



University of New Haven

TAGLIATELA COLLEGE OF ENGINEERING

---

ECECS Department

# **DSCI 6670 Artificial Intelligence Final Project**

## **Cancer Diagnosis using Machine Learning**

By:

**Ram Nikhilesh Mekala**

**Srimanikanta Arjun Karimalammanavar**

Course Instructor:

**Dr. Vahid Behzadan**

## Introduction

It's safe to say there are too many manual processes in medicine. The lab values, diagnoses, and other chart notes had to be written on paper. We always knew this was an area in which technology could help improve the workflow and hoped it would also improve patient care. Since then, advancements in electronic medical records have been remarkable, but the information they provide is not much better than the old paper charts they replaced. If technology is to improve care in the future, then the electronic information provided to doctors needs to be enhanced by the power of analytics and machine learning.

## Project Motivation

The field of the medicine has developed with lots of data accumulation. Especially with the upcoming of advanced algorithms regularly, the field of AI has become a "pioneer" in achieving accuracy at "human levels". We use some tools of Machine learning and Deep learning to try and diagnose the classes of cancer by reading "**clinical literature**" which are reviewed and tailored by researchers and scientists from the field.

## Data

The data that we have gathered is from a Kaggle competition - [Personalized Medicine: Redefining Cancer Treatment](#). The data is collectively gathered from 4 files

- training\_variants - a comma separated file containing the description of the genetic mutations used for training. Fields are ID (the id of the row used to link the mutation to the clinical evidence), Gene (the gene where this genetic mutation is located), Variation (the aminoacid change for this mutations), Class (1-9 the class this genetic mutation has been classified on)
- training\_text - a double pipe (||) delimited file that contains the clinical evidence (text) used to classify genetic mutations. Fields are ID(the id of the row used to link the clinical evidence to the genetic mutation), Text (the clinical evidence used to classify the genetic mutation)
- test\_variants - a comma separated file containing the description of the genetic mutations used for training. Fields are ID (the id of the row used to link the mutation to the clinical evidence), Gene (the gene where this genetic mutation is located), Variation (the aminoacid change for this mutations)
- test\_text - a double pipe (||) delimited file that contains the clinical evidence (text) used to classify genetic mutations. Fields are ID(the id of the row used to link the clinical evidence to the genetic mutation), Text (the clinical evidence used to classify the genetic mutation)

## Data Preprocessing

Since we are going to be dealing with text data, we first need to clean it by replacing every special character with space, multiple spaces with space, converting all characters with into lower case and remove the **stopwords** from our data using **Natural Language Toolkit** package. We look for empty rows in the **text** column and fill it by concatenating the corresponding row values of **Gene** and **Variation**. Then we split the data into **70%** train data and **30%** test data.

**Term Frequency – Inverse Document Frequency** is a numerical statistic that is intended to reflect how important a word is to a document in a collection or **corpus**. It is often used as a weighting factor in searches of information retrieval, text mining, and user modeling.

The tf-idf weighted value for word  $t$  in document  $d$  thus combines term frequency  $tf$  with  $idf$ :

$$tf_{t,d} = \log_{10}(\text{count}(t,d) + 1)$$

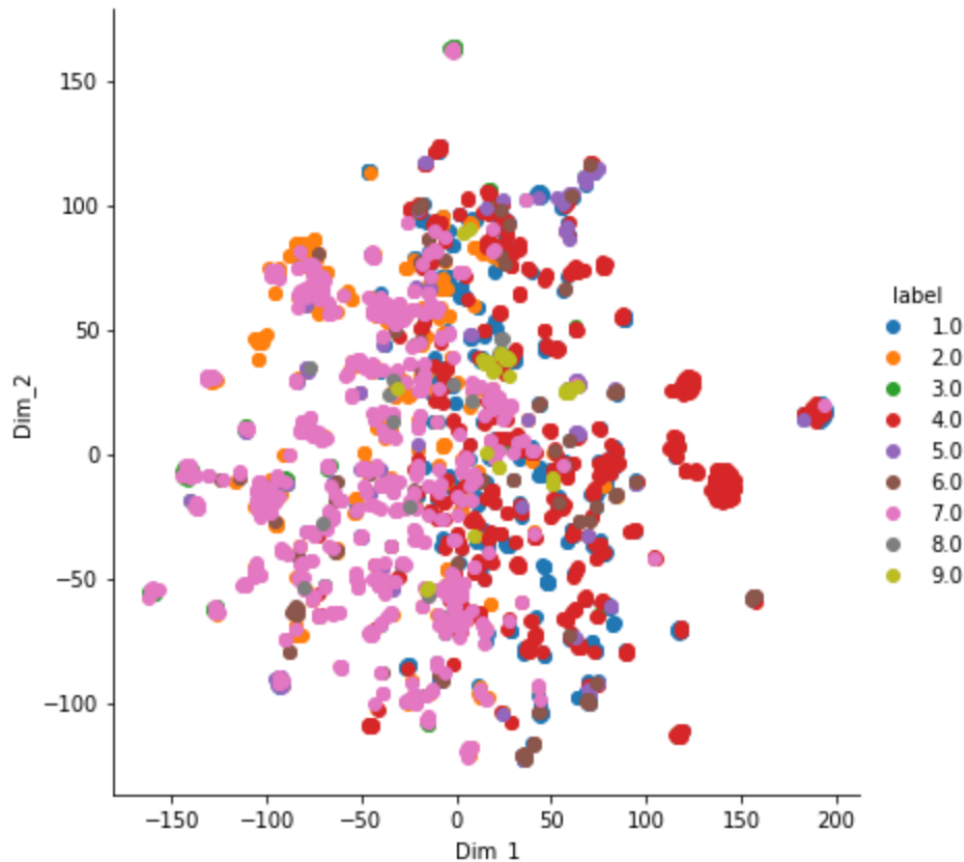
$$idf_t = \log_{10} \left( \frac{N}{df_t} \right)$$

$$w_{t,d} = tf_{t,d} \times idf_t$$

### **Average Word2Vec**

Word 2 vec model is very powerful models when we need to have the dependency. It will construct a  $d$ - dimensional vector for every word we have. Higher the dimension more powerful the model. We have constructed the word 2 vectors using the genism module. Then we have looped all over training data and calculated the w2v for every word present in the review and sum them divide with total number of words in the review.

**t-Distributed Stochastic Neighbor Embedding (t-SNE)** is a technique for dimensionality reduction that is particularly well suited for the visualization of high - dimensional datasets. We use t-SNE to plot our **y\_train** and get the following plot.



## Machine Learning Models

We are going to be using 3 classifying Machine learning models for our project.

1. Support Vector Machine with **Linear Kernel**
2. Support Vector Machine with **Radial Basis Function Kernel**
3. **Multinomial** Naïve Bayes Classifier

## GridSearchCV

Grid Search is the process of performing hyperparameter tuning in order to determine the optimal values for a given model. As mentioned above, the performance of a model significantly depends on the value of hyperparameters. Note that there is no way to know in advance the best values for hyperparameters so ideally, we need to try all possible values to know the optimal values. Doing this manually could take a considerable amount of time and resources and thus we use GridSearchCV to automate the tuning of hyperparameters.

## Support Vector Machines

A SVM is a supervised learning model that uses classification algorithms for two-group classification problems. After giving an SVM model sets of labeled training data for each category, they're able to categorize new text.

**“The objective of the support vector machine algorithm is to find a hyperplane in an N-dimensional space(N — the number of features) that distinctly classifies the data points.”**

### SVM with Linear Kernel

**Linear Kernel** is used when the data is Linearly separable, that is, it can be separated using a single Line. It is one of the most common kernels to be used. It is mostly used when there are a Large number of Features in a particular Data Set.

### SVM with Radial Basis Function Kernel

RBF kernel is a function whose value depends on the distance from the origin or from some point. Gaussian Kernel is of the following format;

$$K(X_1, X_2) = \text{exponent}(-\gamma \|X_1 - X_2\|^2)$$

$\|X_1 - X_2\|$  = Euclidean distance between X1 & X2

## Naïve Bayes Methods

A Naive Bayes classifier is a probabilistic machine learning model that's used for classification task. The crux of the classifier is based on the Bayes theorem.

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

### Multinomial Naïve Bayes Classifier

Multinomial Naive Bayes is a specialized version of Naive Bayes that is designed more for text documents. Whereas simple naive Bayes would model a document as the presence and absence of particular words, multinomial naive bayes explicitly models the word counts and adjusts the underlying calculations to deal with in.

## Data Preprocessing

We create a dictionary with words and their frequencies. We rank each words based on their frequency as word with highest frequency ranked 1 and lowest frequency ranked lowest. We then apply **padding** to make all the sentences of equal length. After applying ranking and padding our data would look like,

Total number words present in first review after padding:  
375

List of word indexes present in first review padding:

[	62	184	96	104	1490	215	39	1	726	47	142	23
104	186	13	552	1	226	254	186	1309	13	552	292	
1100	2359	13	552	316	162	797	142	14	577	1100	632	
3892	599	719	1251	381	491	350	56	7	215	39	868	
918	485	1848	2543	46	1017	148	2308	1058	1100	13	552	
48	4149	0	1848	1059	710	12369	2645	77	200	895	599	
810	803	277	236	943	726	13	552	123	127	26	20	
2765	895	23	273	381	491	350	13	552	1100	141	24	
677	127	19	13	552	283	8038	122	283	0	141	323	
7205	5321	270	13	552	70	162	588	18616	386	632	327	
79	5243	1089	49	5003	7780	142	146	807	102	1341	532	
64	1	2645	39	215	404	728	3394	16	232	215	7	
60	583	129	19480	1731	27	336	1230	20	13	552	638	
65	24	62	45	215	2193	2020	39	265	354	30	1	
638	1118	1102	1	184	980	142	23	104	252	4773	9086	
66	330	3984	66	140	16	53	215	7	60	625	174	
2	5	39	1	60	4773	981	3984	2149	212	39	1	
1924	60	53	681	1	1030	142	699	16	1359	65	6378	
16	538	65	8519	81	11424	328	77	618	705	1857	1	
13	552	215	1861	65	9249	81	1415	328	221	846	377	
49	215	1	142	14	601	385	404	93	1924	13	552	
1	191	728	8614	142	14	104	263	2150	1828	2955	448	
142	0	24	6531	17268	198	373	4244	12794	316	161	142	
146	77	4387	19	1	52	142	23	104	4773	9086	65	
246	807	102	286	13	552	1	142	146	1537	243	6	
22	260	8093	1177	260	4773	9086	205	2329	13	552	7443	
213	142	146	4773	9086	4771	16427	221	1414	4773	9086	142	
14	1537	3074	1130	5735	111	3984	4700	636	1508	48	983	
65	871	13	552	579	162	528	1750	142	146	48	770	
148	13	552	42	176	1059	20	438	807	102	7	897	
13	552	88	2645	215	39	186	805	131	141	4724	1714	
810	538	0]										

The above figure is the first review with words replaced by their corresponding ranks and the 0 at last shows that we used zeroes to pad the sequence.

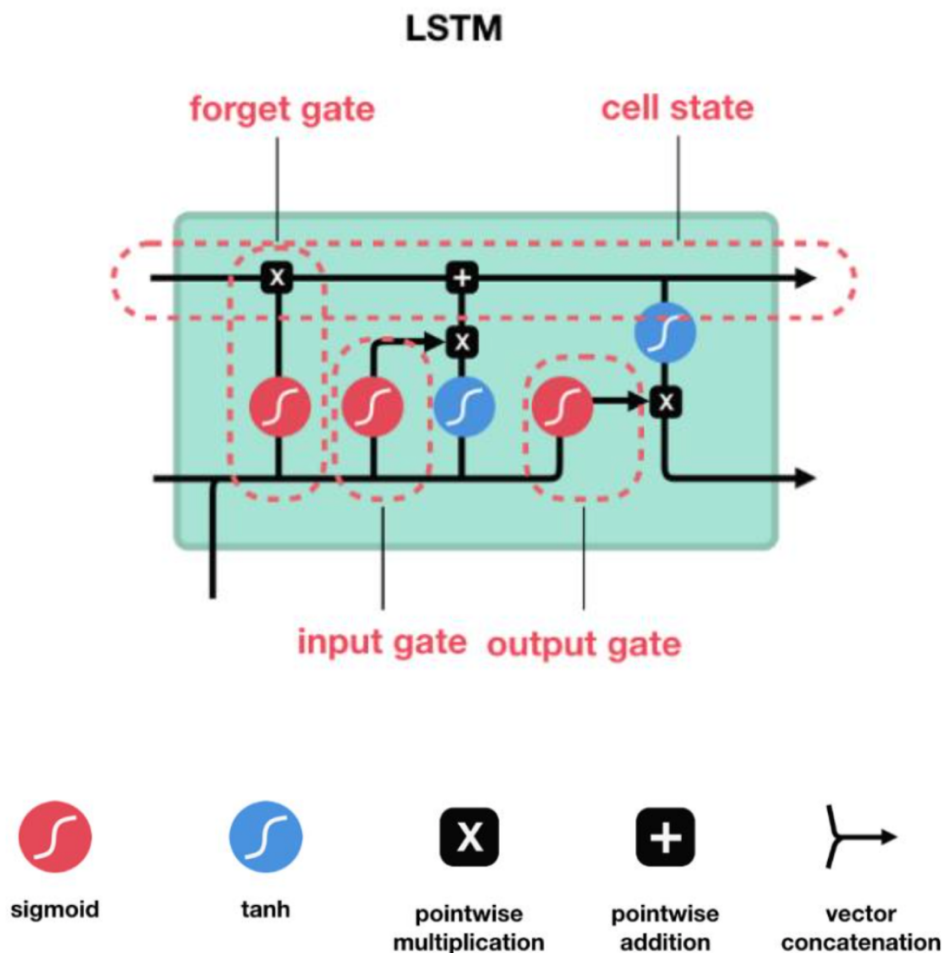
## Deep Learning Models

We are going to be using 2 Deep Learning models,

1. Long Short -Term Memory
2. Bidirectional RNN with GRU

## Long Short-Term Memory

The core concept of LSTM's is the cell state, and its various gates. The cell state act as a transport highway that transfers relative information all the way down the sequence chain. You can think of it as the “memory” of the network. The cell state, in theory, can carry relevant information throughout the processing of the sequence. So even information from the earlier time steps can make its way to later time steps, reducing the effects of short-term memory. As the cell state goes on its journey, information gets added or removed to the cell state via gates. The gates are different neural networks that decide which information is allowed on the cell state. The gates can learn what information is relevant to keep or forget during training.



## Bidirectional RNN with GRU

If we want to have a mechanism in RNNs that offers comparable look-ahead ability as in hidden Markov models, we need to modify the RNN design. Instead of running an RNN only in the forward mode starting from the first token, we start another one from the last token running from back to front. *Bidirectional RNNs* add a hidden layer that passes information in a backward direction to more flexibly process such information.

Bidirectional RNN are really just putting two independent RNNs together. The input sequence is fed in normal time order for one network, and in reverse time order for another. The outputs of the two networks are usually concatenated at each time step.

With a Gated Recurrent Unit (GRU), an Update gate is introduced, to decide whether to pass previous output to next cell or not. In other words, it decides what information to throw away and what new information to add.

## Results

From the models' architecture and processed data, we can see that **Bidirectional RNN with GRU** has provided the best performance with log loss of 0.3762.

Models	Log-Loss
Linear SVM	1.313
SVM – RBF Kernel	0.697
Naïve Bayes	1.122
LSTM	0.541
Bidirectional RNN	0.3762



## Conclusion

We have tried the simplest of models for text classification and we have achieved better performance. To improve this, we can try applying **Convolutional Markov Model**, **XLNet**, **Binary Partitioning Transformer (BPT)** etc..