Predict The Effect of Genetic Variants to Enable Personalized Medicine

|  |  |
| --- | --- |
| Rachel Balon  *Wayne State University*  *Department of Industrial and Systems Engineering* rachelcbalon@gmail.com | Rajatsubhra Chakraborty  *Wayne State University*  *Department of Computer Science*  hj0220@wayne.edu |

*Abstract—*A cancer tumor is made up of thousands of genetic changes that have occurred throughout time. In spite of technological advancements, the work of separating genetic variants that operate as a driver for the formation of tumors from genetic alterations that act as passengers (Neutral Genetic Mutations) is still done manually, even today. This is a time-consuming process in which pathologists manually interpret each genetic mutation based on the clinical evidence they receive. Despite the fact that these clinical shards of evidence are classified into nine categories, the criterion for classification is yet uncertain. With Natural Language Processing (NLP) approaches, the primary goal of this research is to propose a multiclass classifier for classifying genetic mutations based on clinical data (i.e., the text description of these genetic variants). Memorial Sloan Kettering Cancer Center (MSKCC) provided the data for this study, which was obtained via Kaggle and used in this investigation. The dataset is made possible by the contributions of world-class researchers and cancer specialists. Text is converted to a matrix of token counts using two text transformation models, namely, TfidfVectorizer, and Word2Vectorizer, each of which has its own set of parameters. Logistic regression, SVM, KNN, Naive Bayes machine learning classification to classify text descriptions using a sparse matrix (keywords count representation). In this study, the confusion matrix is used to evaluate the accuracy score of all of the proposed classifiers. Last but not least, the empirical results demonstrate that the logistic regression all other proposed classifiers, achieving the greatest test log-loss of 0.951.

*Keywords: NLP, cancer, genetic mutation, machine learning, log-loss.*

# Introduction

Personalized medicine is a tremendous opportunity to personalize diagnostics, pharmacological therapy, and prevention. We are all alike, but also unique. The thought that medicine would ignore those variations is as absurd as going to a shoe store and buying any old pair of shoes without examining the size. It allows us to make unique predictions about illness risk that can help someone adopt a prevention plan that is suited for them. It also enables for more personalized pharmacological therapy rather than a "one size fits all" approach. The individual is becoming increasingly important in medicine, and many of us are having our genomes analyzed and stored in our medical records to enable this individualized approach. Lots to do, but this might be the largest medical breakthrough in decades.

However, even though cancer incidence and prevalence are rising at an alarming rate, progress in cancer treatment has been modest, and therapeutic benefits are assessed in weeks to months rather than years or decades. The traditional approach to patient care is for doctors to provide therapy based on the results of a pathological examination, symptoms of disease, and a patient's medication history. Several cancer forms can now be recognized before clinical symptoms appear as a result of breakthroughs in diagnostic technology and early detection biomarkers. Biochemical, epigenetic, genetic, imaging, metabolomic, and proteomic indicators are some of the types of markers that can be used to diagnose cancer. It is possible to identify these markers in clinical samples using technologies that can be multiplexed. Use of many markers in the same sample often boosts the sensitivity and specificity of cancer detection and aids the physician in making an early and accurate diagnosis of the condition. This information is extremely important because, as a result of the profiles of indicators outlined above, unique individual treatment regimens can be devised based on the existence and stage of cancer. However, molecular diagnosis with extra information may be different from pathological diagnosis in clinical practice. Pathological diagnosis is still the gold standard in clinical practice.

Cancer can be caused by genetic abnormalities that are either somatic or inherited. Following cancer genetics will help you understand hereditary malignancies, which constitute a significant portion of medical genetics. Family history accounts for only 10–15 percent of all malignancies, with the remainder cancers impacted by environmental factors, infections, and way of life. This information assists scientists in determining the likelihood of a person developing cancer over his or her lifetime. The presence of an allele that segregates in an autosomal-dominant form in a cancer-predisposing condition is rare, and the presence of such a syndrome is associated with an elevated risk of cancer development. Furthermore, non-genetic variables have a role in the occurrence of mutations or other genetic alterations. The development of cancer has also been reported in individuals who do not have a family history of the disease.

If the classification of gene mutations can be accomplished by textual evidence, it will aid in the discovery of cancer tumors in a more efficient and timely manner when compared to the manual technique taken by pathologists. The text evidence in this case has been processed utilizing NLP techniques, which is a novel notion in this field. In addition, the use of machine learning algorithms for classification has been added into the program. Logistic Regression classifier, K-NN classifier, Naive Bayes classifier, and Support Vector Machine classifier are the machine learning classification techniques used in this study.

# Literature Review

Cancer is a lethal disease that, if not identified early enough, can be highly painful and ultimately result in death. Each year, many people die of cancer, and the majority of cases are detected at a late stage. It is consequently critical to expedite cancer tumor detection procedures in order to preserve lives. Cancer is produced by gene mutations, which result in a disastrous pattern. Numerous machine and deep learning models are used and verified to efficiently classify gene mutations. The following is a list of some of the world's most recent studies on the subject.

Sondka et al. focused on establishing the qualities that would define a gene's presence in the Cancer Gene Census (CGC) and its categorization based on these features, in order to better assess its contribution to oncogenesis. Tomasetti et al evaluated the association between the number of normal stem cell divisions and the risk of developing seventeen different forms of cancer in 69 nations globally.

Additionally, Watson and Lynch reported in that male mutation carriers have a 74 percent chance of developing colorectal cancer. In comparison to the general population, female mutation carriers have lesser speculation and hence are at a higher danger. Following that, Ali et al. reported in that these distinct behaviors result in genetic variations in tumor-suppressing genes, protooncogenes, and oncogenes, as well as genes involved in routine cellular processes. Later Asano et al. developed the mutant-embellished PCR test with an emphasis on EGFR exons 19 and 21. Messiaen et al. investigated and validated protein truncation in [27], starting with puromycin-treated EBV cell lines. Additionally, they identified the germline mutation in sixty-four out of sixty-seven individuals and thirty-two new mutations. All mutations were analyzed at the genomic and RNA levels.

In addition, Hollestelle et al. investigated and reported on a detailed molecular characterization of a cluster of forty-one human breast cancer cell lines in their study. Following this discovery, Ma et al. revealed a repair technique for heterozygous MYBPC3 (a kind of mutation) discovered in human preimplantation embryos using the unique CRISPR-Cas-stationed accuracy in a subsequent study. Following a discussion of the numerous investigations, the focus of this study is on the categorization of gene mutations into nine groups, which would further improve the diagnosis of cancer tumors by the use of clinical text evidence. TfidfVectorizer and Word2Vectorizer are two text transformation models that are used to convert text into a matrix of token counts, respectively. It is determined how well the suggested framework performs using the five machine learning classifiers, which are logistic regression, support vector machine, K-NN, and Naive Bayes. This study is being done with the health of individuals in mind, as well as to make the identification of gene mutations more efficient than the manual techniques now in use.

# Methodology

A variety of natural language processing (NLP) approaches and two text transformation models (TfidfVectorizer, and Word2Vec) are addressed in this part, as well as ML classification models of various types. Computers are able to interpret natural language because of Natural Language Processing (NLP), which is a technique developed by IBM. Syntactic Analysis is a branch of natural language processing that is focused on the grammatical component of the language. It is used to determine if natural language conforms to grammatical dogmas. The application of these grammatical principles to the words and the inference of their meaning can be accomplished through the employment of certain procedures. It is based on the meaning communicated by the text that Semantic Analysis is performed. Understanding and interpreting the meaning of the words, as well as the structural analysis of sentences, are carried out in this section.

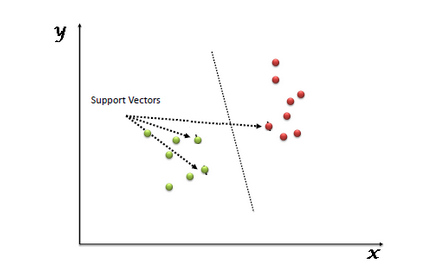
## Models

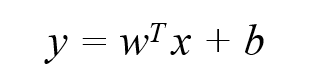
*1) Univariate Models*: The approach of Univariate analysis is used to compare and analyze the relationship between a single predictor and a response variable. The word "uni" signifies "one," stressing that the analysis considers only the influence of one variable on a dependent variable. Univariate analysis is considered one of the simplest types of data analysis since it does not consider causes or correlations in the way that regression does. Essentially, Univariate Analysis gathers data and summarizes it, along with any related trends. The Univariate analysis was used on gene feature, variation feature and text feature. In the current method we have also used Laplace smoothing along with Univariate analysis for predicting the features according to their classes.

*2) Naïve Bayes*: a type of linear classifier that is well-known for its simplicity and efficiency. The probabilistic model of naive Bayes classifiers is based on Bayes’ theorem, and the adjective naive comes from the assumption that the features in a dataset are mutually independent. Although the independence assumption is frequently violated in actuality, naïve Bayes classifiers nonetheless perform rather well under this implausible assumption. Naive Bayes classifiers, in particular for small data sizes, can outperform more powerful options. Due to its relative robustness, ease of implementation, speed, and accuracy, naive Bayes classifiers are utilized in a wide variety of applications. Several applications include disease diagnosis and treatment decision-making, classification of RNA sequences in taxonomic investigations, and spam filtering in e-mail clients. However, severe violations of the independence assumptions and non-linear classification problems can result in naïve Bayes classifiers performing quite poorly. We must bear in mind that the type of data and the nature of problem influence the categorization model to use. In fact, it is always prudent to compare various categorization models on a given dataset, considering both prediction performance and computational efficiency.

*3) Logistic Regression*: this is a categorical response variable categorization algorithm. The goal of Logistic Regression is to find a correlation between features and outcome probability. Logistic regression estimates the chance of your sample falling into one of two classifications. It can be used for multivariable classification, but it excels in binary classification because it is based on probability. The output of logistic regression is a numbered categorization, while the actual output is a numerical probability between 0 and 1. Depending on the probability, a 1 or 0 will be assigned. The procedure simply rounds the value to a negative or positive class (0 or 1).

*4) Support Vector Machines*: In machine learning, a supervised machine learning method known as the "Support Vector Machine" (SVM) is a supervised machine learning technique that may be used for both classification and regression tasks. In practice, it is most often applied to categorization difficulties. As a result of the SVM method, we depict each data item as an individual point in n-dimensional space (where n is the number of features you have), with each feature's value representing a specific position. Afterwards, we carry out classification by identifying the hyper-plane that most clearly distinguishes between the two classes.





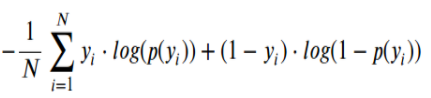
*5) K-Nearest Neighbors*:

## Other Model Features

*1) Splitting the Data – Training, Cross-Validation, Test*: It is well-known in the field of statistics and machine learning that model fitting is prone to overfitting. This is the phenomenon where a model trained to maximize performance on a single dataset has the tendency to make choices specific to that dataset, which results in a model that is too specific to be applied to any future data. This is less than ideal, since the entire goal of prediction and classification models is to construct one that can be generalized to the entire field and therefore used again and again. To reduce this phenomenon, often what is done is the known dataset is split into two – the Training set and the Testing set. The training data (which is often slightly larger) is used to train the model and then the model is applied to the testing data to assess its performance. This is beneficial in preventing overfitting as we are no longer attempting to maximize performance based on the data that the model receives for training.

We can further apply this during hyperparameter tuning (discussed below). In training hyperparameters for a model, the same overfitting phenomenon can occur which will result in an overfit hyperparameter being selected. This is where a Cross-Validation (CV) split comes into play. By splitting the training data further into a final training set and pulling some points into the Cross-Validation dataset, hyperparameter tuning will occur by training on the final training data and testing on the CV data, then final training and testing can occur.

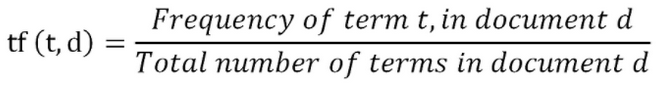
*2) Model Comparison Metric*: A probability-based classification metric known as Log Loss is the most important classification metric available. Although it can be difficult to interpret raw log-loss statistics, log-loss is nevertheless a useful tool for comparing different models. For any given problem, a smaller log loss value indicates more accurate forecasting. Log Loss is defined as the negative average of the log of corrected predicted probability for each instance divided by the number of instances.



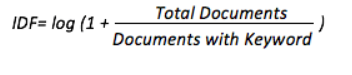
In this case, *yi* represents the actual class, while *log(p(yi)* denotes the likelihood that the class exists. *p(yi)* is the probability that *yi* equals 1. *1-p(yi)* is the probability that *yi* is zero.

*3) Feature Engineering*: It is the overall process of breaking down a text corpus into distinct pieces that may then be used as input for various natural language processing algorithms that is described as tokenization. Typically, tokenization is preceded by other optional processing processes, such as the removal of stop words and punctuation symbols, stemming or lemmatizing, and the generation of n-grams, among other things.

One choice is Term Frequency-Inverse Document Frequency (TF-IDF). This is a highly essential concept in information retrieval. Text data is represented numerically by the TF-IDF, and this is its primary function. The end result is that it reveals which word is the most essential to us among a collection of papers, which is sometimes referred to as a corpus in more technical terminology. It is frequently employed as a weighting factor in information retrieval searches, text mining searches, and user modeling searches, among other things. A common use for such models may be found in internet-based organizations such as Google, Bing, and Yahoo, where they want to know what the most essential word(s) in your inputted query are, so they can analyze it and then display you the results that are most relevant to your query. It is made up of two terms: the first computes the normalized Term Frequency and the second computes the normalized IDF (Indicator Distribution Function) (TF).Two terms are used in this calculation: the Inverse Document Frequency (IDF), which is computed as a log function of the number of documents in a corpus divided by the number of documents in which a specific term appears; and the Inverse Word Frequency (IWF), which is computed as the logarithm of the number of documents in a corpus divided by the number of documents in which a specific word appears. As the name implies, term frequency is a metric for determining how frequently a term appears in a document. Because every document in a large text dataset is varied in length, and a word can frequently appear more times in a long document than in a short one, it is important to consider the length of each document. As a result, we will divide it by the total number of phrases in the document in order to normalize it.



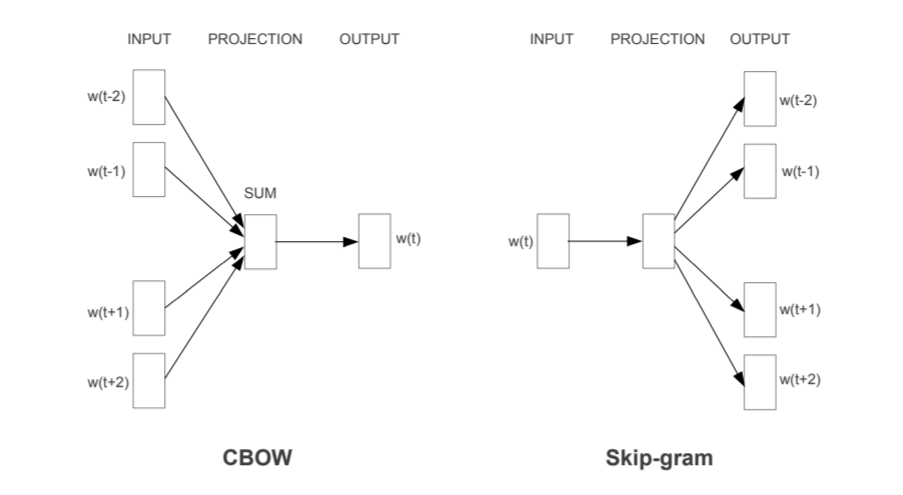
Inverse term frequency used to determine how essential a term is. Consequently, when calculating the TF, we treat all of the terms (words) as equally essential. Words such as "I," "Is," and "We" appear more frequently in a paper, yet these words do not offer any new information to our understanding or do not provide any important information to us. As a result, we can determine the inverse document frequency with relative ease.



Another choice of feature engineering is Word2Vec. This method generates vectors of words that are distributed numerical representations of word features – which could be composed of words that represent the context of the individual words present in our vocabulary – and these vectors of words are distributed numerical representations of word features. Word embeddings eventually aid in the establishment of an association between a word and another word with a comparable meaning by means of the vectors that are generated. In order to generate the word embeddings, Word2Vec can employ two distinct model architectures: the Continuous Bag of Words (CBOW) model & the Skip-Gram model.

*a) CBOW*: Despite the fact that Word2Vec is an unsupervised model in which you can provide a corpus without any label information and the model can generate dense word embeddings, Word2Vec internally leverages a supervised classification model to obtain these embeddings from the corpus in order to improve performance. CBOW architecture consists of a deep learning classification model that takes as input context words, X, and attempts to predict our target word, Y, based on those context words.

*b) Skip-Gram model*: When building the Skip-Gram model, like with all of the other word2vec models, we employ a trick that is also employed by a large number of other Machine Learning methods. Learning word embeddings is not an example of supervised learning because we do not have the labels associated with the words to use as guidance. In this case, we are using semi-supervised learning because we do not have direct labels linked with the words, but instead we are using nearby words (such as the words surrounding a context word in a sentence) as labels.



*4) Categorical Variable Encoding*: The models we tested required all inputs and outputs to be numeric, therefore we had to encode our categorical variables into a numerical format. Choosing the right encoding schema for each model was important in order to simplify processes and to reduce processing time.

One method of encoding variables is One-Hot encoding. In this process, a binary variable is created for each category, and each point has either a 1 or 0 as its value for the variable according to whether the point is a member of a class (1) or not (0) [1]. This method is ideally suited to our Naïve Bayes, Logistic Regression, and Support Vector Machines. This method has a limitation in the fact that the output of encoded variables can be very large. For example, in our data there are 224 different genes, and so to one-hot encode this there needs to be 224 columns added to the data, one for each gene. If a model is already susceptible to the curse of dimensionality or requires a lot of time and resources to train on large data, then this method of encoding is not ideal for that model [2].

For models which might suffer from the use of one-hot encoding, we decided to use Response Coding. This is a method of variable encoding in which each data point is assigned a probability of belonging to each class [3]. This allows for a significant decrease in the amount of space necessary to hold data, and therefore decreases the susceptibility to the curse of dimensionality and decreases the processing time and resources necessary.

*5) Class Balancing*: Our dataset contained an inequal number of data points from each class. Class 7 had almost 1,000 datapoints belonging to it, making it the largest class represented, while the smallest set was Class 8 with only 19 points of data.

Chart

Description automatically generated

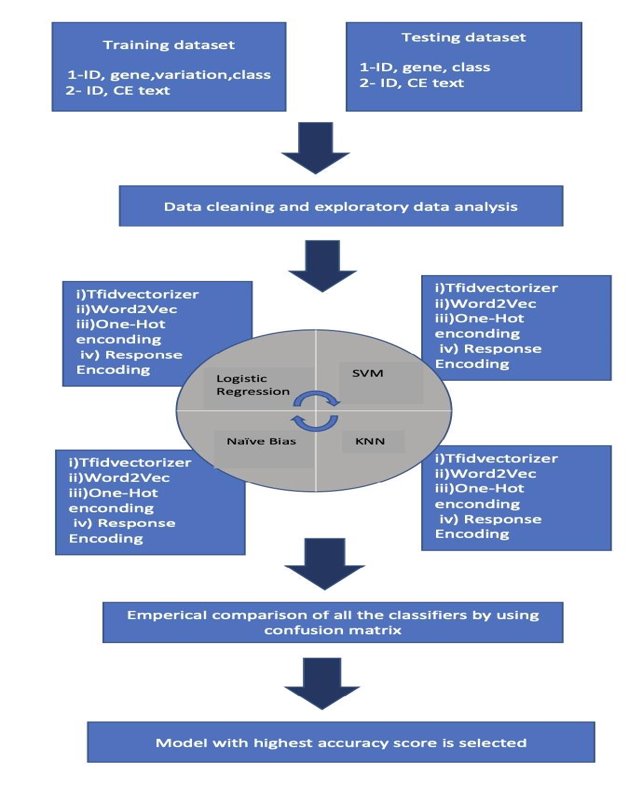
Fig. 1. Distribution of classes in the entire dataset

Because of this, it was vital that when we performed our split of the data into Training, Cross-Validation, and Test data to put a proportionate number of points in each group. This can also be known as stratifying the dataset. It ensures that each group contains at least one point from each class, which will allow our model to train, validate, and test on all classes.

In addition to splitting the data according to class, we also decided to experiment with class balancing in our modeling. Our models during training are more inclined to predict the majority class simply because that is what it sees more of, and this satisfies the need to minimize error. There is also not enough data for the algorithm to find patterns in the minority class which would distinguish it from other classes.

This is handled in class balancing by giving more weight to the minority classes and less weight to the majority classes, which will penalize the model more heavily when it misclassifies the minority class than if it misclassifies the majority class. However, it is also important that the weighting not significantly increase the error rate in the majority classes [4]. The programs we used did their best to balance these trade-offs.

We ultimately decided to test models with and without class balancing to determine whether it is necessary to do so in order to create the best model.



# Results

## Dataset

The dataset is from the Memorial Sloan Kettering Cancer Center (MSKCC) and consists of two files; one file contains the Gene information (the gene in which the genetic mutation is located) and Variation information (the aminoacid change) as well as the Class the mutation was classified on. The other file contains the clinical evidence that was used to classify that genetic mutation. The two files can be joined on the ID column which is also present in both datasets.

The Text column defines the clinical evidence that was used to classify the genetic mutation. This classification of whether the mutation was a passenger mutation or a driver (malignant) mutation was performed manually for each row in the data by researchers and oncologists. Usually this process requires the researcher to scour the medical literature that is relevant to the genetic variant of interest and identify from it whether the mutation is malignant or not.

## Data Preprocessing

The first necessary step in preprocessing our data was to check for Null values in any of the dataset columns:

Table

Description automatically generated

Fig. 2. Which rows in the dataset have a Null/NaN value for Text, Variation, or Gene Column

We can see here that the only column with null entries is the Text column, in which there are 5. For these rows, we replaced the Null value with a concatenation of the Gene and Variation columns.

The Gene and Variation columns needed no cleaning. For the Text column, we did some minor clean-up to optimize the dataset to text mining. We removed all stopwords as defined by Python’s NLTK library, replaced all special characters with a space, and then replaced all instances of multiple spaces with a single space. Finally, we converted all text to lowercase in order to ensure that word equivalency would not be impacted.

## Model Results

*Univariate Analysis – Gene Feature:* This will tell us how useful the Gene feature is in predicting the variant class. Hyperparameter tuning showed us that a very small alpha value gave the best log loss:

Chart

Description automatically generated

Fig. 3. Hyperparameter tuning for Univariate Model of Gene Feature

Train log-loss for this model was 0.991, and test log-loss was 1.172, indicating that this variable alone was very good at predicting the variant class. It correctly classified 20% of the test data points, however, indicating that this variable alone is not enough for a good accuracy.

A picture containing background pattern

Description automatically generated

Fig. 4. Confusion Matrix for Univariate Model of Gene Feature

*Univariate Analysis – Variation Feature:* This will tell us how useful the Variation feature is in predicting the variant class. Hyperparameter tuning showed us that a very small alpha value gave the best log loss:

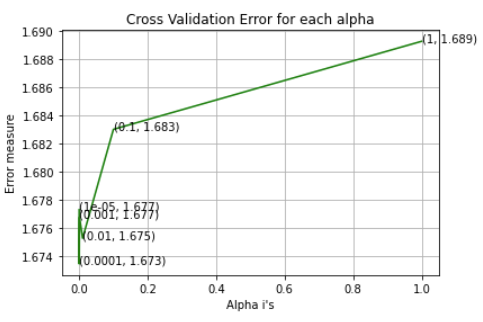


Fig. 5. Hyperparameter tuning for Univariate Model of Variation Feature

Train log-loss for this model was 0.689, and test log-loss was 1.673, indicating that this variable alone is not good at predicting the variant class. It correctly classified 27% of the test data points, which was better than the Gene variable alone though. Also, as seen in the confusion matrix, this feature favors predicting class 7, which the Gene feature struggled with, indicating that this variable does have a place in the model.

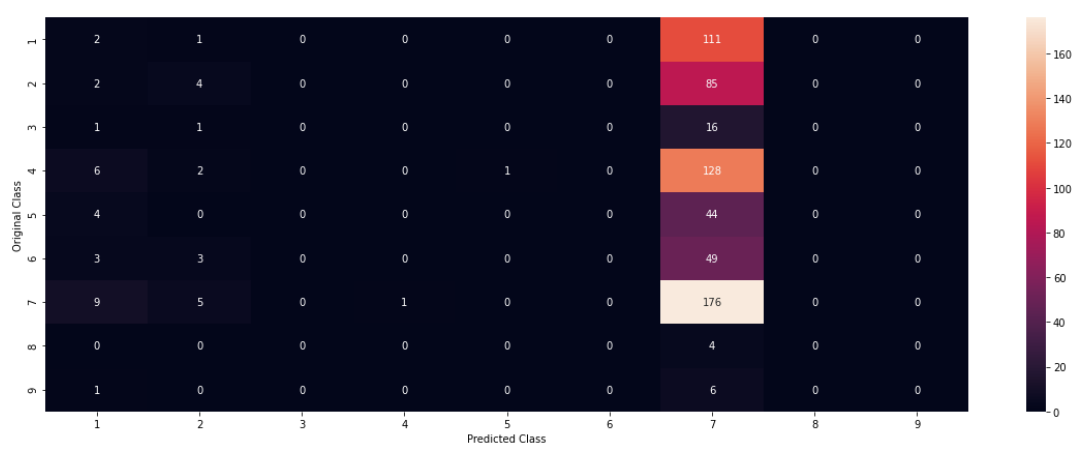


Fig. 6. Confusion Matrix for Univariate Model of Variation Feature

*Univariate Analysis – Text Feature:* This will tell us how useful the Text feature is in predicting the variant class. Hyperparameter tuning showed us that a very small alpha value gave the best log loss:

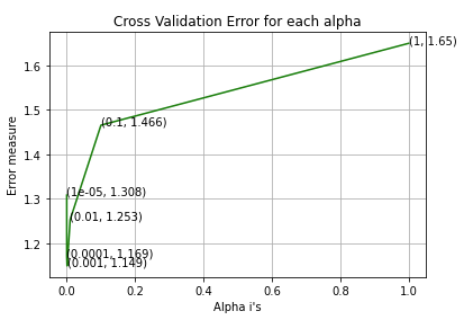


Fig. 7. Hyperparameter tuning for Univariate Model of Text Feature

Train log-loss for this model was 0.662, and test log-loss was 1.141, indicating that this variable alone is very good at predicting the variant class. It correctly classified 18% of the test data points, which was worse than the Gene variable alone. From the confusion matrix, we can see slight improvements in classification for some areas, all of which indicates that this feature might be very useful in predicting classes.

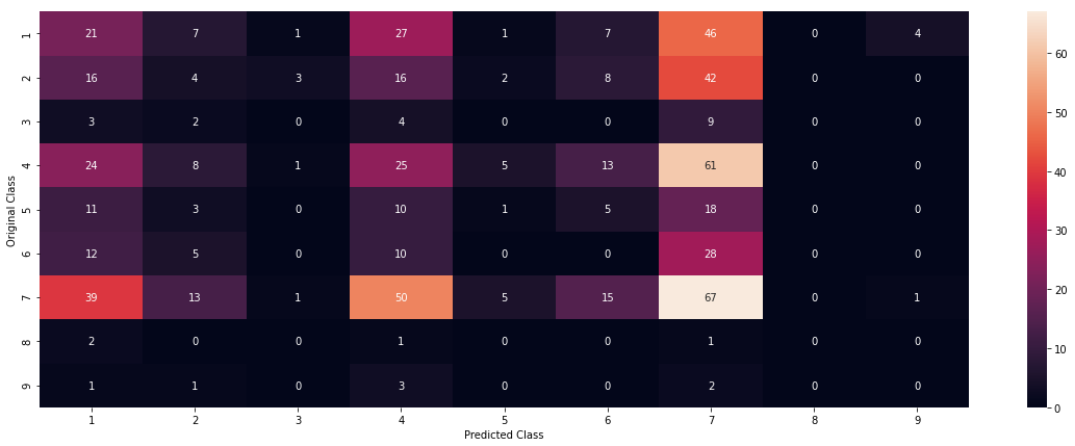


Fig. 8. Confusion Matrix for Univariate Model of Text Feature

*Naïve Bayes:* This basic model used one-hot encoded categorical variables and TF-IDF vectorization of the Text feature. This was our baseline model, and we would attempt to improve on it with successive models. Hyperparameter tuning showed us that a very high alpha value gave the best log loss:

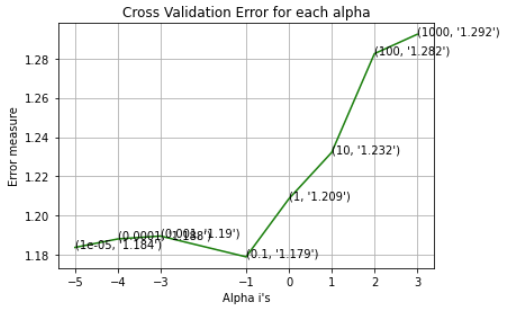


Fig. 9. Hyperparameter tuning for Naïve Bayes Model

Train log-loss for this model was 0.752, and test log-loss was 1.164, indicating that this model performed quite well. It correctly classified 64% of the test data points, which is better than a purely random model.



Fig. 10. Confusion Matrix for Naïve Bayes Model

*Logistic Regression:* This model used one-hot encoded categorical variables and TF-IDF vectorization of the Text feature. It was tested both With Class Balancing and Without Class Balancing. For the Class Balancing model, hyperparameter tuning showed us that a very low alpha value gave the best log loss:

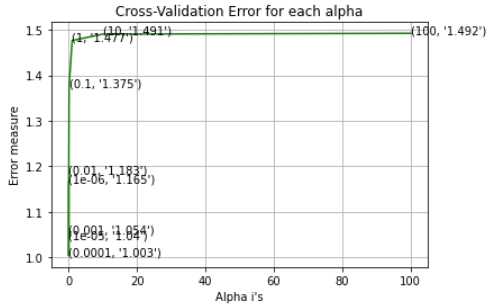


Fig. . Hyperparameter tuning for Logistic Regression Model with Class Balancing

Train log-loss for this model was 0.356, and test log-loss was 0.951, indicating that this model performed very well. It correctly classified 68% of the test data points, which is better than our Naïve Bayes model.



Fig. . Confusion Matrix for Logistic Regression Model with Class Balancing

For the Logistic Regression Model Without Class Balancing, the hyperparameter tuning gave the same value of alpha as the one with class balancing:

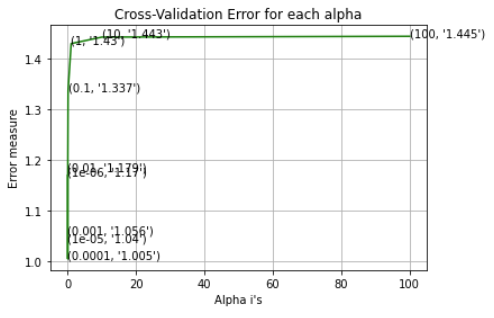


Fig. 13. Hyperparameter tuning for Logistic Regression Model Without Class Balancing

Train log-loss for this model was 0.348, and test log-loss was 0.955, indicating that while this model performed very well, it might have a slightly-increased tendency to overfit than the Logistic Regression with Class Balancing. It correctly classified 67% of the test data points, which is just one percent less than the other Logistic Regression model.



Fig. 14. Confusion Matrix for Logistic Regression Model Without Class Balancing

*Linear Support Vector Machines:* This model also used one-hot encoded categorical variables and TF-IDF vectorization of the Text feature. It was only tested With Class Balancing because it is well-known that SVM do not perform well on unbalanced data [5]. Hyperparameter tuning showed us that a very low alpha value gave the best log loss:

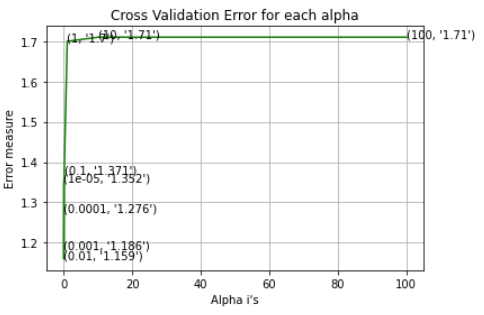


Fig. 15. Hyperparameter tuning for SVM Model

Train log-loss for this model was 0.303, and test log-loss was 0.993, indicating that this model was preferred over the logistic regression without class balancing but not as good as the one with class balancing. It correctly classified 67% of the test data points, which is about the same accuracy as the Logistic Regression models.



Fig. 16. Confusion Matrix for SVM Model with Class Balancing

*KNN Classifier:* This model used response coding for the categorical variables and TF-IDF vectorization of the Text feature. While it should have been trained and tested with Class Balancing, there is no reliable python package capable of balancing the classes and writing our own was outside the scope of this project. We decided to test it anyways because our dataset is not too unbalanced, in that no previous models attempted to get a high accuracy via assigning every point to one class. Hyperparameter tuning showed us that a high alpha value gave the best log loss:

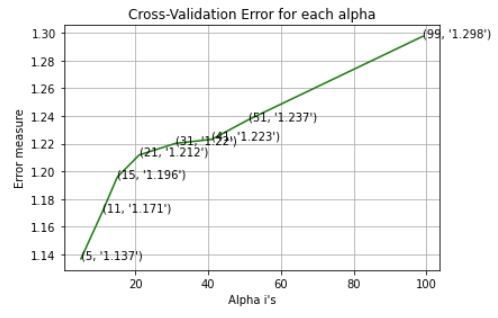


Fig. 17. Hyperparameter tuning for KNN Model

Train log-loss for this model was 0.856, and test log-loss was 1.096, which is not as bad as we suspected it might be. It also correctly classified 64% of the test data points, which is not too much lower than the models above.

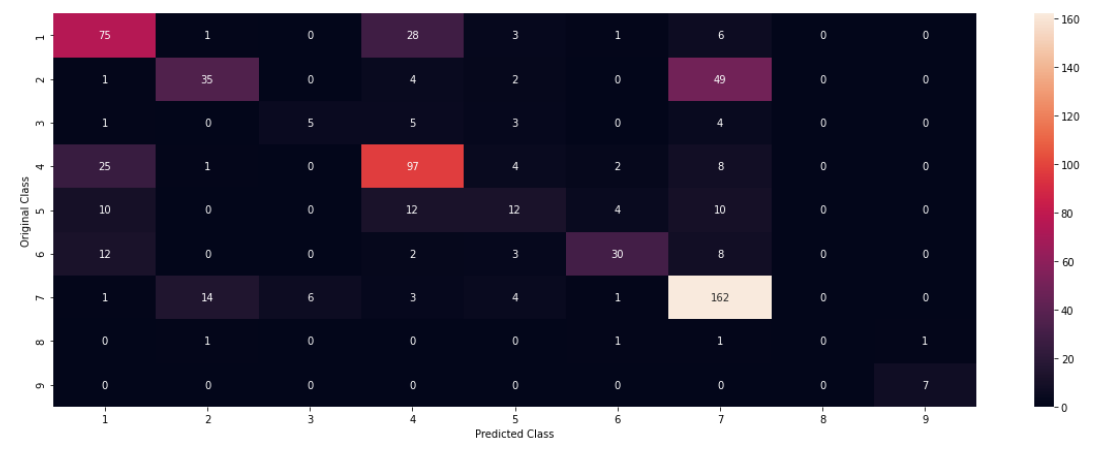


Fig. 18. Confusion Matrix for KNN Model with Class Balancing

*Revisiting Logistic Regression – Word2Vec:* Out of all the models above, Logistic Regression Without Class Balancing performed the best. To compare whether another Text Engineering method would improve our model, we implemented Word2Vec on this best model. Hyperparameter tuning showed us that a high alpha value gave the best log loss:

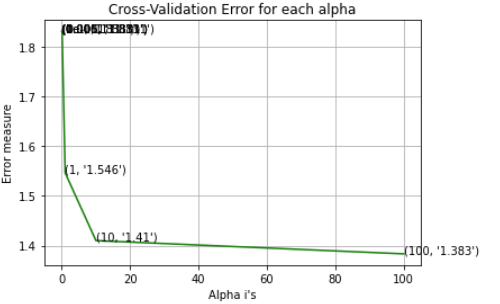


Fig. 19. Hyperparameter tuning for Logistic Regression Model with Class Balancing and Word2Vec

Train log-loss for this model was 1.263, and test log-loss was 1.391, which is much worse than the TF-IDF model was. It also only correctly classified 52% of the test data points, which is not even better than our baseline Naïve Bayes model. This indicates that TF-IDF is a better choice for our data.

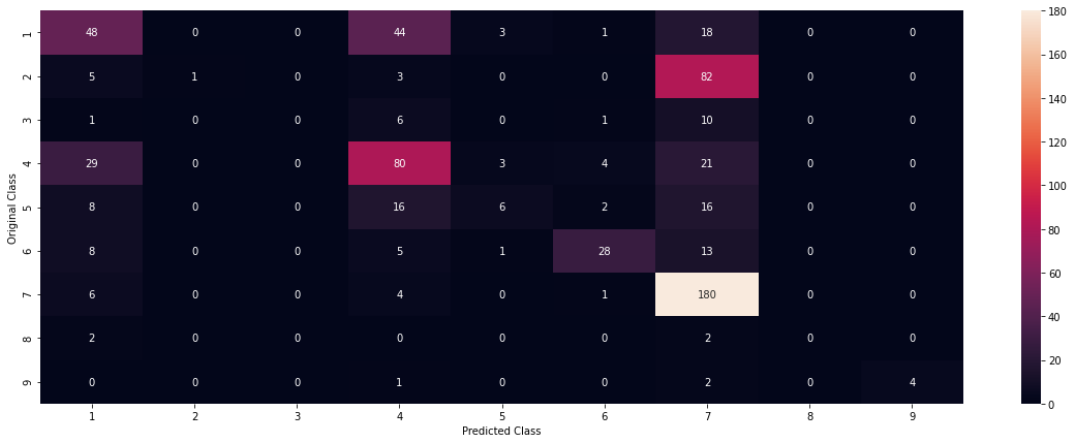


Fig. 20. Confusion Matrix for Logistic Regression Model with Class Balancing and Word2Vec

*Final Comparisons*: The following table shows the train, CV, and test log losses, as well as the accuracy for each model:

Table I.

Log-Loss and Accuracy for All Models

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Model** | **Train Log-Loss** | **CV Log-Loss** | **Test Log-Loss** | **Accuracy** |
| Naïve Bayes + TF-IDF | 0.752 | 1.179 | 1.164 | 64% |
| **Logistic + Class Balancing + TF-IDF** | **0.356** | **1.003** | **0.951** | **68%** |
| Logistic + Without Class Balancing + TF-IDF | 0.348 | 1.055 | 0.955 | 67% |
| SVM + Class Balancing + TF-IDF | 0.303 | 1.029 | 0.993 | 67% |
| KNN + TF-IDF | 0.856 | 1.137 | 1.096 | 64% |
| Logistic + Class Balancing + Word2Vec | 1.263 | 1.373 | 1.391 | 52% |

We can see here that the Naïve Bayes – our baseline model – performed at 1.164 Log-Loss and 64% accuracy. KNN performed only slightly better, followed by SVM and Logistic Regression Without Class Balancing. The best model we ran was the Logistic Regression model With Class Balancing. Upon attempting to improve this using Word2Vec Instead of TF-IDF with our previous best model, we actually saw a sharp increase in log-loss and an accuracy worse than even our baseline model. This indicates that a Logistic Regression model With Class Balancing is the best fit for this dataset.

# Conclusion

Personalized medicine and the research of it are not only amazing opportunities, but necessary ones. Oftentimes, we try to group people together and treat the group as a single entity, which is especially dangerous in medicine as there are even more areas in which people differ. This is especially pronounced in the field of Cancer diagnosis and identification, because cancer naturally falls into category-like groups. It is of great interest to the field of data mining and machine learning that we be able to automatically group similar patients together in automatic ways, rather than the subjective ways we have done so in the past. The more granular the groups can become, the more personalized the medicine becomes which will hopefully translate to better outcomes. Machine learning and data mining also provide the opportunity of automation in the areas of medicine that are often tedious and prone to human error. These reasons are why we felt so strongly that a data mining approach would be of profound impact to the field.

Our specific dataset focused on taking data compiled by researchers and oncologists about certain genetic mutations and attempted to predict what class each variant belonged to which, subsequently, would allow doctors and researchers to know whether or not a specific genetic variant was a driver or passenger mutation. The creation of a model that could automatically predict this from textual data and other evidence would greatly reduce the manpower that is traditionally necessary to complete the classification of genetic variants.

We focused on a subset of models that were commonly used in the field of data mining, as well as generalizable to many situations and customizable depending on data source characteristics. Combining these models with various feature engineering and modeling techniques allowed us to experiment with a wide range of variable model features in order to find the best model for our data.

Out of all the models and model feature groups we ran, the best model based on our model comparison metrics was a Logistic Regression model With Class Balancing, TF-IDF Text featurization, and one-hot encoding. This gave us our lowest log-loss value on test data of 0.951, and the best accuracy at 68%. Other models that were similar to this also produced comparable results, including the Logistic Regression model Without Class Balancing with a log-loss of 0.955 and the SVM model With Class Balancing and future research should focus on these models and whether they can be improved.

Our best model’s accuracy was only 68% for the entire dataset, which is high for the field and problem application but statistically lower than what would be preferred. However, a deeper look into where this model failed can provide insight into what exactly was it about this model that was good and what needs to be improved.

Table II.

Accuracy of Each Individual Class for Best Model

|  |  |  |
| --- | --- | --- |
| **Actual Class** | **Number of Points Correct** | **Accuracy** |
| 1 | 76 | 67% |
| 2 | 47 | 52% |
| 3 | 5 | 28% |
| 4 | 103 | 75% |
| 5 | 17 | 35% |
| 6 | 29 | 53% |
| 7 | 165 | 86% |
| 8 | 1 | 25% |
| 9 | 7 | 100% |

It is clear from this table that the best predicted class is class 9, at 100% accuracy. However, this class only had 7 points available in the testing set, so this perfect accuracy is less likely to be a result of an amazing model and more likely to be simply because of chance. Class 7, however, had significantly more points belonging to it, and resulted in an 86% prediction accuracy. In fact, besides class 9 the accuracy actually decreases as the number of points in the class increases. This combined with the fact that the best model was the logistic regression that included class balancing indicates that a model for this type of data will best be utilized on datasets that are already class-balanced and which contain a large enough amount of data.

However, this does not disregard our proposed application of data mining. It is true that it does not seem to be advantageous (in most cases) to make decisions for underrepresented or smaller-sized classes based on machine learning. Nonetheless, if it is known beforehand that a certain class is more prevalent in practice or in data, then applying a machine learning model such as ours to that data in order to identify that particular class might be more promising.

The largest issue we experienced throughout this analysis was that there are only three columns in this dataset, and one of them is a textual column that is so long that most word processors are unable to show the whole cell value. This means that the corpus of words was very large, yet our accuracy was only at 68%. We are not experts in genetics or gene variants, but knowing that each row was created by in all likelihood a single person, we see that there might be model performance improvements if the text data were reviewed by other field experts. An alternative method of fixing this could be that when compiling the text data, the experts would only include words that they suspect could provide insight into the gene and leave out descriptive words that might be descriptive but provide no insight. We removed stopwords via data processing via scripts, however our stopwords were only most commonly-used-words in general. Another improvement might be if a different stopword list could be used, one with both common words in general English and common words in genetics description and cancer study. TF-IDF does its best to exclude these by including the measure of how many documents a word appears in, however even slight changes to the stopword list might result in impactful changes to model performance.

In conclusion, the model we found to perform best – the Logistic Regression with Class Balancing, TF-IDF, and One-Hot encoding – was a pretty good model in comparison to others in the field but appears to be lacking statistically. Instead, we were able to demonstrate that creating data mining models on well-balanced class data with enough points included is a simple yet effective approach to automating this field. We also found that machine learning models perform best in predicting whether a data point belongs to the majority class or not. Finally, we were able to show that the application of machine learning to the field of classifying and predicting gene variants is a promising endeavor that should not be overlooked.

# Works Cited

|  |  |
| --- | --- |
| [1] | J. Brownlee, "Why One-Hot Encode Data in Machine Learning?," Machine Learning Mastery, 28 July 2017. [Online]. Available: https://machinelearningmastery.com/why-one-hot-encode-data-in-machine-learning/. [Accessed 24 April 2022]. |
| [2] | S. Kumar, "Stop One-Hot Encoding your Categorical Features — Avoid Curse of Dimensionality," Medium.com, 2 March 2021. [Online]. Available: https://medium.com/swlh/stop-one-hot-encoding-your-categorical-features-avoid-curse-of-dimensionality-16743c32cea4. [Accessed 24 April 2022]. |
| [3] | D. Thakkar, "Response Coding for Categorical Data," Medium.com, 5 July 2019. [Online]. Available: https://medium.com/@thewingedwolf.winterfell/response-coding-for-categorical-data-7bb8916c6dc1. [Accessed 24 April 2022]. |
| [4] | K. Singh, "How to Improve Class Imbalance using Class Weights in Machine Learning," Analytics Vidhya, 6 October 2020. [Online]. Available: https://www.analyticsvidhya.com/blog/2020/10/improve-class-imbalance-class-weights/. [Accessed 24 April 2022]. |
| [5] | J. Brownlee, "Cost-Sensitive SVM for Imbalanced Classification," Machinel Learning Mastery, 31 January 2020. [Online]. Available: https://machinelearningmastery.com/cost-sensitive-svm-for-imbalanced-classification/. [Accessed 24 April 2022]. |