Machine Learning Engineer Nanodegree

Capstone Project

Hatem Kamal
Aug 25th, 2021

I. Definition

Project Overview

Last year in college we had a project to make the diagnosis system for a hospital this project was about the database and the system of the hospital only. But as an improvement I thought about adding what I've learnt here to the system where the user adds the symptoms and get an output of what disease he might have.

So this is a classification problem where there's some input and the output is the corresponding disease type.

For this problem we will need a punch of dataset of diseases with their symptoms provided the data is preprocessed before it's fed to the machine learning model for training.

The data set is provided in this link:

https://www.kaggle.com/itachi9604/disease-symptom-description-dataset?select=dataset.csv

Problem Statement

Before the patient goes to the doctor we need to provide an initial point of what disease the patient might have. Also if the patient wants to see what disease he might have from the website.

I got some data of diseases with their symptoms from a kaggle dataset that needed to be trained by a model to get the disease.

This is the first step of the project then this data needed to be explored and preprocessed before being used. Then split the data to train and test sets. Then train the model and test it if. It's good then we deploy it to be used in production.

Metrics

The distribution of the diseases in the data is good from 5:10 samples per disease so the accuracy score was selected as an evaluation metrics. So about 80% accuracy for the data provided might give good results.

II. Analysis

Data Exploration

As described above the data has the diseases and their symptoms per each row, also each disease is repeated more than 1 time with possible symptom for the disease. And as the number of symptoms isn't fixed we have some null values in the original data as in this figure:

```
In [93]: df = pd.read_csv('data.csv')
         df.head()
```

Out[93]:

	Unnamed: 0	Disease	Symptom_1	Symptom_2	Symptom_3	Symptom_4	Symptom_5	Symptom_6	Symptom_7
0	0	Fungal infection	itching	skin_rash	nodal_skin_eruptions	dischromic _patches	NaN	NaN	NaN
1	1	Fungal infection	skin_rash	nodal_skin_eruptions	dischromic _patches	NaN	NaN	NaN	NaN
2	2	Fungal infection	itching	nodal_skin_eruptions	dischromic _patches	NaN	NaN	NaN	NaN
3	3	Fungal infection	itching	skin_rash	dischromic _patches	NaN	NaN	NaN	NaN
4	4	Fungal infection	itching	skin_rash	nodal_skin_eruptions	NaN	NaN	NaN	NaN

The diseases and the symptoms are provided in string form. And each column has a number of non-null data as in this figure:

```
df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 304 entries, 0 to 303
Data columns (total 18 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   Disease     304 non-null    object
 1   Symptom_1   304 non-null    object
 2   Symptom_2   304 non-null    object
 3   Symptom_3   304 non-null    object
 4   Symptom_4   272 non-null    object
 5   Symptom_5   234 non-null    object
 6   Symptom_6   186 non-null    object
 7   Symptom_7   158 non-null    object
 8   Symptom_8   140 non-null    object
 9   Symptom_9   120 non-null    object
 10  Symptom_10  110 non-null    object
 11  Symptom_11  68 non-null     object
 12  Symptom_12  47 non-null     object
 13  Symptom_13  30 non-null     object
 14  Symptom_14  19 non-null     object
 15  Symptom_15  18 non-null     object
 16  Symptom_16  10 non-null     object
 17  Symptom_17  1 non-null      object
dtypes: object(18)
memory usage: 42.9+ KB
```

So as stated in the figure above we have 304 samples of data each sample has 3 symptoms at least and the most number of symptoms per sample was 17.

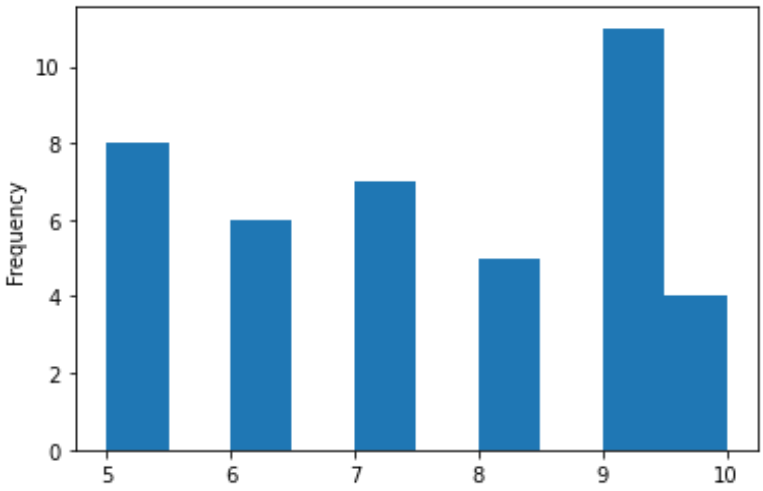
It was provided in the documentation of the dataset that it has 41 different diseases and 131 symptoms.

The structure of the data needs to be modified before being fed to any machine learning model as in this structure the input isn't fixed length and contains a lot of null values.

Exploratory Visualization

To visualize how the data provided is distributed we plotted those figures to get how the diseases and symptoms is distributed across the data

Distribution of diseases

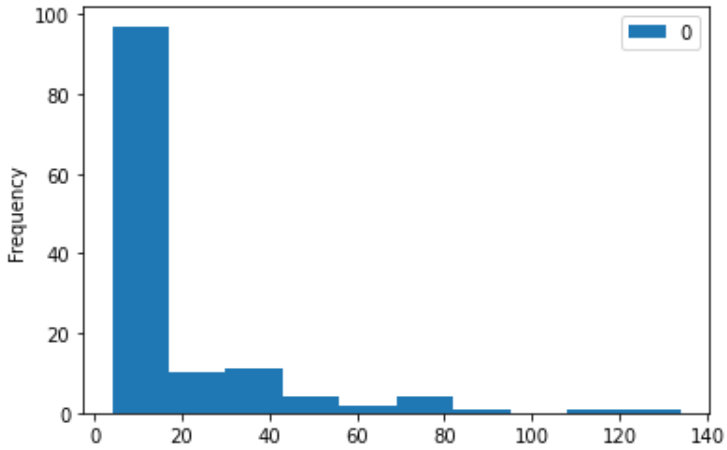


Number of Diseases

As we see here the number of diseases that has 10 samples is 4 diseases, 9 samples is 11 diseases, 8 samples is 5 diseases, 7 samples is 7 diseases, 6 samples is 6 diseases and 5 samples is 8 diseases.

We did the same for the symptoms and got these results and as we see that most of the symptoms is repeated for about 4:12 in the data and some repeated about 130 times across the data. And the 5 most repeated symptoms are: ' fatigue': 134, ' vomiting': 117, ' high fever': 89, ' loss of appetite': 81, ' nausea': 77.

Distribution of symptoms



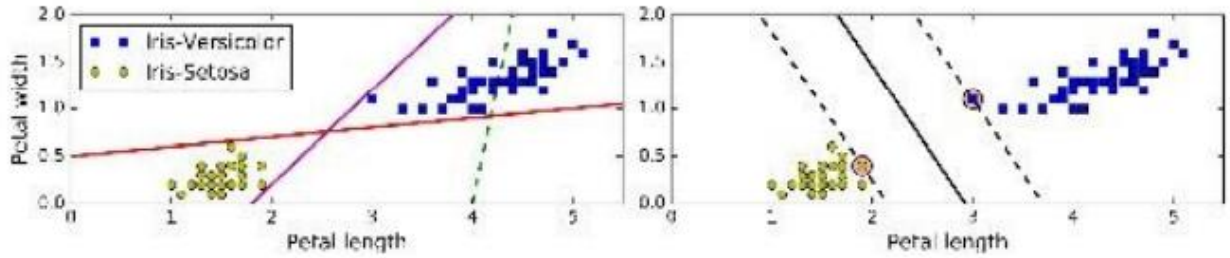
Number of symptoms

Algorithms and Techniques

To feed the data to the machine learning model we can't use the data as it is right now. One hot encoding is a better way to feed the model with data as it has fixed length for all the data provided. That is instead of data to be as described it will be like this:

itching	skin_rash	continuous_sneezing	shivering	stomach_pain	acidity	vomiting	indigestion	muscle_wasting	patches_in_throat	...
0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	...

For the model as it's a classification process so I used the multivalued linear Support vector machine algorithm with one versus one decision function shape, A Support Vector Machine (SVM) is a very powerful Learning model, capable of performing linear or nonlinear classification, regression, and even outlier detection. SVMs are particularly well suited for classification of complex but small- or medium-sized datasets. Where instead of having just decision boundary dividing the data it decides the max margin from the boundary and the closest sample from each class.



Notice the difference between the separation in the right figure and the left one, notice how there's a margin between the boundary and the data. And notice how it's separable by straight line.

In the left plot we see the decision boundaries of three different linear classifiers. The dashed line model is so bad that it does not even separate the classes properly. The other two models with solid lines work perfectly on this training set, but their decision boundaries come so close to the instances that these models will probably not perform as well on new instances.

So the solid line in the right shows the decision boundary of an SVM classifier; not only separates the two classes but also stays as far away from the closest training instances as possible. You can think of an SVM classifier as fitting the widest possible street (represented by the parallel dashed lines) between the classes. This is called large margin classification.

Benchmark

To test the performance of the model I will use the accuracy score as the data is balanced and it should be at least 80% accuracy on the test set.

III. Methodology

Data Preprocessing

First we need to clean the data we had a column with no meaningful name and it's just a duplicate of the data index as in this figure. So we dropped it from our data.

	Unnamed: 0
0	0
1	1
2	2
3	3
4	4

After checking for duplicated rows we didn't find any duplicates so we continue the process with the one hot encoding. I made use of the masking property of pandas Data frame to solve this proplem. The solution is as follows:

- Get all the unique values of the column.
- Loop through them and use the masking to get in what row they are and put the value of 1 in the column with that name. if the column isn't in the data yet pandas auto adds it and add this value and put null in other rows.
- This process is repeated for each column in the original dataset.
- After finishing the process, we fill the null values with 0.
- Then we drop the original columns as they're not needed for the model
- Then we divided the data into training and test set in 9:1 ratio.

Implementation

As described in the preprocessing section the data is one encoded before being fed to the model for training. We loop through each unique value in each column and add 1 to its corresponding row and column. Then we fill the null values with 0.

Then we test this process with looping through each row and loop through the original data and see if the symptom written in the original data has its value as 1 in the corresponding row and column if not we print this row index.

Then we shuffle our data and split it to train and test sets.

We save the data locally to upload it next to the s3.

We define our sklearn estimator using ml.m5.xlarge training instant and fit our data to the linear svc model.

The hyper parameters for the model is C=1 (this hyper parameter control how the margin is the lower C leads to wider margin) and using the linear kernel with one vs one decision function shape.

When implementing the function, you have to write it like this:

Model = OneVsRestClassifier(SVC(C=1, kernel="linear", decision_function_shape='ovo'))

As the SVC library in the sklearn doesn't support the multicolumn output or multiclass classification.

Then we tested our model deploying it on ml.t2.medium deploying instance. And test it on the test set getting accuracy score of 1.

So we used this model for our final deployment and wrote the lambda function dealing with it. The lambda function is as follows.

In the lambda handler we: define our symptoms with their order and the diseases with their order. Then we read our event body data which we made to be a comma separated values but in string form.

Then we create a list of 131 zeros, mapping the values from the event to the symptom and get the index.

Then we change the value of that index in the list to 1. We made another list of 131 zeros and fed them to our model as it accepts only 2d data.

After getting the values of the response we decode it and find all integers in it and getting the first 41 if there's 1 on theme we get its index and get the disease corresponding to that index and return it with the return function.

Refinement

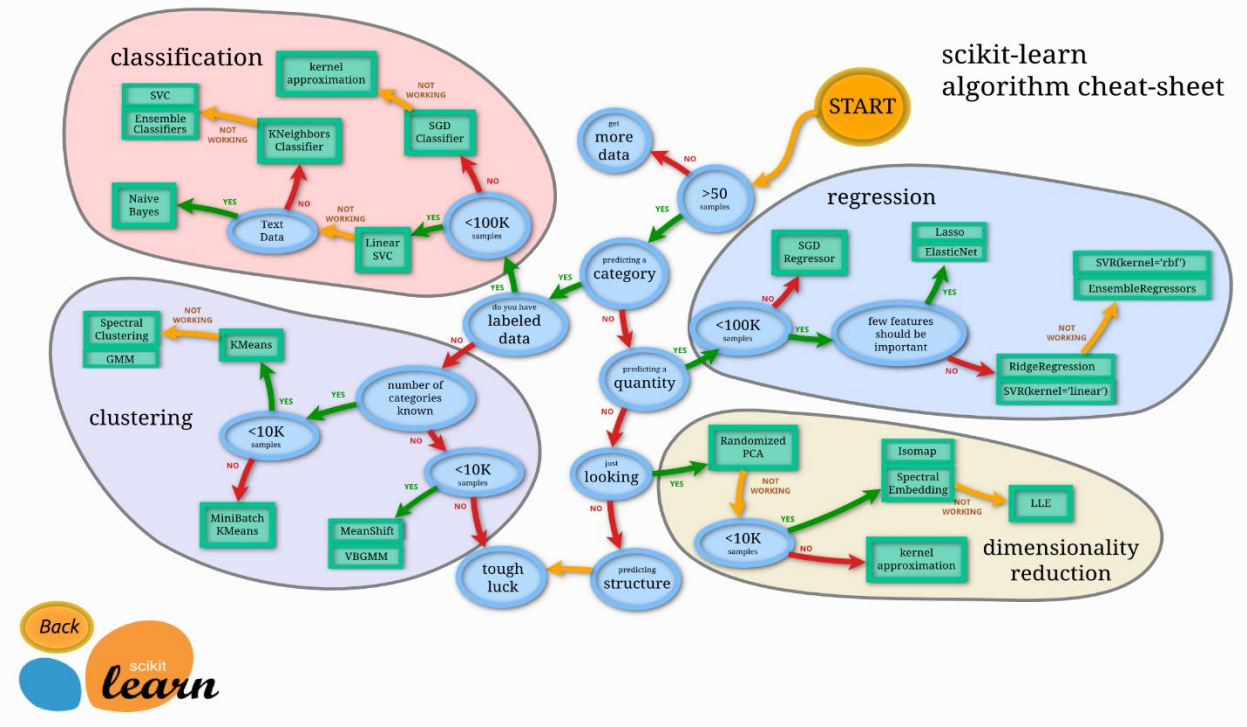
The logic I used for the preprocessing wasn't quite well first as I firstly create new column and put create columns with their values but in the loop this process overwrites a previously preprocessed data this leaded to poor evaluation for the model.

I used the method described above instead where each value is written as 1 and the we fill the nulls with 0s

IV. Results

Model Evaluation and Validation

Going through the sklearn tutorial guide in the picture below as the problem was a classification with labeled data we used the svc linear model.



The hyper parameters for the model is C=1 (this hyper parameter control how the margin is the lower C leads to wider margin) and using the linear kernel with one vs one decision function shape.

When implementing the function, you have to write it like this:

OneVsRestClassifier(SVC(C=1, kernel="linear", decision_function_shape='ovo'))

As the SVC library in the sklearn doesn't support the multicolumn output or multiclass classification.

I used the accuracy score to test its performance in the test set and got accuracy score of 1 which is great.

As described the model was tested on the test set made from the preprocessed data.

When I faced the preprocessing issue described above the data really affected the results of the accuracy score.

Assuming the one who input the data input them correctly it can be trusted for about 80% as it just gives an initial guess and needs to be followed up by a doctor.

Justification

The final result was a lot better than the stated in benchmark either it's accuracy score or f1 score as it was 1 in the accuracy score.

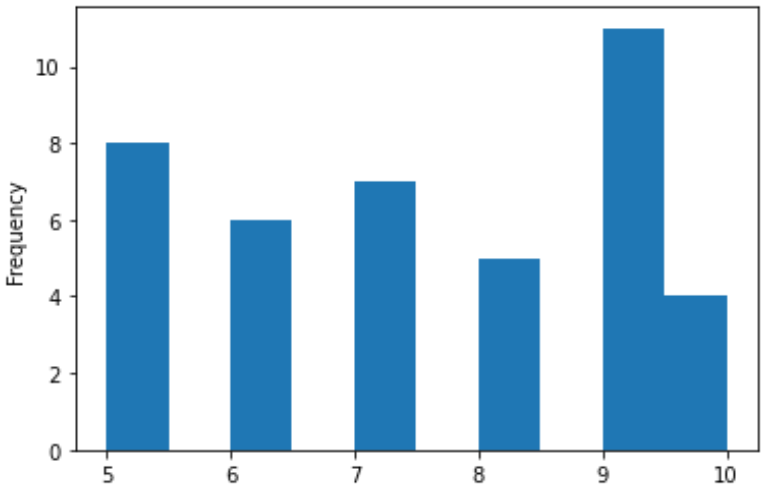
The final result is good for the data provided may be getting more data about the topic be better as it wasn't tested under a real patient symptom yet.

V. Conclusion

Free-Form Visualization

To visualize how the data provided is distributed we plotted those figures to get how the diseases and symptoms is distributed across the data

Distribution of diseases



Number of Diseases

As we see here the number of diseases that has 10 samples is 4 diseases, 9 samples is 11 diseases, 8 samples is 5 diseases, 7 samples is 7 diseases, 6 samples is 6 diseases and 5 samples is 8 diseases.

Reflection

Now we have our project done we've gone through a lot first gathering and getting the data and prepare the front end to send the data to API when created.

Then we preprocessed the data using one hot encoding and we have to emphasize the importance of writing a robust test for it as it's the most crucial part in the entire project.

Then we use the AWS sklearn estimator to be trained on the training data and deploy it to test the model.

We made the lambda function where we get the data and clean it before predicting the results from the user input and return his results.

The process of preprocessing and writing the lambda function was really interesting and challenging.

This model can be tested in some symptoms from real patients first before being used in production.

Improvement

May be using a decision tree model and instead of the user input the symptoms from a long list of symptoms he gets to choose from smaller amount and base on what he chooses he gets a list of other symptoms and so on tell he get his result.

Getting more data about the topic would really be a lot better if we compared it to the final results that we got.

References

- Hands on machine learning with scikit learn and tensorflow