PREDICTING ANSWERS TO YES/NO QUESTIONS WITH SUPERVISED LEARNING

Zeynep Öncü
ANR: 995721

THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE IN COMMUNICATION AND INFORMATION SCIENCES, MASTER TRACK DATA SCIENCE BUSINESS & GOVERNANCE,
AT THE SCHOOL OF HUMANITIES AND DIGITAL SCIENCES
OF TILBURG UNIVERSITY

Thesis Committee:
Dr. A. T. Hendrickson
Dr. H. J. Brighton

Tilburg University
School of Humanities and Digital Sciences Department of Cognitive Science & Artificial Intelligence
Tilburg, The Netherlands
January 2019

Preface

First and foremost, I would like to thank Drew Hendrickson for providing me with such an interesting topic. He has helped me throughout my research, answered my endless questions and always motivated me by looking from different angles at this project. Next, I would like to thank Thiago Castro Ferreira who helped me with my questions and explained the topics that I was interested.

I would like to thank my family and especially, my parents who always supported me throughout this master's degree, my struggles and achievements. They always helped me with everything in this process. I would like to thank all my friends who showed me the brighter side of the world. I would like to thank Maike Fischer for helping with everything and the choice of this master's degree, Mehrnaz Pour-Morshed and Claudia Wegter for spending the endless days and nights studying with me. Lastly, I would like to thank my dear Çağdaş Tekin, who supported and helped me throughout this research by sharing his skills and knowledge that contributed to me in my master's degree.

Abstract

This study is a comparative analysis of supervised learning tasks using categorical texts and free-text questions about facial details of people. The datasets used for this research were collected through online experiments. This study used pre-trained word representations, which are known to be perform better than the traditional text mining approaches. Categorical texts and word embeddings of question texts were used as features in classification algorithms and their performances were evaluated with accuracy rates in predicting answers to the questions. In other words, the research question was formed as “Which pre-trained word embedding and classification models perform best in terms of accuracy rate in predicting answers to yes/no questions?”

The findings showed that use of pre-trained word embedding models indeed led to better predictive performances in some classification algorithms compared to the baseline for this dataset.

Table of Contents

Table of Figures

Table of Tables

1. Introduction

Machine learning and artificial intelligence have become prominent instruments in the fields of information, security, entertainment, law enforcement, and surveillance (Zhao, Chellappa, Phillips, & Rosenfeld, 2003). These technologies are present in our daily lives, for example face recognition is used to unlock our phones and predictive text technology is used to speed up our texting. Widespread use of the Internet generates an excessive amount of text data and this creates a surge of interest for the text mining approaches that are used to extract meaningful information from the data. A large amount of research is devoted to the methods and techniques of text mining:: feature extraction, information retrieval, and natural language processing (Allahyari et al., 2017). This reseach used text mining approaches to determine which pre-trained word embedding and classification models perform best best in terms of accuracy rate in predicting answers to yes/no questions.

Harari (2015) expressed in his “Sapiens” book, “We can connect a limited number of sounds and signs to produce an infinite number of sentences, each with a distinct meaning. We can thereby ingest, store, and communicate a prodigious amount of information about the surrounding world..” Artificial intelligence and machine learning solutions can be applied to replicate similar tasks with enough data and appropriate models. The data first needs to be pre-processed before it can be used in machine learning algorithms. Categorical data, such as values for eye colors, could easily be represented in numerical form by assigning numbers to each value (for example, 1 for green, 2 for blue, etc.). On the other hand, free-form text is difficult to categorize as it can be in many different forms as opposed to categorical values. Then, how can free-form text data be represented for machine learning algorithms to utilize it?

There are existing text mining solutions that aim to answer this question. After preprocessing steps such as cleaning, filtering, tokenization, lemmatization, and stemming, the most common way is to create a vector space model (VSM). VSM is an algebraic model, where the total number of words in a document is the total number of dimensions and each word is represented with a vector indicating its position in the vector space in relation to other words. This vector is referred to as the “weight” of the word or word embedding (Bengio, Courville, & Vincent, 2014), and commonly calculated from the co-occurrences of the words in documents (Allahyari et al., 2017). However, only applying this method did not produce sufficient results with the machine learning models. Therefore, some re-weighting techniques were applied to boost the context informativeness of vector representation of texts and reduce the dimensionality of VSMs (Baroni, Dinu, & Kruszewski, 2014).

In recent years, application of predictive algorithms such as deep neural networks has improved word embedding models. The word representations obtained from these models were proven to be better for feature engineering methods in text mining compared to those obtained from traditional frequency-based approaches. (Baroni et al., 2014). Mikolov, Sutskever, Chen, Corrado, and Dean (2013), and many others have developed predictive models in order to achieve computationally efficient and rather accurate

contextual word weights. The prominent predictive word embedding models throughout the literature are Word2Vec, FastText, and GloVe. They use simple distance functions to calculate similarity between two or more words, and they can provide a list of words similar to a given word. Moreover, there are pre-trained models that were trained on vast amounts of text data and these models are freely accessible online. As the pre-trained models eliminate the time and effort needed to train on large corpuses, they have gained attention of many researchers.

There are past studies that compare various word embedding models (Naili, Chaibi, & Ben Ghezala, 2017) or use the pre-trained word representations in machine learning algorithms to evaluate the performances (Eisner, Rocktäschel, Augenstein, Bošnjak, & Riedel, 2016). More information about other works that inspired this study can be found in Related Work section. There are only a few pre-trained word embedding models that can be obtained online, and they could be used for many research projects. Even though many studies have attempted to create large datasets of face descriptions for similar machine learning tasks (Gatt et al., 2018), there seems to be a gap in the literature as these pre-trained word embedding models had never been used in a study involving people's physical attributes. Therefore, this research is devoted to obtain new findings by using existing models and evaluating supervised learning algorithms.

Therefore, the following research question is proposed "Which pre-trained word embedding and classification models perform best in terms of accuracy rate in predicting answers to yes/no questions?". Several steps were taken to answer this question, and these discussed thoroughly in this paper. Pre-trained word embedding models were used to obtain new representations of the question texts. These word representations and some categorical variables were used as features in the classification models to predict the "yes" or "no" answers. Accuracy rates of classification algorithms were used to measure their performances. Principal Component Analysis was applied to reduce the dimensionality in the dataset, and the performances of the classification algorithms were re-calculated. Overall results were compared by creating two different datasets, Word2Vec and FastText, in order to determine which pre-trained word embedding model and classification algorithm perform best.

In the Results section, the baseline model for the classification algorithms was determined to be 56% in both Word2Vec and FastText datasets. The findings showed that Random Forest algorithm had the highest accuracy rate among the classification algorithms, which were compared and led to 67.2% and 67.4% in accuracy rates on Word2Vec and FastText datasets respectively.

2. Related Work

This section will introduce the topics that will be discussed in this paper and explain them under the related subheadings. It will give insight about the theoretical background, relevant works conducted and methods that were used in order to clarify the approach and goal of this research.

2.1. Description of Face

Perception, recognition, and identification of patterns are easy and routine tasks of people's daily lives. However, building a similar concept in machines is an ongoing research area. Face recognition is a machine's capability to identify or verify patterns of faces given an image of one or more persons in a scene. This requires a large amount of training and test data. Therefore, machine learning models are constantly improved and modified to better approach the specific problem at hand. Emerging technologies enable the techniques of image processing, pattern recognition, neural networks, computer vision, and computer graphics (Zhao et al., 2003). Available information regarding age, gender, ethnicity, hair, and the eye color of the subject may improve the recognition or verification of the face. This was described by Zhao et al. (2003), as segmentation of the face could help the outcome of the search. For example, there are billions of images online; in order to access a specific image by giving keywords, the engine utilizes the process of multi-label classification by querying through the tags and descriptions similar to the image (Datta, Joshi, Li, & Wang, 2008).

While describing objects, artifacts, or people is a standard process in human communication, for machines, it can be a difficult task since it requires a deeper understanding of language in specific domains such as identification and understanding of distinct features. Consequently, generating and using descriptive texts from images have been interesting and prominent research topics. The research by Gatt et al. (2018) stated that descriptive texts generated by humans are often feature-based and emphasized distinct physical attributes (such as eye and hair color, shape of the face, and ethnicity) and sometimes contained the subject's emotional state. Additionally, these descriptions might contain inferences or analogy (Gatt et al., 2018). However, direct text description generation by machines was stated that it relied on deep learning approaches and required a large amount of training and test data. Further machine generated texts often tended to be repetitive and similar words might be used (Gatt et al., 2018).

The models used for evaluation of machine text generation depends on human judgement, known to be expensive and hard to obtain in good quality (in terms of objectivity and independence). As a result, many studies used human generated texts from either existing databases or crowd-sourcing experiments (Hodosh, Young, & Hockenmaier, 2015; Gatt et al., 2018). Similar to the mentioned studies, this research data was collected through crowd-sourcing, and the text descriptions were generated by the participants of the experiments. This factor allowed to rely on the information without using any evaluation metrics, but considering that human judgment might contain analogy or inferences (Gatt et al., 2018). Furthermore,

the dataset also contained categorical information (collected by multiple choice-single answer format) regarding the segmentation of the faces, and this information was used in the analysis.

2.2. Information Extraction in Text Mining

As mentioned in the introduction, the core of this study concerns text data. Therefore, it is important to recall some definitions in this subject frame. Text mining approaches and methods are related to data mining, which are steps that are taken in the process to discover useful patterns in data through various models. Information extraction is defined as discovering information from unstructured or semi-structured text and is an important task for text mining (Aggarwal & Zhai, 2013; Allahyari et al., 2017). There are different approaches in text mining that was also used in this research, such as natural language processing. This approach utilizes artificial intelligence, computer science, statistics, and linguistic models to understand the natural language (Allahyari et al., 2017).

2.3. Supervised learning

Unsupervised and supervised learning algorithms are used to retrieve the informational patterns in the data. Briefly, unsupervised learning is undertaken to detect the hidden structure from unlabeled data. Supervised learning is implemented to learn the patterns in the labelled training data in order to make predictions on the unseen or test data (Allahyari et al., 2017). This research tried to find useful patterns in the datasets and used the extracted information from texts to make predictions by using supervised learning algorithms. However, the pre-trained word embedding models that were used in this research were built by unsupervised learning tools.

There is a broad range of classifiers such as Decision Trees, Random Forest, Naïve Bayes, Support Vector Machines, Logistic Regression, and K-Nearest Neighbors etc., which are widely used methods in text mining. In supervised learning models, high dimensionality of the data may affect the performance of the classifiers and cause overfitting. Allahyari et al. (2017) noted that Support Vector Machines are extensively used for text classification models because they draw a linear or non-linear hyperplane between the classes and are fairly robust algorithms that perform sufficient in a highly dimensional feature space. For this reason, it can be an ideal model for text classification and widely used for pattern recognition.

2.4. Evaluation of the Supervised Learning Models

Performance evaluation of classification models was implemented by dividing the data into subsets of trainingand test sets (usually 80:20 or 70:30 ratios are used in research studies). Commonly, classifiers are trained on the training set and predictions are made on the test set (Allahyari et al., 2017). The results of the predictions and actual labels of the test set are recorded in the confusion matrix. The labels, true positive and true negative, are the number of times positive or negative predictions were made

correctly. False positive and false negative correspond to the number of times predicted positive and negative values were made incorrectly. Accuracy is used to evaluate the model and defined as the ratio that correctly classified labels divided into the total number of labels. Furthermore, precision is a measure that calculates the predicted labels among the identified positive labels, recall is the fraction of the correct labels among all the positive labels, and F-1 score is the geometric mean of precision and recall (Allahyari et al., 2017). In this research the accuracy rate was used as an evaluation metric since the aim was to measure the overall performance of the classifiers by the proportion of correctly predicted results.

2.5. Word representations

The performance of machine learning algorithms extensively relies on the choices of the features and data representation. Therefore, when classifiers can extract meaningful information from the data, they yield better predictions (Bengio et al., 2014). Text data is the simplest form of unstructured information. While it is easily perceived and processed by humans, it may be harder for machines to process the language and structure (Allahyari et al., 2017). There are several word representation approaches, such as clustering-based and distributed representations approaches. This study focused on the distributed approaches. Bengio, Ratinov and Turian (2010) defined this notion as dense, low-dimensional, and real-valued word representations that were also called word embedding. In other words, it is a mathematical expression related to each word and often times it is defined as a vector in machines.

Each number in a word vector array represents a feature that could give semantic and syntactic information regarding the word. Moreover, the length of the array would be equal to the vocabulary size and this array can be associated to a dictionary containing all unique words in the sentence (Bengio et al., 2010). For example, the sentence “quick brown fox jumps over the lazy dog” the dictionary would be ['quick', 'brown', 'fox', 'jumps', 'over', 'the', 'lazy', 'dog']. A vector representation of a word (also referred to as a one-hot encoded vector) is the number, 1, stands for the position where the word occurs and the number, 0, for the other words in the sentence. The word “jumps” in vector representation of that sentence in the dictionary would be [0,0,0,1,0,0,0,0]. Accordingly, it can be assumed that each word can be represented independently and would share an eight-dimensional space. However, distributed representations introduce dependencies between words, reduce the sparsity of the vectoral space by compressing the featural space to only highlight the most informative descriptions of the words (Mikolov, Chen, Corrado, & Dean, 2013).

2.6. Word Embedding Models

Deep neural networks are very efficient in terms of pattern recognition and are extensively used in various areas of research in text mining, face and speech recognition. Word embedding models are often constructed from deep neural networks underlying predictive learning models. There are two types of word embedding models, frequency-based word embedding and prediction-based word embedding

(Bengio et al., 2010). Frequency-based word embedding is the technique of counting the word occurrences, co-occurrences, or TF-IDF (estimating how important the word is). Prediction-based word embedding utilizes neural network to estimate the weights of the words using the combination of multiple models (NNS, 2018).

Large numbers of studies have investigated and improved the use of word embedding models, even though this process is often slow. For instance, Shen et al. (2015) created a word embedding correlation model based on translation methods in order to capture word similarities between word translation and Q&A pairs. In the study, questions were given for three topics and answers were matched accordingly by using the semantic similarities and predictive algorithms. As a result, the model outperformed the state-of-art models used for predictive translation (Shen et al., 2015). Prediction-based word embedding models are represented as distributed models. They will be further explained along with the description of the pre-trained word embedding models in the following Methods section.

2.7. Pre-trained word embedding models

In recent years, leading technology entities such as Google, Facebook and Stanford Group have developed powerful pre-trained word embedding models, and they are freely available. The researchers were interested in the pre-trained distributed representations as they significantly improved the performance of the machine learning tasks (Bengio et al., 2010). Further online research showed that Word2Vec, GloVe, and FastText are most the commonly used pre-trained word embedding models.

However, this raises the question: which word embedding model is better for the analysis? The choice of the existing models may depend on the available data and the type of the study. Some studies used the word embeddings as extra features in their supervised system (Bengio et al., 2010) and some compared the performances of existing word embedding models (Baroni et al., 2014; Naili et al., 2017). For instance, Naili et al (2017) developed a comparative analysis using Word2Vec, GloVe, and Latent Semantic Analysis in topic segmenting. They concluded the choice of the word embedding model depends on the language, but Word2Vec developed by Mikolov et al. (2013), was the best choice for their analysis in terms of vector representations. Moreover, some research papers were solely based on using pre-trained models and assessing the performances for their analyses (Baroni et al., 2014; Eisner et al., 2016; Sjökvist, 2018). This shows the various applications of word embedding models in the literature for various topics create favorable results, for instance in sentiment analysis or information retrieval.

3. Methods

The models that were selected in this research will be described and explained in this section. Distributed representations of question texts were obtained from two types of pre-trained word embedding models, Word2Vec and FastText. Combinations of these representations and other features were used in classification algorithms for supervised learning. These classification models were Naïve Bayes, Logistic Regression, K-Nearest Neighbors, and Random Forest.

3.1. Word2Vec

The Word2Vec model is designed to reduce the sparsity of the documents and achieves this by mathematically grouping the similar words within the vocabulary together in a vector space. This creates a type of neural network (Mikolov, Chen, et al., 2013). The output of this model is an array of numbers between positive and negative one. Calling words in this model gives an array of the words' vector representation as shown in Figure 1. The form of this model makes it easier for computers to process the information and recognize the patterns ('A Beginner's Guide to Word2Vec and Neural Word Embeddings', n.d.).

Moreover, a similarity score of the words can even be obtained using simple mathematics. For instance, a query can be created where a positive word is "woman" and a negative word is "man". What would be the negative words that have the closest probability to the positive word "king"? (see in the Figure 2 below).

```
:  1 word = model['boy']
   2 print(word[0:5])
   3 print(word.shape, word.dtype)

[ 0.23535156  0.16503906  0.09326172 -0.12890625  0.01599121]
(300,) float32
```

Figure 1. Word2Vec distributed representation of word "boy" in the form of 300-dimensional array of numbers between (1, -1)

```
1 model.most_similar(positive=['woman', 'king'], negative=['man'])
```

```
[('queen', 0.7118192911148071),
 ('monarch', 0.6189674139022827),
 ('princess', 0.5902431607246399),
 ('crown_prince', 0.5499460697174072),
 ('prince', 0.5377321243286133),
 ('kings', 0.5236844420433044),
 ('Queen_Consort', 0.5235945582389832),
 ('queens', 0.5181134343147278),
 ('sultan', 0.5098593235015869),
 ('monarchy', 0.5087411999702454)]
```

Figure 2. Word2Vec similarity function. Top 10 words with their given probability ratios for the positive words: “woman” and “king” and negative word: “man”.

Word2Vec uses either of two techniques, continuous bag-of-words or the skip-gram approach. The goal of the continuous bag-of-words (CBOW) approach is to predict the target word by using the context, and the skip-gram takes the word to create the context that the word may occur. Mikolov et al. (2013) gave an extensive mathematical explanation of how their three-layer architecture of input, hidden, and output layer work to supply both approaches in their study. Word2Vec offers two competent alternatives to the standard computation of word probability distributions. These components are hierarchical softmax, which estimates the overall probability distribution by using an output layer in a proportional log instead of the vocabulary size (Eisner et al., 2016) and negative sampling, which sets a boundary for the number of vectors that need to be updated. Therefore, only a sample of words are kept based on a noise distribution (Naili et al., 2017).

The Google Code Archive provides an available pre-trained Word2Vec model from part of the Google News dataset, which contains about 100 billion words in a 300-dimensional space (‘Word2vec’, n.d.). The use of this dataset gives an advantage as it was trained with a massive dataset built from billions of words. Therefore, the robustness of the model helps to capture deeper meaning of the words and reduces the time needed for researchers to build a similar model themselves. Some researchers also decided to train the pre-trained model once more with their own dataset, in order to capture the representations for the very uncommon words that were not included in the Google News dataset (Eisner et al., 2016; Naili et al., 2017).

3.2. FastText

FastText is an open-source library that was built by Facebook AI Research and is an extension of Word2Vec (Bojanowski, Grave, Joulin, & Mikolov, 2016). The methods utilize similar calculations as Word2Vec, likewise representing the words in the sentences with application of n-grams, bag-of-words, sub-word information, and sharing information across classes through hidden layer. The model also uses hierarchical softmax in order to deal the unbalanced distribution of the classes, which speeds up the

computation process. This has the advantage of efficient text classification and learning word vector representations (Mikolov, Chen, et al., 2013).

The main difference between the Word2Vec and FastText is that FastText embeddings consider the internal structure of the words while learning the representations. Therefore, while Word2Vec has limitations in rarely occurring words, this happens less often in FastText, and they can be captured faster (Jain, 2016). For instance, Sjökvist (2018) stated the disadvantages of Word2Vec, as the calculation did not include the morphological background of the word as FastText would such as in the following example words: 'walk', 'walking' or 'walked'.

Jain (2016) indicated that the results from FastText embeddings were significantly better than Word2Vec in encoding syntactic information tasks in his analysis, although, the latter worked better in the semantic tasks. Studies were often comparing the results of both word embedding applications, which also inspired this study. Moreover, this research compares both embedding models, which will be shown in the Results Section 5.

After obtaining the vector representations of the words, there are two common ways of calculating the sentence vector representation according to Mikolov et al. (2014). These are simply addition of each word's vector representation in one sentence or taking the average of each word vector representation in those sentences.

3.3. Classification models

The Scikit-learn library contains machine learning tools for data mining and data analysis, which attained for the analysis of this research. These classification algorithms from the library were also mentioned in the Related Work section. The following section will briefly discuss their advantages to use.

3.3.1. Naïve Bayes classifier.

Naive Bayes is a probabilistic model that is widely used for text classification, and it may be the simplest model. This algorithm considers each class using a distribution of different instances and assumes that the classes are independent of each other. Even though, in real-world applications this assumption of independence may be considered as 'false' or so called 'naïve', it can perform well, such as in spam filtering and sentiment analysis (Allahyari et al., 2017). Specifically, it produces sufficient results in small datasets and frequency-based probability estimation tasks. There are two types of Naïve Bayes classifications: Multi-variate Bernoulli and Multinomial Model. What both in common is that they use posterior probability. The multi-variate Bernoulli Model aims to find the presence and absence of the words using the representation of binary vectors. Also, it was found to be appropriate for this analysis. Bernoulli Model does not count the total frequency of the words occurring, where Multinomial Model does consider.

3.3.2 K-Nearest Neighbors (KNN) classifier.

The KNN model is a distance-based algorithm that learns each instance by memorization. The classifier groups the labels by proximity or likelihood of similarity according to various distance metrics. In the Scikit-learn library, the default distance metric for this classifier is Euclidean distance.

The KNN model generally performs well in rather smaller datasets and can be used for anomaly detection, semantic searches, and recommendation systems (Soni, 2018). Mikolov et al. (2013) also used this algorithm to build the word embedding models. Vector representations of similar words supposed to be close to be each other by their distances (Figure 2 above Word2Vec most similar function used this algorithm). Therefore, it was interesting to use this classifier to estimate the performance of the classifier in this dataset. The disadvantages of the classifier might be that, it is highly sensitive to the distances when setting a boundary between the data points, and large datasets could cause overfitting. These factors were also considered during the data transformation part in the analysis and the segmentation of categorical information was kept to as few as possible.

3.3.3 Logistic Regression classifier.

Logistic Regression is a predictive algorithm that is used to interpret the relationship between one (binary) dependent variable and one or more (nominal, ordinal, or interval) independent variable(s). The task of logistic regression is to apply the sigmoid function, which maps a S-shaped curve to fit the dataset to a value in between zero and one (Donges, 2018a). In the literature, Support vector machines (SVM) were found to achieve high performances in text mining (Shen et al., 2015). Logistic Regression and SVM could be comparable, following the similar methods of dividing the dataset with a linear kernel. The differences between the classifiers are defined as in the following paragraph. SVM try to maximize the margins between the support vectors, while Logistic Regression assesses the probability of the classes. SVM are less sensitive to the outliers than the Logistic Regression models. Moreover, Logistic Regression is a more common method and easy to apply. SVM algorithm's prediction complexity may increase with the number of dimensions which can cause long hours of training time. In general, Logistic Regression classifier often used as a benchmark method in machine learning tasks (Donges, 2018a).

3.3.4 Random Forest Classifier.

Random Forest algorithm can be considered as an ensemble method as it considers the blend of two methods, using multiple decision trees and bagging methods to assess the outcomes of classifications. Decision Trees classifier is considered as easy to interpret for the researchers and this can be applicable to the Random Forest classifier as well. The benefits of using the Random Forest classifier over Decision Trees are that it averages the outcomes of multiple trees and does not search for the most important feature from the whole dataset. Instead, the algorithm divides the dataset into subsets and looks for the most important features from each subset. Therefore, overall assessment of each subset gives more

information to the learning of the classifier, which can decrease the chances of overfitting (Donges, 2018b). Furthermore, the algorithm creates decorrelated trees when multiple features are correlated to each other, and reduces the variance. Accordingly, these properties make Random Forests widely applied methods for the text classification tasks (Donges, 2018b).

The Scikit learn library tool for this classifier, allow to users specify the max-features hyperparameter. This is the maximum number of features that the classifier can attempt in the individual tree. The common use of maximum features is usually the square root the total number of features (Foo, 2014).

3.3.5 Principal Component Analysis (PCA).

Text in nature creates high dimensional data. PCA is one of the earliest and most common algorithms that is used for dimensionality reduction and feature extraction (Pearson, 1901). The aim of the analysis is to find a new set of dimensions (features) by reshaping and spreading the data points orthogonally. These new features are ranked according to the variance of data points and could also be interpreted as the set of features that contribute the highest in terms of their informativeness (Kumar, 2018). The advantage of this model is that it creates linearly independent dimensions without losing any information from the data points (Kumar, 2018). In theory, after applying PCA to the dataset and obtaining new sets of features, it would help avoid overfitting by the classifiers. This may result in a better predictive performance in the supervised learning.

4. Experimental Setup

This section will give a detailed description of how this research is approached in order to answer the research question and explain the decisions were made to conduct the analysis. First the datasets will be described. Second, the steps: data pre-processing, feature selection, and feature transformation will be explained by giving the reasons. Third and last, some descriptive information will be provided to give insights about the datasets. Overall, this analysis was done using Python version 3.6 and the code could be found in the GitHub repository: https://github.com/zeyzeyy/thesis_guesswho.

4.1. Data Set Description

Two online experiments were conducted using a corpus of 196 photographs of people from head-to-shoulder, directly glancing at the camera with a natural expression (hereafter referred to as faces). Guess-Who and Rating task experiments were conducted online on Amazon Mechanical Turk. Guess-Who and Rating files were stored in JSON format and the first entries of both files were attached to Appendix A.

4.1.1. Rating task experiment.

In the first experiment, participants were shown images of faces and asked to write demographic, physical, and non-physical attributes. Each participant of the Rating experiment approximately completed five face descriptions.

4.1.2. Guess-Who experiment.

The second experiment was designed in the form of the Guess-Who game. The corpus of photographs was divided into six different boards, each containing a collection of 16 faces. There were a total of 200 participants in this experiment. Their task was to ask yes/no questions to identify a target face. As there was no actual target face, the computer randomly generated a “yes” or “no” answer after the participant submitted his or her question. Depending on the random answer given by the computer to the question, participants eliminated faces on the screen to find the target. On average, participants asked four questions per game (or board) to eliminate the faces, and each participant played approximately four different boards. Guess-Who dataset contained questions, answers, board numbers, all faces, and eliminated faces could be visualized in Appendix A, Figure A-1.

4.2. Dataset pre-processing

Our research aim was to predict the true answers for all questions that were asked in the Guess-Who dataset. The following part will explain the decisions that were made to reach objective of this research. There are two JSON files from each experiment, Rating dataset 1 and 2 and Guess-Who dataset 1 and 2. Rating dataset 1 was excluded from the analysis as it was missing the categorical variables that

were going to be used. Moreover, Guess-Who 1 and 2 dataset values were removed when "question" or "eliminatedFaces" contained empty values. Accordingly, Rating 2, Guess-Who 1 and Guess-Who 2 datasets were combined in one CSV file (Rating and Guess-Who datasets can be visualized in Appendix B, in the figures B-1 and B-2).

In the Related Work, Section 2 under description of the faces title, it was mentioned that segmentation of the face characteristics helped in identification and recognition. These segmentations could be by gender, ethnicity, age, hair, and eye color. Accordingly, these attributes about the faces were used as features from the Rating dataset.

FaceId in the Rating dataset was the unique identifier for each image of a face, which also matched with the values in the allFaces and eliminatedFaces columns in the Guess-Who dataset. Therefore, the following steps were taken to obtain the information of FaceId, its given physical descriptions, question, response and whether the face was eliminated or not, in one row on the CSV file. The allFaces column contained all the faces the participant saw on the computer screen, and the eliminatedFaces column showed the faces that were eliminated according to the question the participant made and the random answer was given by the computer (questionResponse column).

In order to merge these two datasets, the allFaces column of each row in Guess-Who dataset was split. A new dataset was created where each row contained a FaceId, a question where this FaceId was on screen, the random response to the question, whether this FaceId was eliminated after the response. Related to this FaceId from the Rating dataset the corresponding categorical details (ethnicity, eyes, gender, hair and age columns) were added. As mentioned earlier, the Rating dataset contained multiple descriptions of one FaceId and for the scope of this research only one random description was chosen for the further analysis. The merged dataset can be found in Appendix Figure B-3, in order to visualize how it was changed and was used for the further implications.

4.3. Data cleaning and descriptive information

This dataset resulted in nine columns and 306,304 rows. In the Rating experiment, gender and ethnicity information was collected through multiple choice (single answer) question format. For this reason, these variables were left the way they were as they did not require clean-up. There were 160,509 males, 142,080 females and 3,715 unsure gender categories found in this dataset. Also, there were three types of ethnicity values: White, African, and Latino. The descriptive information about these variables could be found in Table 1 below. Moreover, eyes, hair, age, and question columns were filled in a free-text format, and they were cleaned up before the further analysis according to following steps.

Table 1

Total counts of the ethnicity values across the genders

Ethnicity	Male	Female	Unsure
African	77,762	52,567	2,590
White	76,174	64,339	1,125
Latino	16,543	6,573	-
Total	170,479	123,479	3,715

4.3.1. Age column.

Exactly half of the age guesses of the Rating participants indicated that the faces showed people 25 years old or younger and 75% indicated that the faces showed people 29 years old or younger, where the mean age was 26.31, and the maximum given age was 61 years. As the participants of the Rating experiment practically guessed an age number for the people on the photographs, this information might not be reliable. However, this variable could still be useful to determine whether a person looked like an adolescent, an adult, or a senior. The observations made in this dataset showed a large number of questions asked in regard to whether a person is old or young. Including an age value might still be controversial and varied from person to person, however it was decided to be converted into age groups by using "Standardized Survey Classifications – Individuals" (n.d). The total counts of the age groups can be found in Table 2 below.

Table 2

Age groups and their count in overall dataset

Age Group	15-24	25-34	35-44	45-54	55-64
Count	137,070	145,123	18,858	4,153	1,100

Note. The age groups values were given in years, e.g. 15-24 years.

4.3.2. Hair and eye color columns.

First, upper-case characters and misspelled words were cleaned up for the hair and eye color columns. Variations of "blond" hair color, such as "strawberry blond" and "ash blond" etc., were converted into "blond" and the same principle was followed for "brown" values. "Dark brown" values were changed to "black", in order to have less variations within the categories.

For the eyes column, there was one instance of an outlier, "grey", which was changed to "blue", and the colors that stated "black" were changed to "dark brown", since black eye color does not exist.

4.3.3. Question column.

Standard processes of text cleaning methods were applied by removing the punctuation marks and replacing upper-case letters with lower-case. Moreover, Python NLTK library has a list of stop-words to remove from the texts as they might not be informative for some analyses. However, some of the words from this list thought to be informative for this analysis and was decided to be included instead. However, stop-words such as prepositions were still excluded. Ten samples of the cleaned-up questions are shown in Figure 3 below. Note that, after cleaning the question texts, some rows of this column became empty values and these rows were excluded from the further analysis.

```
1 data['question'].sample(n=10)
```

```
57236            wearing grey shirt under black shirt
118504                                           girl
156621                                        smiling
146905                              wearing undershirt
244311    second shirt visible underneath black shirt
18073                                wearing glasses
90277                             shirt say life impact
203699                                wearing necklace
154927                            wearing hair ponytail
51662                                            asian
Name: question, dtype: object
```

Figure 3. Cleaned questions in the merged dataset excluding the stop words.

4.4. Data transformation

This section describes the last steps that needed to be taken before finalizing the datasets for performing the analysis.

4.4.1. Obtaining target column.

Recall the research question, which pre-trained word embedding and classification models perform best in terms of accuracy rate in predicting answers to yes/no questions? At this stage, eliminated faces and the response from the computer columns were in the dataset. In order to perform the classification algorithms a target column was needed, which was called as true answer column. If the example question text “Is their skin white?” was taken from the Figure 3, and the response for this FaceId “IMG_0153” given by the computer was a “No”, then this FaceId should be eliminated. Accordingly, the true answer of this question would be a “Yes”. Hence, this column was created with the given criteria in Table 3 below.

Table 3

Criteria followed to obtain target column true answer

Response	Eliminated	True Answer
No	True	Yes
No	False	No
Yes	False	Yes
No	True	No

4.4.2. Word representations.

As it was mentioned in the Methods Section 3, Word2Vec and FastText pre-trained models were used in order to obtain the word representations of the question texts. The Word2Vec model could be found from the Python library Gensim, and the file for the pre-trained model could be obtained from: https://drive.google.com/file/d/0B7XkCwpI5KDYNlNUTTlSS21pQmM/edit. Also, the pre-trained FastText model could be accessed from: https://fasttext.cc/docs/en/english-vectors.html.

Le and Mikolov (2014) stated that these word representations could be simply summed or averaged in order to find a representation of a sentence or phrase. The same methods were applied for our cleaned-up question texts. Therefore, a function was created to look up the vector representations of the words in the pre-trained word embedding model, and returned a vector representation of the question column by either summing vectors for each word in the question text or averaging them.

Consequently, this process resulted in four different datasets. They all contained the face ID column, five columns of categorical features (age, gender, ethnicity, eye and hair color), 300 columns representing question text vectors instead of the question column, and the target column (true answer). These datasets were called Word2Vec addition, Word2Vec average, FastText addition, and FastText average. To visualize, the first five rows of Word2Vec addition dataset were attached as a Figure C-1 in the Appendix C.

4.4.3. Trainingand test datasets.

Last step in our experimental setup was to divide the datasets into trainingand test datasets. The purpose of this division was to evaluate the performance of the different classifiers for supervised learning tasks. The target column true answer has "yes" for 44% of the data and "no" for the rest. When creating the trainingand test split for the datasets, this process was randomized in a way that each dataset shared the same questions (as each question entry repeated for several FaceIds), and both trainingand test sets held the same ratio between the two values in true answer column. Further analysis was conducted by training the classifiers on the training set and evaluating the performance on the test set. The findings were further described in the following Results section.

5. Results

The aim of this research was to discover which pre-trained word embedding and classification models perform best in terms of accuracy rate in predicting answers to yes/no questions. In this section, the findings of this research will be shown by providing evaluation of the performances of the classification models. These findings will be described in five parts. Section 5.1. will show the baseline model to compare the accuracy rates with the other models. Section 5.2. will describe the Principal Component Analysis (PCA) presets. Section 5.3. will show the results of predictions from Word2Vec datasets. Section 5.4. will show the results of predictions from the FastText datasets. Section 5.5. will compare the overall findings.

5.1. Baseline Model

Since there was no previous study exactly similar to this research, a baseline model was set according to the chances of levels in the target variable. Word2Vec addition and average datasets contained 304,480 rows and corresponding to this the target variable true answer consisted of 169,568 instances of "no" and 134,912 instances of "yes". When the baseline ratio was estimated according to ZeroR method (Brownlee, 2014), the number calculated as 169,568/(169,568+134,912) and .557 was obtained. This calculation yielded exactly the same for FastText datasets. Therefore, 55.6% was determined as the baseline for the classification models that were trained over datasets.

5.2. Dimension reduction

All four of these datasets contained 308 columns and approximately 300,000 rows. For the feature selection and dimension reduction reasons, PCA was applied using the Scikit-learn library from Python. The hyperparameter of this algorithm was set such that 95% of the information would be kept in the features. Afterwards, the linearly transformed features were used and implemented in the classification algorithms again on the training sets.

5.3. Word2Vec Predictions Results

The results in this section were obtained from the analyses on Word2Vec addition and Word2Vec average datasets. Accordingly, the below findings will give accuracy rates from Naïve Bayes, Logistic Regression, K-Nearest Neighbors (KNN), and Random Forest classifiers in four sub sections. These classifiers were trained on the training set of the data, and predictions were made on the testing subset of this data. Therefore, the accuracy rates were calculated from the predictive performances of the classifiers on the test set. Comparably, the corresponding classification models' results were also provided and described after PCA on the datasets. These findings are presented in the Figure 4 below.

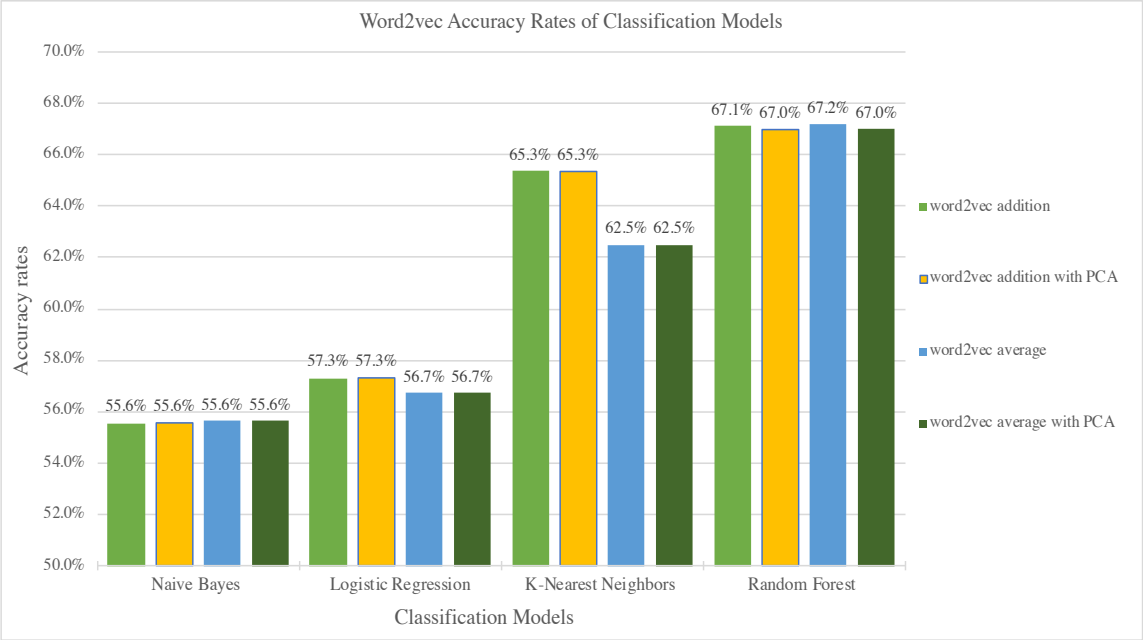

Figure 4. Performance evaluation of classification models (Naïve Bayes, Logistic Regression, KNN and Random Forest) by their accuracy rates on Word2vec addition and average test sets.

5.2.1. Naïve Bayes classifier.

As it could be seen from the Figure 4 above, Naïve Bayes classification had a 55.6% accuracy rate, similar to the baseline threshold, and it was the lowest performing classifier among all. There were no differences between the Word2Vec addition and average datasets nor after the application of PCA on the datasets. In the previous sections of this research, it was mentioned that Naïve Bayes considered the classes independently. For this reason, the algorithm achieved sufficient results in text mining when it involved word counting methods, such as TF-DIF methods in spam filtering. Despite the fact that the word representations actually had dependency between them. Assumptions of the independency might not be the case. This could explain the low predictive performances of Naïve Bayes classifier.

5.2.2. Logistic Regression classifier.

The second lowest classification performance occurred from the Logistic Regression classifier. It resulted in approximately 1% higher accuracy than the baseline model. The classifier generated better results in the Word2Vec addition test dataset to some small extent. The PCA effect on the prediction tasks were insignificant according to the outcomes. Corresponding to Method Section 3 of this paper, it was explained that Logistic Regression was a rather intuitive and simple model. It could be used for setting a baseline for prediction models, which these findings showed a similar case. The Logistic Regression model was decided to be used for the sake of comparing various classification models and also including

a prediction model that worked with a linear kernel to divide the data points. The accuracy rate was approximately 57% for both Word2Vec test sets. To conclude, this result might indicate that the datasets were too complex to be linearly separable.

5.2.3. K-Nearest Neighbors classifier.

Furthermore, the KNN algorithm provided an extensively better performance than the Naïve Bayes and Logistic Regression classification models. In order to achieve this performance on the test set, the *k* number was fixed to seven (accuracy rates on other parameters could be visualized in Figure 7). It could be seen that there is no difference between the Word2Vec addition and average test set, and corresponding PCA features of these datasets. Accuracy rate was yielded to 65.3% from Word2Vec addition test set, meaning a 17.3% increase compared to the baseline model. Next to this, a 62.3% accuracy rate was achieved from the Word2Vec average test set, which led to a 12.0% increase compared to the baseline model.

As it was mentioned in Section 4, the KNN algorithm learned by memorization and made predictions by calculating distances between the data points. Therefore, it could achieve better results from the distributed vector representations, since they were obtained by approximating related distributional distances between the words. Further, the developers of this model also used this algorithm in many of their studies (Le & Mikolov, 2014; Mikolov et al., 2013). This could explain the higher predictive performances of the KNN algorithm compared to the other classifiers. Moreover, averaging the word representations might cause the loss of some of the important dependencies in terms of distances. As KNN is sensitive to this, it could be visualized in the drop of Word2Vec average results in Figure 4 above.

5.2.4. Random Forest classifier.

Among all the classification models, Random Forest produced the best predictions with a 67.2% accuracy rate in the Word2Vec average test set. This means an increase of 20.6% from the baseline model. Moreover, there was very little decrease (.1%), on the PCA datasets, which could not be a significant difference. The outcome of this model was aligned with the other results of Random Forest text classification tasks in the literature, and the decisions of selecting this model. Recalling from the algorithm's properties mentioned in Method Section 3, the model avoids overfitting and instead, searches for the most important features by several decision trees.

In the Scikit-Learn library, Random Forest algorithm's maximum number of features could be set to a value and the common practice of this parameter showed to set to the square root of the total number of features (Foo, 2014) and this was set to 17 in the analysis. Furthermore, this library provided an interesting tool for Random Forest algorithm, which aided to visualize the associated weight of significance for each feature on the prediction model. This measure was visualized in the Figure 5 below.

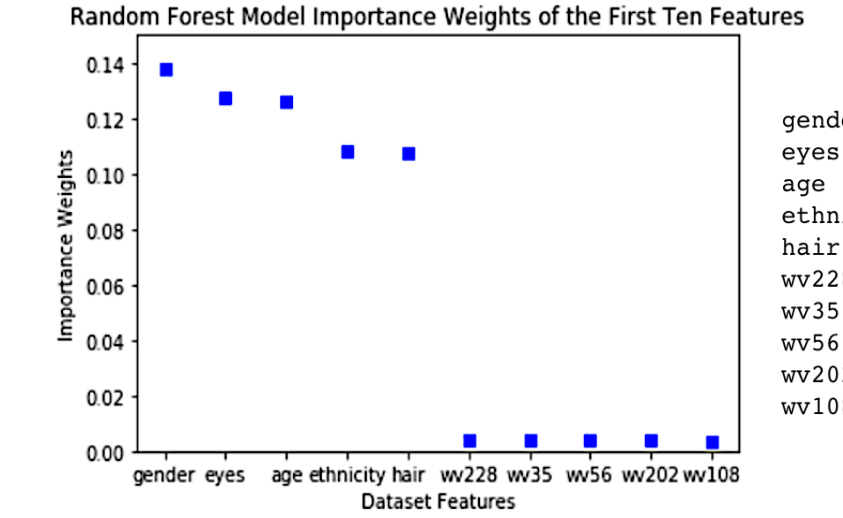

Figure 5. Random Forest model feature importance weights of the first ten features from word2vec addition dataset and corresponding exact weights. *wv* columns here are the vector representations of the questions.

It could be seen from Figure 5 that gender, ethnicity, age, eyes, and hair color features were the ones that mostly contributed to the classification model by between 10-13% each. They could be considered as the most informative variables as they gave descriptive information about the people. Furthermore, the word representation numbers were less important for the Random Forest model as they contributed .4% and below to the overall performance of the algorithm.

5.3. FastText Predictions Results

In this section the results will be presented in a similar way as was done in the previous Word2Vec prediction results. Therefore, four subsections will show the classification models trained on the FastText training set and evaluation of the performances on the test set will be assessed by the accuracy rates. Briefly, the differences in the results between these datasets and corresponding PCA features datasets will be compared. In the Figure 6, the overview of the classification accuracy rates was presented.

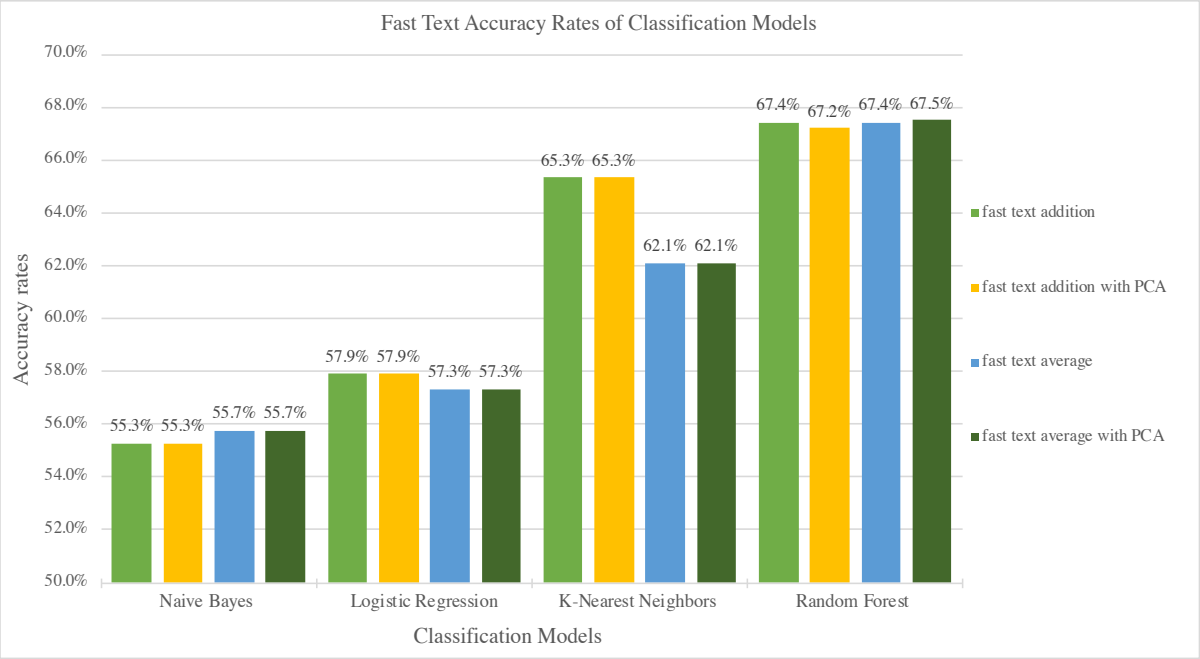

Figure 6. Performance evaluation of classification models (Naïve Bayes, Logistic Regression, KNN and Random Forest) by their accuracy rates on FastText addition and average test sets.

5.3.1. Naïve Bayes classifier.

As shown in Figure 6, the accuracy rates were generally aligned with the Word2Vec results. The Naïve Bayes classifier performed the lowest. The FastText average datasets accuracy rates yielded marginally higher results, with .4% difference in predictions compared to the FastText addition datasets, and almost at the baseline. This could be considered that the classifier was insignificant for prediction model of these datasets. There were no significant differences after PCA was applied on the datasets.

5.3.2 Logistic Regression classifier.

The Logistic Regression classifier also yielded almost similar results as the Word2Vec datasets. However, the FastText addition accuracy rate was .6% higher than the Word2Vec addition accuracy rate. Comparably, it could be observed that the performance on the addition datasets were slightly higher than the average datasets. The Logistic Regression classifier could be considered at the baseline level with its slightly higher percentage of accuracy on the test set. Hence, Logistic Regression could not be considered as a sufficient classifier for this task but rather could be the baseline of our models.

5.3.3. K-Nearest Neighbors classifier.

The KNN results aligned with the outcome of the previous Word2Vec findings. Again, the classifier achieved a better accuracy rate on the FastText addition test set than the FastText average test

set. Furthermore, when the *k* parameter was set to seven (the same as the Word2Vec settings), the algorithm yielded an accuracy rate of 65.3% on FastText addition dataset, with an increase of 17.3% from the baseline model. Again FastText average datasets showed lower performances in the classification with 62.5% accuracy rate. The PCA did not affect the performance of the classifiers. In order to assess the performances of other *k* values, we ran the classifier with the range of *k* values between three and nine. This produced the graph represented in the Figure 7 below.

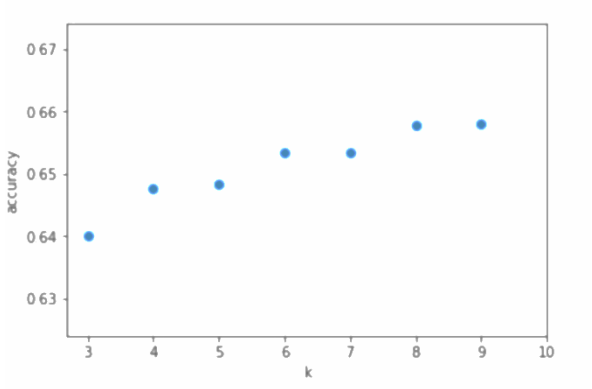

Figure 7. KNN accuracy rates achieved from the performance on the FastText addition test set, represented on the horizontal axis and the values of *k* in range of (3, 9) on the vertical axis.

The best result of the KNN algorithm was achieved when the *k* was set to nine, and the accuracy rate reached to 66.0%. However, this did not show a significant difference in the results than the previous preset of $k = 7$. In conclusion, the KNN algorithm was acceptable in terms of the baseline threshold.

5.3.4. Random Forest classifier.

Performance of the Random Forest classifier did not change among addition, average, and PCA features datasets, and the results were similar to those of the Word2Vec datasets. The same presets were applied by assigning 17 to the maximum number of features parameter of this algorithm. Moreover, the classifier achieved the best classification performance in overall FastText test datasets. The approximate accuracy rate was 67.4%, and it led to a 21.1% increase from the baseline model.

5.3 Overall Findings of the Classification Models

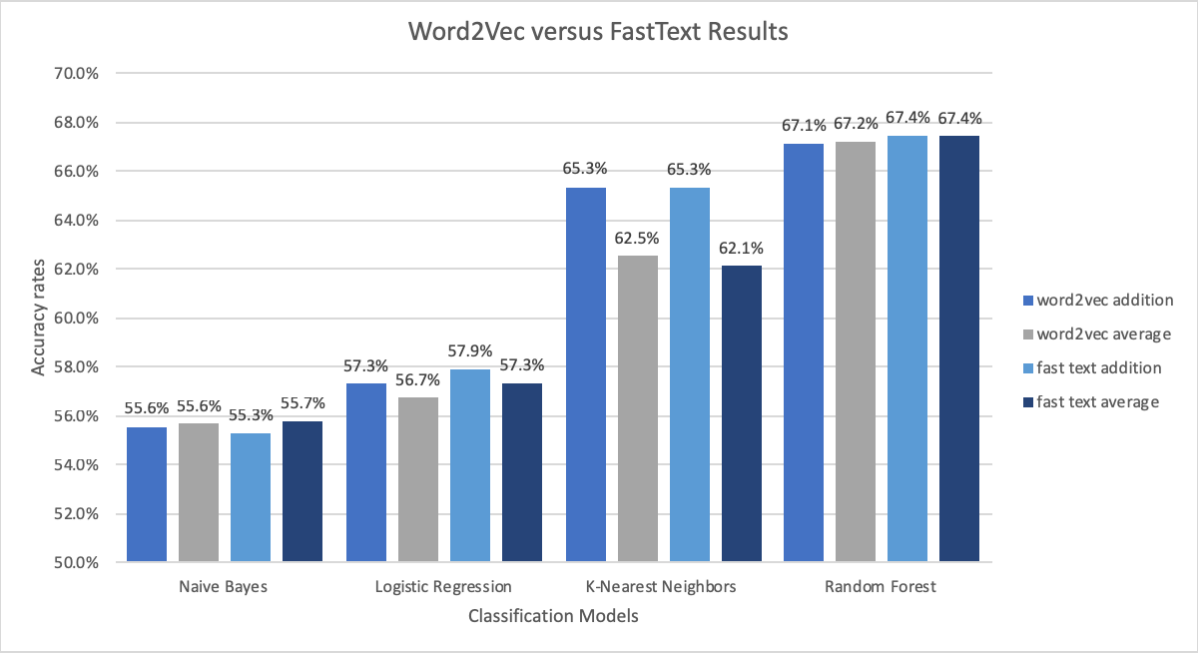

Figure 8. Overall findings of the accuracy rates of classification models among the Word2Vec and FastText datasets, excluding the PCA results

In summary, classification models were applied to Word2Vec and FastText datasets by training on the training set and performing predictions on the test set. Overall, accuracy levels were similar between these two types of word representations. It could be concluded that dimensionality reduction through application of PCA algorithm did not change the overall results on the predictions. Therefore, PCA results were not included on the final comparison in the Figure 8 above. Additionally, the only difference between the average and addition datasets was for the KNN algorithm. Even then the difference can be still be considered small, with a percentage of approximately 3%.

Naive Bayes had the lowest performance among all the classifiers and did not perform higher than the baseline level. Logistic regression accuracy rates were to those of the baseline model, which was also the case in the mentioned previous literature studies. The KNN algorithm performed significantly higher than the baseline model, both on the Word2Vec and FastText addition datasets. The best predictive performance was achieved from the Random Forest classifier. Despite the fact that the results were approximately equal among all the datasets for this classifier, it could be said that the FastText datasets yielded slightly higher accuracy rates in predictions.

6. Discussion

This section will discuss the results in the previous section and evaluate the findings of this research. The goal of this study was to determine which pre-trained word embedding and classification models perform best in terms of accuracy rate in predicting answers to yes/no questions. This study was constructed in the form of a comparative design. The findings from the four datasets, Word2Vec addition, Word2Vec average, FastText addition, and FastText average, will be discussed in the following section.

Previous studies found Word2Vec, FastText, and GloVe were prominent and powerful word embedding models. Various research conducted comparative studies or evaluated these word representations through supervised or unsupervised learning models. Elaborating more on this, Naili et al. (2017) chose to use GloVe, Word2Vec, and Latent Semantic Analysis word embedding models in topic segmentation, and they concluded that the choice of word embedding models could vary according to the language and type of the analysis. However, Word2Vec showed highest performances in terms of vector representations of the words. Some found Word2Vec was more intuitive and simple to use (Mikolov, Chen, et al., 2013), as Word2Vec was also easy to call through the Python library.

FastText and Word2Vec were developed using similar mathematical algorithms. Jain (2017) stated that FastText could be more efficient and faster in learning the words. Another advantage was that the model tried to learn more of the morphological background of the words. Consequently, Word2Vec and FastText word embedding models were used for this reseach. In order to have some variety in the word representations, two different corpuses were chosen for the pre-trained models. Further, the classification models were chosen and applied according to the article from Allahyari et al. (2017) and preliminary web research. Logistic regression, Naïve Bayes, K-Nearest Neighbors (KNN), and Random Forest classifiers were chosen as the prediction algorithms and evaluation of the performances were given by the accuracy rates.

Overall, the findings showed that the predictive performances of our classification models were aligned at a high level between the two word embedding models. The best predictive performance was achieved from the Random Forest algorithm with a 67.4% accuracy rate, and even though the difference was very small FastText datasets had the highest accuracy rates. The baseline model was 55.6%, so certainly, the Random Forest algorithm improved upon the baseline model by 21.1%. In the literature, it was mentioned that this algorithm achieved high predictive performances as it used ensemble methods and avoided overfitting. The dataset was very large with approximately 300,000 rows and 307 columns, and there was a high chance of overfitting. The randomness aspect, selection of features from several subsets of Decision Trees, and assessing the contribution of each features from the subsets by bagging methods, made this model avoid overfitting and achieve overall high classification performances, which also showed in the results of the research.

KNN performed second best with a 65.3% accuracy rate in FastText and Word2Vec addition datasets. The KNN algorithm learned the distances between the data points. It was noticed that as the number *k* got higher, the algorithm performed better. When *k* was equal to 9, it became stable (this could also be visualized in Figure 7). The algorithm performed well, especially when adding vector representations, as opposed to averaging them. The nature of the vector representations of the words were connected to each other with their distance weights. For instance, Mikolov, Sutskever, et al. (2013) also used this algorithm in their model for observing the similarities between words. Therefore, high predictive performances achieved from KNN algorithm were not surprising.

On the contrary, Naïve Bayes had different properties than the Random Forest and KNN algorithms. The assumption of the independent variables property did not work well with the word embedding models, since the vector representations of the words were contingent on one another. Therefore, Naïve Bayes was found to be too simplistic of a model for this type of analysis and performed the worst among all models.

In a similar way, Logistic Regression matched the accuracy rate of the baseline level in all of the datasets. This could indicate that the classes were not linearly separable. Moreover, one of the reasons for including Logistic Regression in the analysis was, that it works in a way related to Support Vector Machines (SVM), as this algorithm drew a kernel between the data points to separate the classes. In theory, if it was observed that the data was not linearly separable with Logistic Regression, then SVM with a non-linear kernel could be run, since SVM would try to separate the data points by its classes by creating a wider margin. However, SVM could not be used for this analysis, due to the algorithm's complexity, thus, the run time increased exponentially with more features. After application of PCA, this algorithm was again run, but it still did not decrease the run time, although application of SVM was highly recommended in the literature when dealing with text data with high dimensionality (Zheng & Lu Bao-Liang, 2011)

Generally, the limitations that were encountered in this research were loading the pre-trained word embedding models, training, and fitting the classification algorithms in the full dataset. Furthermore, one of the-state-of-art methods such as Elmo, or another word embedding model, GloVe by Stanford NLP Group, were considered for this comparative analysis; however, because of limitations in time and length of this thesis, they were excluded.

In conclusion, this research was devoted to implement a comparative analysis of the pre-trained Word2Vec and FastText vector representations and evaluated these word embedding models with the trained supervised learning algorithms. This study used models that were proven to be more accurate and faster than the traditional word embedding algorithms. Particularly, using pre-trained word representations and the categorical features of gender, ethnicity, age group, hair, and eye color achieved up to a 67.4% in accuracy rate of predicting the yes/no answers to the questions that were asked in the

Guess-Who dataset. This led to 21.1% increase above the performance from the baseline model. In the literature, these word embedding models were used for various natural language processing applications such as information retrieval, speech recognition, or question answering. However, there was a gap found in the literature that these pre-trained models were never used for question answering about human facial characteristics. It was very hard to achieve a large dataset solely made for this type of analysis and train the word embedding models. Therefore, this study could contribute to the field of supervised learning by showing that sufficient classification performances could also be achieved through pre-trained word embedding models and very little categorical information.

7. Conclusion

This section will repeat the research question, summarize the findings, and describe the possible future implications to conclude this study.

The research question is "Which pre-trained word embedding and classification models perform best in terms of accuracy rate in predicting answers to yes/no questions?" These questions were taken from an experiment called Guess-Who where participants tried to identify a target face by asking questions to a computer and eliminating faces depending on the description in their questions and the computer's random responses. Hence, the goal was to predict the yes/no answers to these questions using the information that was given about the faces.

Therefore, this research was designed as a comparative analysis. In order to obtain the vector representations of the question texts, two types of pre-trained word embedding models (Word2Vec and FastText) were utilized. For obtaining vector representations of questions, summing or averaging vectoral representations of the words were used as two different methods. Therefore, four different datasets were obtained for further analysis: Word2Vec addition, Word2Vec average, FastText addition, and FastText average.

These datasets contained 300 columns of vector representations, five categorical variables, and one target variable. A total of four classification algorithms were used to predict the target value (true answer), which were Naïve Bayes, Logistic Regression, K-Nearest Neighbors, and Random Forest. The classification algorithms were trained with 305 features on the training set to predict the target variable on the test set, and the accuracy rate was used for the evaluation of the models. The results showed that the classification algorithms performed similarly for Word2Vec and FastText datasets. The highest accuracy rate was reached by the Random Forest algorithm with 67.4% in FastText average datasets, following that, the K-Nearest Neighbors algorithm reached 65.3% in FastText addition dataset. Moreover, Random Forest's most important feature assessment helped to visualize that the categorical features add the highest information to the performance by 10-14% (see Figure 5). Therefore, this concluded that the segmentation of the faces was very informative and helped the most in the predictive algorithms. On the other hand, Principal Component Analysis did not affect the results of text classifications, as the findings noted there was no significant change in the classifiers' performances.

Word embedding models had struck the interests of the researchers, and it was becoming prominent for commercial use as it could be utilized in many areas. The comparison and application of these models were highly beneficial within text mining and machine learning fields. For further research, this comparative analysis may also include one of the state-of-art methods, such as Elmo word embedding model by Allen Group. GloVe has also proven to perform well and been used in the literature for comparative analyses, and therefore could be added to obtain differentiating results between the word embedding models. Additionally, the Guess-Who and Rating datasets are unique and contain a variety of

information, for instance, the descriptive texts of the non-physical information about the faces. These can also be informative in future analyses. Again, this information can be used with the pre-trained models as additional features; however, this would add even more natural language processing methods in the study. Besides this, the segmentation of the faces contributed the most in the Random Forest classification model. Therefore, it would be interesting to use the photographs of the faces in deep learning models to obtain even more information from the faces. Then the analysis can even be conducted with a similar research design, using the deep learning feature representations of the photographs and pre-trained text representations of the questions as features in order to predict the answers to yes/no questions in the Guess-Who dataset. This would finally highlight whether by gaining more characteristics about the faces, the classification algorithms would perform at a higher rate in terms of accuracy.

Acknowledgements

This research was supervised by Dr. Andrew T. Hendrickson and research material, Guess-Who and Rating datasets were obtained from him for the purpose of this thesis research.

Furthermore, there are large amount of academic resources about word embedding models, which would lead to beyond the scope of a master's research thesis. Online sources such as the blogs; Towards Data Science, Machine Learning Blog, Analytics Vidhya and Stack Overflow helped to understand the general concepts, gain knowledge and apply algorithms in machine learning field.

References

A Beginner's Guide to Word2Vec and Neural Word Embeddings. (n.d.). Retrieved 2018, November 21, from https://skymind.ai/wiki/word2vec

Aggarwal, C. C., & Zhai, C. X. (2013). *Mining text data*. *Mining Text Data* (Vol. 9781461432). https://doi.org/10.1007/978-1-4614-3223-4

Allahyari, M., Pouriyeh, S., Assefi, M., Safaei, S., Trippe, E. D., Gutierrez, J. B., & Kochut, K. (2017). A Brief Survey of Text Mining: Classification, Clustering and Extraction Techniques. Retrieved from http://arxiv.org/abs/1707.02919

Baroni, M., Dinu, G., & Kruszewski, G. (2014). Don't count, predict! A systematic comparison of context-counting vs. context-predicting semantic vectors. *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 238–247. https://doi.org/10.3115/v1/P14-1023

Bengio, Y., Courville, A., & Vincent, P. (2014). Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *35*(8), 1798–1828. https://doi.org/10.1109/TPAMI.2013.50

Bengio, Y., Ratinov, L., & Turian, J. (2010). Word representations: a simple and general method for semi-supervised learning. *In Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, (Published by Association for Computational Linguistics), 384–394. Retrieved from http://www.iro.umontreal.ca/~lisa/pointeurs/turian-wordrepresentations-acl10.pdf

Bojanowski, P., Grave, E., Joulin, A., & Mikolov, T. (2016). fastText. Retrieved December 20, 2018, from https://research.fb.com/fasttext/

Brownlee, J. (2014, Nov 5). *How To Get Baseline Results And Why They Matter. Machinert Learning Mastery*. [Blog Post] Retrieved from https://machinelearningmastery.com/how-to-get-baseline-results-and-why-they-matter/

Datta, R., Joshi, D., Li, J., & Wang, J. Z. (2008). Image retrieval. *ACM Computing Surveys*, *40*(2), 1–60. https://doi.org/10.1145/1348246.1348248

Donges, N. (2018a, Apr 23). *The Logistic Regression Algorithm*. [Blog post] Retrieved from https://machinelearning-blog.com/2018/04/23/logistic-regression-101/

Donges, N. (2018b, Feb 22). *Random Forest Algorithm*. [Blog post] Retrieved from https://towardsdatascience.com/the-random-forest-algorithm-d457d499ffcd

Eisner, B., Rocktäschel, T., Augenstein, I., Bošnjak, M., & Riedel, S. (2016). emoji2vec: Learning Emoji Representations from their Description, 48–54. https://doi.org/10.18653/v1/W16-6208

English word vectors. (n.d.) Retrieved from https://fasttext.cc/docs/en/english-vectors.html

Foo, F. [Fred F.].(2014, May 30). Re: Understanding max_features parameter in RandomForestRegressor. [Blog comment]. Retrieved from https://stackoverflow.com/questions/23939750/understanding-max-features-parameter-in-randomforestregressor

Gatt, A., Tanti, M., Muscat, A., Paggio, P., Farrugia, R. A., Borg, C., … van der Plas, L. (2018). Face2Text: Collecting an Annotated Image Description Corpus for the Generation of Rich Face Descriptions, 3323–3328. https://doi.org/10.1002/j.2164-4683.1997.tb00371.x

Harari, Y. N. (2015). *Sapiens: a brief history of humankind* (pp. 24-25). New York: Harper,

Hodosh, M., Young, P., & Hockenmaier, J. (2015). Framing image description as a ranking task: Data, models and evaluation metrics. In *IJCAI International Joint Conference on Artificial Intelligence* (Vol. 2015–Janua, pp. 4188–4192). https://doi.org/10.1613/jair.3994

Jain, J. (2016). FastText and Gensim word embeddings. Retrieved December 20, 2018, from https://rare-technologies.com/fasttext-and-gensim-word-embeddings/

Karani, D. (2018, Sep 1). *Introduction to Word Embedding and Word2Vec*. [Blog post] Towards Data Science. Retrieved from https://towardsdatascience.com/introduction-to-word-embedding-and-word2vec-652d0c2060fa [Accessed 4 Dec. 2018].

Le, Q. V., & Mikolov, T. (2014). Distributed Representations of Sentences and Documents, *32*. https://doi.org/10.1145/2740908.2742760

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient Estimation of Word Representations in Vector Space, 1–12. https://doi.org/10.1162/153244303322533223

Mikolov, T., Sutskever, I., Chen, K., Corrado, G., & Dean, J. (2013). Distributed Representations of Words and Phrases and their Compositionality, 1–9. https://doi.org/10.1162/jmlr.2003.3.4-5.951

Naili, M., Chaibi, A. H., & Ben Ghezala, H. H. (2017). Comparative study of word embedding methods in topic segmentation. *Procedia Computer Science*, *112*, 340–349. https://doi.org/10.1016/j.procs.2017.08.009

NSS. (2018, Mar 22). *Intuitive Understanding of Word Embeddings: Count Vectors to Word2Vec*. [Blog post]. Retrieved from https://www.analyticsvidhya.com/blog/2017/06/word-embeddings-count-word2veec/.

NLPL word embeddings repository. (n.d.). Retrieved December 1, 2018, from http://vectors.nlpl.eu/repository/

Pearson, K. (1901). On Lines and Planes of Closest Fit to Systems of Points in Space. Philosophical Magazine. 2 (11): 559–572. doi:10.1080/14786440109462720.

Shen, Y., Rong, W., Jiang, N., Peng, B., Tang, J., & Xiong, Z. (2015). Word Embedding based Correlation Model for Question/Answer Matching, 3511–3517. Retrieved from http://arxiv.org/abs/1511.04646

Sjökvist, H. (2018). *Text Feature Mining Using Pre-trained Word Embeddings*. (Master's Thesis) Retrived from: https://www.math.kth.se/matstat/seminarier/reports/M-exjobb18/180601a

Soni, D. (2018, March 12). *Introduction to k-Nearest-Neighbors*. [Blog post] Retrieved from https://towardsdatascience.com/introduction-to-k-nearest-neighbors-3b534bb11d26

Standardized Survey Classifications – Individuals (n.d). Retrieved from http://www.pgagroup.com/standardized-survey-classifications.html

Word2Vec. (n.d.). Retrieved from https://code.google.com/archive/p/word2vec/

Zhao, W., Chellappa, R., Phillips, P. J., & Rosenfeld, A. (2003). Face Recognition: A Literature Survey. In *ACM Computing Surveys* (Vol. 35, pp. 399–458). https://doi.org/10.1145/954339.954342

Zheng, J., & Lu Bao-Liang, B. L. (2011). A support vector machine classifier with automatic confidence and its application to gender classification. *Neurocomputing*, *74*(11), 1926–1935. https://doi.org/10.1016/j.neucom.2010.07.032

Appendices and Supplementary Materials

Appendix A


```
uniqueID : 6746
completion_code : Vi3CG-#-XK811
demographics :
   country : United States of America
   gender : male
   ethnicity-details :
   ethnicity : white
   age : 32
experiment : guess-who-version2
boardNumber : 4
condition : male_only
actionType : questionAsked
question : Is this person white?
questionResponse : No
finalSelection : NA
questionNumber : 3
totalFaceCount : 16
facesEliminatedCount : 8
eliminatedFaces : ['IMG_9258', 'IMG_9957', 'IMG_9710', 'IMG_9048', 'IMG_9876', 'IMG_7882', 'I
MG_7806', 'IMG_8607']
allFaces : ['IMG_8048', 'IMG_9258', 'IMG_0701', 'IMG_8622', 'IMG_9957', 'IMG_9710', 'IMG_955
9', 'IMG_9048', 'IMG_0641', 'IMG_9243', 'IMG_9454', 'IMG_9876', 'IMG_7882', 'IMG_7806', 'IMG_
8607', 'IMG_7623']
currentFacesEliminatedCount : 0
```


Figure A-1. Guess-Who JSON file first entry


```
experimentCode : guess-who-rating-task-v2
completion_code : FMii2-#-vNw4F
uniqueID : 85762
demographics :
   country : United States of America
   gender : female
   ethnicity-details :
   ethnicity : white
   age : 36
state : finished-instructions
trialNumber : 2
faceID : IMG_8622
responses :
   non-physical-description : Is intellectual  Does a lot of reading  Not into sports accept
maybe soccer  Maybe a hipster
   physical-description : Has beautiful eyebrows and a nice hair cut. His complexion is very
clean and young looking. His ears are big but who cares[comma] although people have probably
made fun of him[comma] unfortunately.
   occupation : Student or starbucks barista
   typical : neutral
   attractive : agree
   photo-gender : male
   ethnicity-details : indian
   ethnicity : african
   eye : brown
   hair : black
   age : 20
```


Figure A-2. Rating JSON file first entry

Appendix B

```
2 temp.head()
```

	allFaces	eliminatedFaces	question	questionResponse
0	IMG_9891,IMG_0153,IMG_0446,IMG_9484,IMG_9846,I...	IMG_9891,IMG_0153,IMG_9846,IMG_8502,IMG_8198,I...	Is the person's hair long?	No
1	IMG_9078,IMG_8697,IMG_9574,IMG_7715,IMG_9906,I...	IMG_9078,IMG_8697,IMG_9574,IMG_7715,IMG_9906,I...	does the person have a ponytail?	No
2	IMG_7942,IMG_7806,IMG_9710,IMG_9048,IMG_0701,I...	IMG_7942,IMG_7806,IMG_9710,IMG_9048,IMG_0701,I...	am I able to see the word 'new' on his shirt f...	Yes
3	IMG_7517,IMG_8153,IMG_9861,IMG_7836,IMG_7988,I...	IMG_7517,IMG_8153,IMG_9861,IMG_7836,IMG_7988,I...	does her hair go past her shoulder	No
4	IMG_9755,IMG_0401,IMG_9927,IMG_0325,IMG_7851,I...	IMG_0401,IMG_9048,IMG_0641,IMG_0566,IMG_7806,I...	is the person male?	No

Figure B-1. Combination of two Guess-Who datasets and representation of first five rows

```
1 df_rating.head()
```

	age	ethnicity	eye	faceID	gender	hair
0	20	african	brown	IMG_8622	male	black
1	24	white	brown	IMG_7942	male	black
2	25	african	black	IMG_0401	male	black
3	23	white	Blue	IMG_0018	male	Blonde
4	26	white	brown	IMG_0777	male	brown

Figure B-2. Rating dataset first five rows

```
1 data.sample(n=5)
```

	faceId	question	response	eliminated?	ethnicity	eyes	gender	hair	age
75385	IMG_0401	Does this person have very...	Yes	True	african	black	male	black	25
184945	IMG_0153	Is their skin white?	No	True	white	brown	female	brown	26
303516	IMG_8517	Does the person have on a ...	No	False	african	brown	male	black	20
17831	IMG_0295	does he have a beard	Yes	True	african	dark brown	female	black	22
203608	IMG_9153	Does the person have glasses?	Yes	True	white	green	female	brown	23

Figure B-3. Guess-Who and Rating datasets combined with response and eliminated faces columns

Appendix C

	hair	eyes	age	true_answer	FaceID	question	ethnicity	gender	wv1	wv2	...	wv291	wv292
199937	black	brown	15-24	no	IMG_8622	boy	african	male	0.235352	0.165039	...	0.019043	-0.010315
52702	black	dark brown	25-34	no	IMG_9469	wear glasses	african	female	-0.215332	-0.088928	...	0.188568	0.051758
126445	black	dark brown	15-24	no	IMG_8243	have ponytail	white	female	-0.070034	-0.009277	...	-0.033264	0.132812
177155	brown	dark brown	15-24	no	IMG_0746	scarf have flowers	african	female	-0.026001	0.043294	...	-0.128662	-0.023234
189931	black	brown	15-24	yes	IMG_9574	hair bun	african	female	-0.241211	0.103882	...	-0.011230	0.169434

Figure C-1. Word2Vec addition dataset with all the features and after the target variable obtained