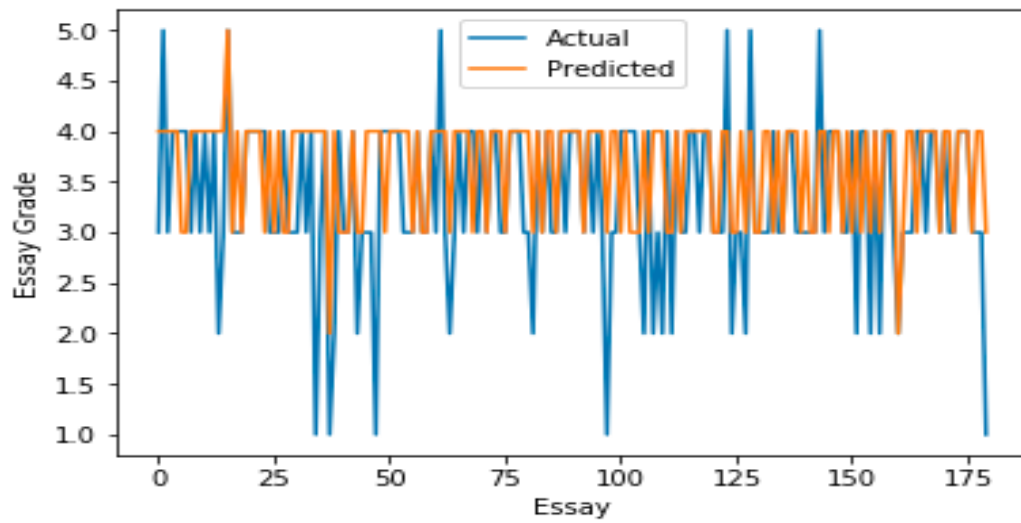


Figure 7.7: Graph of Actual and predicted value of KNeighborsClassifier model

Actual	Predicted
3.0	4.0
5.0	4.0
3.0	4.0
4.0	4.0
4.0	4.0
4.0	3.0
4.0	3.0
3.0	4.0
4.0	4.0
3.0	4.0
4.0	4.0
3.0	4.0
4.0	4.0
2.0	4.0
3.0	4.0
5.0	5.0
3.0	3.0
3.0	4.0

Figure 7.8: Actual and predicted value of KNeighborsClassifier model

7.2.3 RandomForestClassifier

A random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and use averaging to improve the predictive accuracy and control over-fitting. The sub-sample size is always the same as the original input sample size but the samples are drawn with replacement if `bootstrap=True` (default). The mean accuracy is 57.77777777777772%

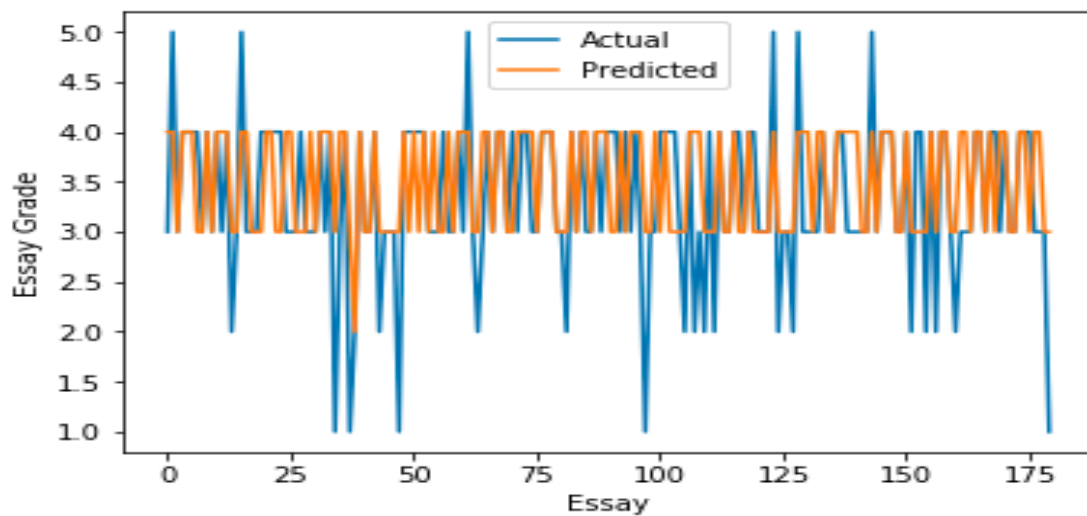


Figure 7.9: Graph of Actual and predicted value of RandomForestClassifier model

Actual	Predicted
3.0	4.0
5.0	4.0
3.0	3.0
4.0	4.0
4.0	4.0
4.0	4.0
4.0	3.0
3.0	3.0
4.0	4.0
3.0	3.0
4.0	4.0
3.0	4.0
4.0	4.0
2.0	3.0
3.0	3.0
5.0	4.0

Figure 7.10: Actual and predicted value of RandomForestClassifier model

7.2.4 Support Vector Machine

Support vector machines (SVMs) are a set of supervised learning methods used for classification, regression and outliers detection. The advantages of support vector machines are:

Effective in high dimensional spaces.

Still effective in cases where number of dimensions is greater than the number of samples.

Uses a subset of training points in the decision function (called support vectors), so it is also memory efficient.

Versatile: different Kernel functions can be specified for the decision function. Common kernels are provided, but it is also possible to specify custom kernels.

The mean accuracy is 65.55555%

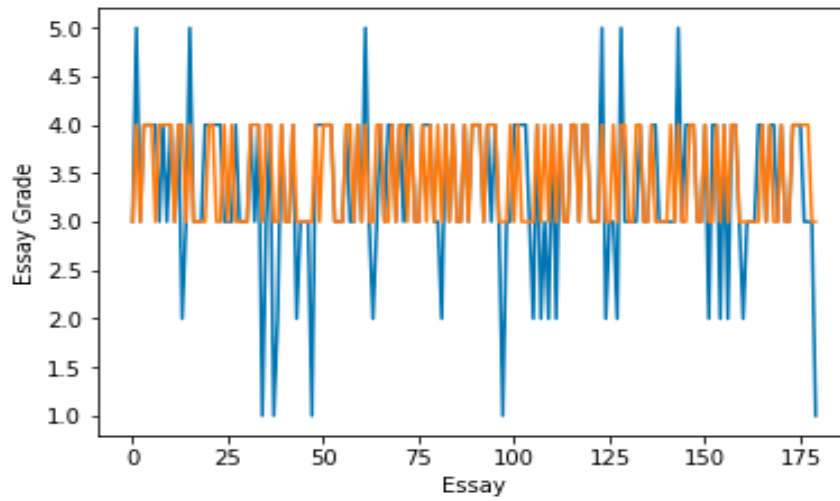


Figure 7.11: Graph of Actual and predicted value of svm model

Actual	Predicted
3.0	3.0
5.0	4.0
3.0	3.0
4.0	4.0
4.0	4.0
4.0	4.0
4.0	3.0
3.0	4.0
4.0	4.0
3.0	4.0
4.0	4.0
3.0	3.0
4.0	4.0
2.0	4.0
3.0	3.0
5.0	4.0
3.0	3.0

Figure 7.12: Actual and predicted value of svm model

RESULTS AND COMPARISON

In this case i have compared different algorithm with the result of mean squared error.

8.1 R Squared Score

R-squared is a statistical measure of how close the data are to the fitted regression line. It is also known as the coefficient of determination, or the coefficient of multiple determination for multiple regression.

The definition of R-squared is fairly straight-forward; it is the percentage of the response variable variation that is explained by a linear model. Or:

$$\text{R-squared} = \text{Explained variation} / \text{Total variation}$$

R-squared is always between 0 and 100%:

0% indicates that the model explains none of the variability of the response data around its mean.

100% indicates that the model explains all the variability of the response data around its mean.

Algorithm	R Squared score
SGDRegressor	0.450000
RandomForestRegressor	0.314432

Table 8.1: R Squared Score

8.2 Variance

The `explained_variance_score` computes the explained variance regression score.

Algorithm	Variance score
SGDRegressor	0.479017
RandomForestRegressor	0.316639

Table 8.2: Variance Score

From the above r-squared score, variance score and mean squared error SGDRegressor is better than RandomForestRegressor.

8.3 Accuracy score

The `accuracy_score` function computes the accuracy, either the fraction (default) or the count (normalize=False) of correct predictions. In multilabel classification, the function returns the subset accuracy. If the entire set of predicted labels for a sample strictly match with the true set of labels, then the subset accuracy is 1.0; otherwise it is 0.0.

Algorithm	Accuracy_score
SVM	65.555555%
RandomForestClassifier	57.7777772%
KNeighborsClassifier	49.444444447%
BernoulliNB	63.333333333333%

Table 8.3: Accuracy Score

8.4 Precision score

The value is between 0 and 1 and higher is better.

Algorithm	Precision_score
SVM	0.263116
RandomForestClassifier	0.231140
KNeighborsClassifier	0.501014
BernoulliNB	0.447211

Table 8.4: Precision score

8.5 recall score

The recall is the ratio $tp / (tp + fn)$ where tp is the number of true positives and fn the number of false negatives. The recall is intuitively the ability of the classifier to find all the positive samples.

The best value is 1 and the worst value is 0.

Algorithm	recall_score
SVM	0.306511
RandomForestClassifier	0.255505
KNeighborsClassifier	0.276635
BernoulliNB	0.565757

Table 8.5: recall_score

Finally Classification algorithms results is :

Algorithm	recall_score	precision_score	Accuracy_score
SVM	0.306511	0.263116	0.65555
RandomForestClassifier	0.255505	0.231140	0.577777
KNeighborsClassifier	0.276635	0.501014	0.49444447
BernoulliNB	0.565757	0.447211	0.63333333

Table 8.6: Classification Result

From the above table of Classification Result average score of BernoulliNB is high than other so BernoulliNB is better algorithm than others algorithms for the purpose

8.6 Mean squared error

In statistics, the mean squared error (MSE) or mean squared deviation (MSD) of an estimator (of a procedure for estimating an unobserved quantity) measures the average of the squares of the errors or deviations—that is, the difference between the estimator and what is estimated. MSE is a risk function, corresponding to the expected value of the squared error loss or quadratic loss. The difference occurs because of randomness or because the estimator doesn't account for information that could produce a more accurate estimate.

The MSE is a measure of the quality of an estimator—it is always non-negative, and values closer to zero are better.

The MSE is the second moment (about the origin) of the error, and thus incorporates both the variance of the estimator and its bias. For an unbiased estimator, the MSE is the variance of the estimator. Like the variance, MSE has the same units of measurement as the square of the quantity being estimated. In an analogy to standard deviation, taking the square root of MSE yields the root-mean-square error or root-mean-square deviation (RMSE or RMSD), which has the same units as the quantity being estimated; for an unbiased estimator, the RMSE is the square root of the variance, known as the standard deviation.

Algorithm	Mean Squared Error
SGDRegressor	0.325971
RandomForestRegressor	0.426575
Support Vector Machine	0.444444
BernoulliNB	0.450000
RandomForestClassifier	0.538889
KNeighborsClassifier	0.727778

Table 8.7: Mean Squared error

CONCLUSION AND FUTURE WORK

9.1 Conclusion

My research shows that how machine learning algorithm works for automated essay grading . It also tells us that natural language processing has come a long way from where it started its journey and maybe because of those earlier ground breaking research. Nonetheless, it is quite evident what machine learning can do if the agent can be trained properly with large set of relevant data and reasonably good algorithm will even make the learning easier.

9.2 Future work

I strongly think that with the machine leaning algorithm that I have used in my research could be graded essay easily . Machine learning agents can be trained with more efficient algorithms than the results obtained would be more accurate and robust. I will use neural network for the grading purpose to solve the problem. However, we firmly believe that this research is just a drop in the ocean for what is about to come in the further in terms of natural language and teaching writers on how to improve their creative writing.

GLOSSARY

- Mean squared error: In statistics, the mean squared error (MSE) or mean squared deviation (MSD) of an estimator (of a procedure for estimating an unobserved quantity) measures the average of the squares of the errors or deviations—that is, the difference between the estimator and what is estimated.
- R-squared: it is known as “multiple R”, and it is equal to the correlation between the dependent variable and the regression model’s predictions for it.
- The `explained_variance_score` : it computes the explained variance regression score.
- Machine learning : it is a field of computer science that gives computers the ability to learn without being explicitly programmed.
- Natural language processing (NLP) :it is a field of computer science, artificial intelligence concerned with the interactions between computers and human (natural) languages, and, in particular, concerned with programming computers to fruitfully process large natural language data.

REFERENCE

- 1) Andrew M. Olney, “Generalizing latent semantic analysis,” IEEE international conference on semantic computing, 2009
- 2) Mingqing Zhang, Shudong Hao and Yanyan Xu, “Automated essay scoring using incremental latent semantic analysis,” Journal of software, vol. 9, no. 2, 2014
- 3)M. M. Islam and A. Hoque, “Automated essay scoring using generalized latent semantic analysis,” Journal of Computers, vol. 7, no. 3, pp. 616–626, 2012.
- 4)Purti Ratna, Henry Artajaya and Anantasatya Adhi, “GLSA based online essay grading system,” IEEE international conference on teaching, assessment and learning for engineering, 2013
- 5) Y. Cao and C. Yang, “Automated chinese essay scoring with latent semantic analysis,” Examinations Research, vol. 3, no. 1, pp. 63–71, 2007.
- 6) Burstein,J.,Wolff,S.,Lu,C.,& Kaplan,R. (1997) . An Automatic Scoring System for Advanced Placement Biology Essays. Proceedings of Fifth Conference on Applied Natural Language Processing(pp.174181). Washington D.C . : Association for Computational Linguistics
- 7) A. Fazal, T. Dillon, and E. Chang, “Noise reduction in essay datasets for automated essay grading,” in On the Move to Meaningful Internet Systems: OTM 2011 Workshops. Springer, 2011, pp. 484–493.
- 8) R. A. Hardy, “Examining the costs of performance assessment, Applied Measurement in Education, vol. 8, no. 2,pp. 121–134, 1995.
- 9)S. Valenti, F. Neri, and A. Cucchiarelli, “An overview of current research on automated essay grading,” Journal of Information Technology Education, vol. 2, 2003, pp. 319-330.
- 10)Y. Attali, and J. Burstein, “Automated essay scoring with e-rater® V.2,” The Journal of Technology, Learning and Assessment, vol. 4, no. 3, 2006.