Machine Learning Engineer Nanodegree

Capstone Project

Ahmet Hamza Emra
July 30st, 2017

I. Definition

Project Overview

One of the deadly disease is cancer and the early diagnosis is very Important. As computers are getting better every day on natural language processing, I would like to create Algorithm to help experts of area. I will be mostly working with text data so I will be doing Natural Language Processing. Natural language processing (NLP) is a field of computer science, artificial intelligence and computational linguistics concerned with the interactions between computers and human (natural) languages, and, in particular, concerned with programming computers to fruitfully process large natural language corpora. Challenges in natural language processing frequently involve natural language understanding, natural language generation (frequently from formal, machine-readable logical forms), connecting language and machine perception, dialog systems, or some combination thereof. The data is coming from Kaggle.com competition.

Problem Statement

A lot has been said during the past several years about how precision medicine and, more concretely, how genetic testing is going to disrupt the way diseases like cancer are treated. But this is only partially happening due to the huge amount of manual work still required. Once sequenced, a cancer tumor can have thousands of genetic mutations. But the challenge is distinguishing the mutations that contribute to tumor growth (drivers) from the neutral mutations (passengers).

Currently this interpretation of genetic mutations is being done manually. This is a very time-consuming task where a clinical pathologist has to manually review and classify every single genetic mutation based on evidence from text-based clinical literature. MSKCC (Memorial Sloan Kettering Cancer Center) is making available an

expert-annotated knowledge base where world-class researchers and oncologists have manually annotated thousands of mutations.

We need to develop a Machine Learning algorithm that, using this knowledge base as a baseline, automatically classifies and predicts the probability of being genetic variations.

The goal is to create a general text-reader running on Android smartphones; the tasks involved are the following:

1. EDA for Gene and Variation to check if there is new and useful feature
2. Text processing
   - Cleaning the text files with stemmer and stop words
   - Count vectorization to convert texts into vectors. •
   - Tfidf transformer in order to find term frequency or inverse term frequency. We will be checking both of them to find optimal one.
3. Classification, we are going to check coupe algorithm (Naive Bayes, Logistic Regression and some ensemble models) to find the best score.
4. Return the predict probability of the new cases.

Metrics

We are going to be using the accuracy score to measure the performance of our model. Since it is classification problem, accuracy is suitably.

$$\texttt{accuracy}(y, \hat{y}) = \frac{1}{n_{\text{samples}}} \sum_{i=0}^{n_{\text{samples}}-1} 1(\hat{y}_i = y_i)$$

which is basically Percentage of how may predictions are correct.

II. Analysis

Data Exploration

The data set is from Kaggle's Personalized Medicine: Redefining Cancer Treatment competition. There are more than 3000 record with provided the information about genetic mutation, whereas with another file of clinical evidence(text) that human experts used to classify the genetic mutation.

```
train[:5]
```

	ID	Gene	Variation	Class
0	0	FAM58A	Truncating Mutations	1
1	1	CBL	W802*	2
2	2	CBL	Q249E	2
3	3	CBL	N454D	3
4	4	CBL	L399V	4

```
train_text[:5]
```

	ID	Text
0	0	Cyclin-dependent kinases (CDKs) regulate a var...
1	1	Abstract Background Non-small cell lung canc...
2	2	Abstract Background Non-small cell lung canc...
3	3	Recent evidence has demonstrated that acquired...
4	4	Oncogenic mutations in the monomeric Casitas B...

the data has following columns:

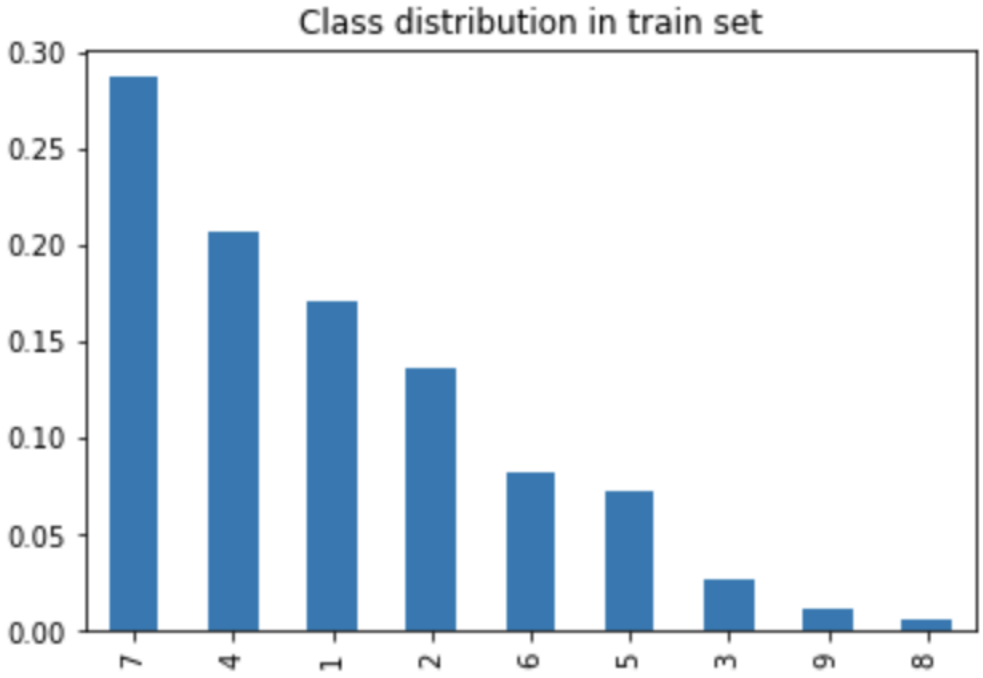
- ID (the id of the row used to link the mutation to the clinical evidence),
- Gene (the gene where this genetic mutation is located),
- Variation (the amino acid change for this mutations),
- Class (1-9 the class this genetic mutation has been classified on).
- Text (the clinical evidence used to classify the genetic mutation)

Exploratory Visualization

The following plots shows some of the finding form eda.

Fig1, represents the percentage of each genes being in training set. It looks like there are no dominant type since no of them occurs more than %10.

Fig2, is about the classes. As one can see form the plot, class 8 & 9 occur less then %5, which could create problem in later, since the classification model needs to have enough data form each class to predict correctly.



Algorithms and Techniques

In order to complete, I work with two sections. First one is preprocessing the data. To convert text files to vectors I used Count vectorization then used Tfidf transformer. The following parameters could be used to tune;

- CountVerctorizer:
    - input
    - encoding
    - decode error
    - strip accents
    - analyzer
    - preprocessor
    - tokenize
    - ngram_range
    - stop words
    - lowercase
    - token pattern
    - max_df
    - min_df
    - max_features
    - vocabulary
    - binary
    - dtype
- TfidfTransformer:
    - Norm
    - Use_idf
    - Smooth_idf
    - Sublinear_tf

Secondly, I check couple of classifying algorithms. The algorithms are;

- Gaussian Naïve Bayes from sklearn
- Logistic Regression from sklearn
- Adaboost Classifier from sklearn
- Gradient Boosting Algorithm from sklearn
- XGBClassifier from xgboost

After checking these algorithms, I decided to work with sklearn's Gradient Boosting Algorithm since the highest accuracy comes from this.

Gradient Boosting builds an additive model in a forward stage-wise fashion; it allows for the optimization of arbitrary differentiable loss functions. In each stage n_classes_ regression trees are fit on the negative gradient of the binomial or multinomial deviance loss function. After choosing algorithm, to optimize the model, GridSearch is used. The parameters can be used to optimize the algorithm;

- Loss
- Learning rate
- Number of estimators
- Max depth
- Max features
- Criterion
- Min samples splot
- Min samples leaf
- Min weight fraction leaf
- Subsample
- max leaf nodes
- min impurity split
- init
- verbose
- warm start
- random state
- presort

Benchmark

I couldn't find a similar project that is done, so my goal will be,

- create %80 accuracy
- report the predict probability of the predictions

III. Methodology

Data Preprocessing

Because I am dealing with text data, I had to preprocess it. For the text data, I used Count vectorization to create vectors. What it does is basically creats matrix by counting tokens or words. I work with 100 words. Then to get more usefull information form document I used Tfidf transformer, which find out term frequency of inverse document frequency. Also for other two columns, I convert them to numeric variable by Label Encoder.

Implementation

I follow this steps after preprocessing;

1. Find the highest accuracy score among the learning algorithms that noted in the previous page. I used accuracy as my score metric.
2. Grid search used to tune the algorithms
    - I used the function with total of 5 cross validation with following parameters;
        - Learning_rate
        - N_estimators
        - Max_features
3. Return the output, first I wanted to return the class because I get low accuracy, I wanted to change the system to probability of classes.

Refinement

As mentioned in previous section, I use Grid search to improve my model. The para metes check following numbers,

- Learning rage:
    - 0.001
    - 0.01
    - 0.1
- n_estimators:
    - 100
    - 200
    - 400
- max_features:
    - 'sqrt'
    - 'log2'
    - None

The best result we got is 0.6365 with learning rage of 0.1, n_estimators of 400, and ma x_features of 'log2'. The improvement I made was 0.0012 which is quite low.

IV. Results

Model Evaluation and Validation

During development, I create two separate sets with sklearn's train_text_split function to evaluate the model. In first step, I try 5 different models and find the following accuracy scores;

Gaussian Naïve Bayes	0.4055
Logistic Regression	0.4945
Adaboost Classifier	0.3898
Gradient Boosting Algorithm	0.6353
XGBClassifier	0.6341

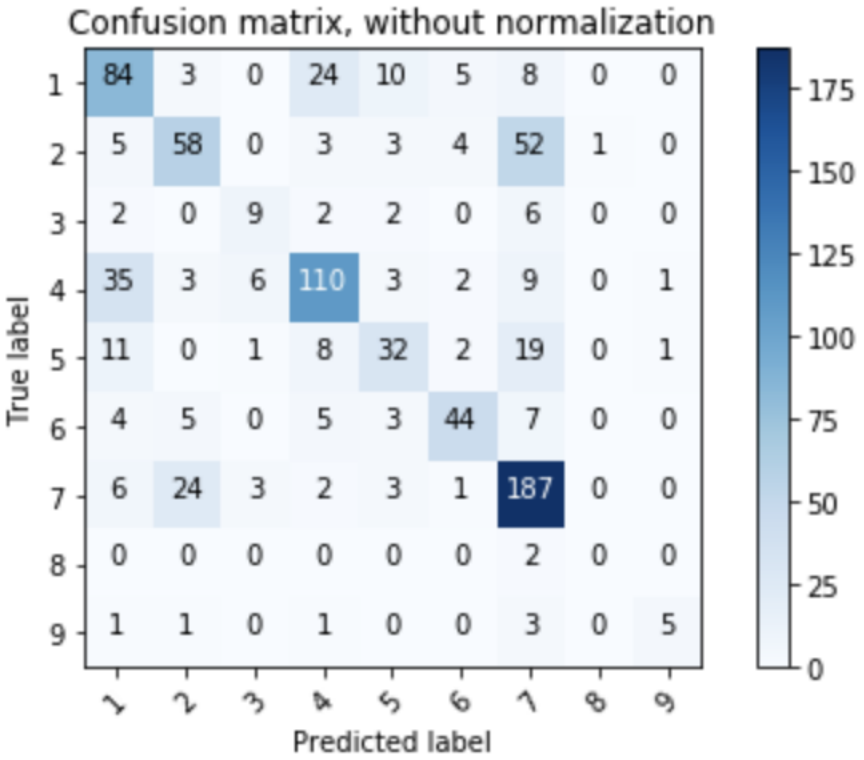
I prefer to go with Gradient Boosting algorithms since it got the highest accuracy score. I was expecting higher accuracy score but unfortunately it seems hard with this process, at this point I would like to do one of this two things. First one is Different Approach: if I can create different processing and create new features it might get a better outcome. but this process would take more time than other and output is not guaranteed. Second one would be change system outcome: in first approach, what I want to do is, create algorithm that can return the class. but since we are getting low accuracy I want to change the output to be probability of being any all classes. Since first one is not promising any better output, I decided to go with second option.

Justification

Unfortunately, I could not reach my benchmark result. In this case, the model predicts only around %60 times correctly. So, model is not very trustable. But it could be helpful with just returning predict probability as some kind a recommender or as another opinion for experts to consider.

V. Conclusion

Free-Form Visualization



this figure shows the confusion matrix which shows how many did we predict correctly and how many did we miss predict. As we can see for classes 8 and 9 we miss more than correct ones. Other than that, I have a lot of miss predicted ones on other classes. So, I might need more data or better cleaning algorithms.

Reflection

The process used for this project can be summarized using the following steps:

1. I found an initial problem and relevant, public data sets were found
2. The data was downloaded and preprocessed
3. A bench mark was created for the classifier
4. The classifier was trained using the data (multiple times, until a good set of parameters were found)

I found step 4 was the hardest and most time-consuming part. Unfortunately, I was not able to find the interesting aspects. I used general solution for this problem and find out it will be needing more than traditional text classification methods.

Improvement

In my opinion, the word2vec or doc2vec models might be helpful to increase the accuracy. Also, we need to create preprocessing and cleaning section better to get more useful features.

References

https://en.wikipedia.org/wiki/Natural_language_processing
https://www.kaggle.com/c/msk-redefining-cancer-treatment
http://scikit-learn.org/stable/python
http://www.nltk.org
https://www.tensorflow.org/tutorials/word2vec
https://radimrehurek.com/gensim/models/word2vec.html