Personalized Medicine: Redefining Cancer Treatment

Randheer Vennapureddy
Computer science
CWID:10457392

1.Problem Statement:

Classify the given genetic variations/mutations based on evidence from text-based clinical literature.

Source: https://www.kaggle.com/c/msk-redefining-cancer-treatment/

2.Machine Learning Objective:

Predict the probability of each data-point belonging to each of the nine classes

This project solves this problem using different Machine Learning Classifiers and make comparisons between these classifiers

3.Real-world/Business objectives and constraints.

- No low-latency requirement.
- Interpretability is important.
- Errors can be very costly.
- Probability of a data-point belonging to each class is needed.

4.Data Description:

Genetic mutation can be classified into nine different classes; This is not a significant task since the interpretation of clinical evidence is very challenging even for domain experts. Hence, illustrating the clinical evidence (text) will be critical for the success of your approach.

Datasets contain two files:

First Training/test variants provide the information about the genetic mutations, whereas the other Training/test text provides the clinical evidence (text) that our human experts used to classify the genetic mutations. Both are interlinked by the ID field.

Hence, the genetic mutation (row) with ID=15 in the file training variants, was classified using the clinical evidence (text) from the row with ID=15 in the file training text. Some of the test data is machine-generated to prevent hand labelling.

Training variants:

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 amino acid change for these mutations),
Class (1-9 the class this genetic mutation has been classified)

Training text: A delimited file that contains the clinical evidence (text) used to classify genetic mutations.
Fields:
ID row used to link the clinical evidence to the genetic mutation),
Text (the clinical evidence used to classify the genetic mutation)

5. Libraries/Packages Used:

- ScikitLearn
- Scipy
- Numpy
- Pandas
- Matplotlib
- nltk

6.Train, cross Validation and Test Datasets

The dataset is randomly split into two parts 80% Train and 20% Test and again split the training set into further two parts Training set and Cross Validation Set with 80% and 20% respectively. Machine Learning model is trained upon train set, Cross Validation is used to tune the Hyperparameters for the model, Performance of the model is tested on test set. While Splitting the data into train, test and CV we give stratify=true as the parameter to make all train, CV and test sets follow the same distribution.

7.Exploratory Data Analysis:

Features Gene and Variation in the dataset are categorical, so we apply one hot encoding and response encoding depending upon the model.

For feature Text we apply text preprocessing

In this process we remove all the extra spaces, replace every special char with space, convert all the characters into lower case and remove all the stop words.

For feature Text we apply One hot encoding which is similar to BOW or Response Encoding depending upon the model type.

8.Machine Learning Models

For Medical dataset, it is important to calculate the probability of a sample belonging to a particular class is as significant as classifying the sample. In order to obtain the probability value we apply calibrated classifier for the trained model, so we consider log loss as the loss function for this dataset.

$$H_p(q) = -\frac{1}{N}\sum_{i=1}^{N} y_i \cdot log(p(y_i)) + (1 - y_i) \cdot log(1 - p(y_i))$$

Binary Cross-Entropy / Log Loss

1.Naïve Bayes

Hyperparameter: Laplace Smoothing Alpha

We tune Hyperparameter using Grid Search approach

Naïve Bayes model usually works well with high Dimensional Data so we used One Hot Encoding for Gene, variation and Text columns and making it total 57039 Features (plus ”ID”).

This Model has given log loss for Train, CV and Test are 0.9, 1.27 and 1.21 respectively with 0.1 as Alpha value (Hyperparameter) and has 39.84% misclassification percentage. Even though Naïve Bayes is a weaker model because it makes lot of assumptions on the data, it is giving a better log loss value. We get optimal value for hyperparameter alpha by plotting a curve (error measure vs alpha values)

We observe there is no overfitting for this model.

2.KNN

Hyperparameter: K

We tune Hyperparameter using Grid Search approach

KNN does not works well upon High Dimensional Data due to curse of dimensionality, so we have used Response Encoding for Gene, Variation and Text columns and making it total 28 Features (plus ”ID”).

This Model has given log loss for Train, CV and Test are 0.705, 1.13 and 1.002 respectively with 15 as K value (Hyperparameter) and has 39.47% misclassification percentage. KNN has given slightly better performance than Naïve Bayes. But Major drawback of this KNN Model is, it is not easy to interpret the results. When dealing with

Real World Medicine Data We need Models which can be interpreted so that domain experts can analyze which features has influenced the model to make that particular decision.

Even though KNN model has slightly better performance than Naïve Bayes, still Naïve Bayes is better model for this Dataset because Naïve Bayes model can be easily interpreted.

We observe there is no overfitting for this model.

3.Logistic Regression

Hyperparameter: we have used L2 regularization for this Model and so lambda (named it alpha because lambda is the keyword in python) is the Hyperparameter

Logistic Regression Works well upon High Dimensional Data so we used one hot Encoding for Gene, Variation and Text columns and making it total 57039 Features (plus \`ID\`).

This Model has given log loss for Train, CV and Test are 0.614, 1.143 and 1.048 respectively with 0.001 as lambda value (Hyperparameter) and has 34.77% misclassification percentage. Logistic Regression given better performance than Naïve Bayes and KNN.

Advantage of using Logistic Regression is its results can be easily interpreted so we can analysis which features among 57039 feature space has made the model to classify a particular sample. Feature Importance for Logistic regression can be known by looking at the absolute values of the respective feature weights.

Logistic Regression is implemented in two ways in this project one with oversampling (Class balancing) and other without Oversampling. The class Balancing dataset with Logistic Regression performed better and it was able to classify the minor classes with high precision (Results can be seen through Confusion Matrix)

We have SGD Classifier with loss=\`log\` as the parameter

We observe there is no overfitting for this model.

4.SVM

Hyperparameter: we have used L2 regularization for this Model and so lambda (named it alpha because lambda is the keyword in python) is the Hyperparameter

We have SGDClassifier with loss='hinge' as the parameter and we have Linear Kernel.

SVM Works well upon High Dimensional Data so we used one hot Encoding for Gene, Variation and Text columns and making it total 57039 Features (plus \`ID\`).

This Model has given log loss for Train, CV and Test has 0.739, 1.132 and 1.063 respectively with 0.01 as lambda value (Hyperparameter) and has 36.84%

misclassification percentage. SVM with linear kernel has given better performance than Naïve Bayes and KNN.

We have implemented linear SVM with balanced Dataset(OverSampling), This was able to classify the minor classes better. Linear SVM model is also a better model for this Dataset because the model can be easily interpreted similar to Logistic regression. Feature Importance for Linear SVM can be known by looking at the absolute values of the respective feature weights.

We observe there is no overfitting for this model.

5.Random Forest

Hyperparameter: n_estimators and depth are two Hyperparameters for this Random Forest

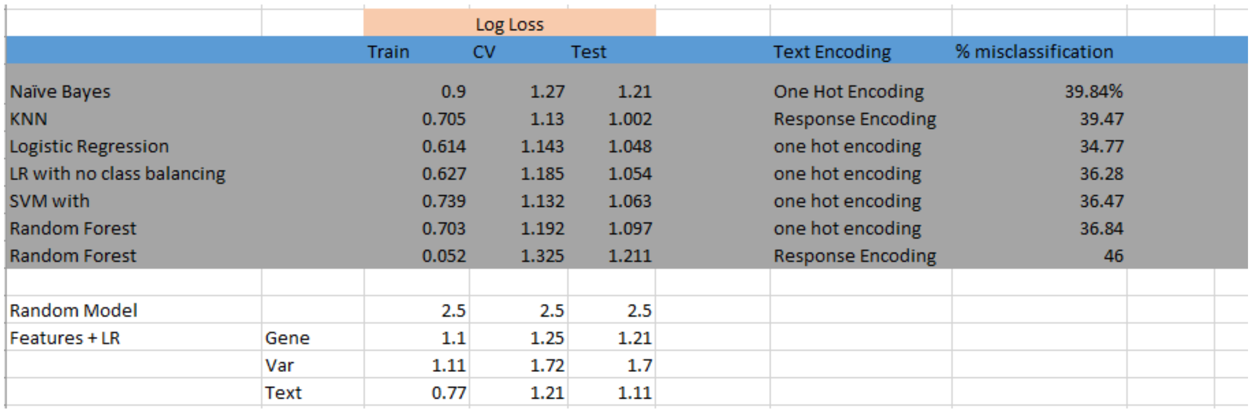
By using Grid search for hyperparameter tuning we got depth=5 and n_estimators=1000 upon testing on cross Validation set.

Random forest and Decision trees perform better in low Dimensional space we have trained two models taking both One Hot Encoding and Response Encoding individually. The Random Forest with one hot encoding has given log loss for Train, CV and Test has 0.703, 1.192 and 1.097 respectively, and 36.84 as misclassification percentage. This has performed better than Naïve Bayes and KNN and has performance similar to SVM and Logistic Regression.

Random forest has been able to correctly classify the minor classes with better precision than other models, But overall even advanced classifier like Random forest performed similar to SVM and Logistic Regression.

Random Forest trained upon response Encoded Dataset has overfitted even after taking the optimal values for Hyperparameter. This model has given 0.052 as log loss comparing to test log loss has 1.097 which means this model has completely overfitted. So this model is not very useful for this Dataset.

6.Comparing Table



		Log Loss			Text Encoding	% misclassification
		Train	CV	Test		
Naïve Bayes		0.9	1.27	1.21	One Hot Encoding	39.84%
KNN		0.705	1.13	1.002	Response Encoding	39.47
Logistic Regression		0.614	1.143	1.048	one hot encoding	34.77
LR with no class balancing		0.627	1.185	1.054	one hot encoding	36.28
SVM with		0.739	1.132	1.063	one hot encoding	36.47
Random Forest		0.703	1.192	1.097	one hot encoding	36.84
Random Forest		0.052	1.325	1.211	Response Encoding	46
Random Model		2.5	2.5	2.5		
Features + LR	Gene	1.1	1.25	1.21		
	Var	1.11	1.72	1.7		
	Text	0.77	1.21	1.11		

7.Analysis:

This dataset we used is imbalanced, there are very few samples for some classes comparing to other classes, to overcome this, we have used oversampling for some models to correctly classify these minor classes. All the models we have used none of models had accuracy more than 80% this happened because the dataset has only around 2000 samples to train upon and there is lot of variation among the features so our models naïve Bayes, KNN, Linear SVM, logistic regression and Random Forest were unable to capture all these information from the features from this small dataset while training. If we had a bigger Dataset with lot of samples for each class the models like SVM, Random Forest would have performed much better.

Overall, among all the models we have trained, Logistic Regression is better for this dataset because it has better log loss than other models and it can work better with high dimensional feature space and logistic regression model can be easily interpreted which will be helpful for the domain Experts to analyze the features and better understand the model. We didn’t apply Dimensionality Reduction for this Dataset because we thought that wouldn’t help us to show the right features which have influenced in classifying the sample by the model. For SVM we only used linear kernel but not RBF kernel because RBF SVM model is not easy to interpret and not a right kernel. I think text kernel would have had better performance. For text preprocessing we have used one hot encoding and response encoding, text preprocessing methods like TF-IDF or Word2Vec would have given better performance, but for One hot encoding this accuracy given by these models is better. Extending logistic Regression if we have implemented Neural Network model upon these Dataset with Word2vec text preprocessing, the model would have performed better or Advanced Neural Networks like RNN would have also got better performance.