Bharadwaj Vaduguru

Dr. Ramakrtishna Koganti

CSE 5301-003

05/09/2020

**Corona virus disease prediction using Logistic regression and decision tree classifier.**

**Abstract:**

Coronavirus disease (COVID-19) is an infectious disease that develops mild to moderate respiratory illness. It transmits through droplets of saliva or discharge from the nose when an infected person coughs or sneezes. In this paper, a machine learning model is developed using logistic regression and decision tree classifier to predict if the person is affected by COVID-19 by considering the attributes as age, cough, fever, difficulty in breathing into consideration. As we are working with a topic in medical field, we are indeed focusing to have less False Negative Rate (FN) when compared to False Positive Rate (FPR) i.e., a person who is not affected may be predicted as affected but a person who is affected should never be predicted as unaffected. Precision, recall, f1 score are used to find the accuracy of the model. In the entire data set, we will be using 4/5th of data is used as training data and the rest 1/5th is used as testing data.

**Introduction:**

COVID-19 pandemic is an ongoing serious concern for the world. People who got affected by the pandemic are increasing exponentially day by day. It being contagious, can be identified with symptoms such as cough, difficulty in breathing, fever, fatigue, body pains. The advantages of incorporating data analysis and modelling techniques in clinical research have been shown to improve diagnosis precision, reduce costs and save human resources. In this paper, logical regression and decision tree models are employed to find the probability of a person getting affected by this disease by taking his symptoms into consideration. It being in the field of medical sciences, precision and accuracy are considered as key factors.

**Problem and Data Processing:**

As discussed earlier, medical field requires more accuracy and precision. The data set used in this paper is excerpted from the UCI Machine Learning Repository, devpyjp.com. We have used Decision tree model, logistic regression (Binary classification).

A confusion matrix is plotted for the data set. A confusion matrix will be as below:

|  |  |  |  |
| --- | --- | --- | --- |
|  | | Actual value | |
| Predicted Value |  | 1 | 0 |
| 1 | True Positive rate | False Positive rate |
| 0 | False Negative rate | False Negative rate |

Type 1 error: False Positive rate

False positive rate= (False Positive)/(False Positive +True Negative)

Type 2 error: False Negative rate

We aim at reducing Type1 and Type 2 error. It being in the field of medical science, Type 2 error or false negative rate tells us about of all positive values, how many did we predicted wrongly so it is to be reduced to the best as we can because a patient who is affected should never be predicted as an healthy patient.

Cross validation helps us to improve the accuracy of the model, here we do Stratified K fold cross validation, this is done by splitting the data set with respect of target of class labels ratio. So that every time we get an equal ration of data points of each class.

A logistic regression model is used and cross validation score is calculated.

**Logistic Regression:**

It is a classification problem. It is used when the target variable is categorical or dichotomous(binary). It is used to describe the data and to explain the relationship between one dependent variable and more ordinal attributes.

A Decision tree model is also generated for the data and cross validation score is calculated.

**Decision tree:**

It is a decision enabling tool that uses tree-like graph or model of decisions, resource costs and utility if the target variable is categorical, it is a kind of supervised learning. They are also known as CART(Classification and Regression Trees). Entropy is given by “S”.

S= -p+log(p+)-p-log(p-)

p+:Proportion of positive examples in S

p−: Proportion of negative examples in S

The best one is opted based on the data and then precision, f1 score and recall score are calculated for the model that was opted. Heatmaps are plotted for each model using python code.

Recall=(True positive)/(True positive+ False negative)

Precision=(True positive)/(True positive + False positive)

F beta=(1+β2)((precision\*Recall)/ β2\*(Precision+Recall)

Table 1. Features of the Dataset:

**Feature Description:**

|  |  |
| --- | --- |
| People affected with Corona Virus | **3506** |
| People who are not affected | **3598** |
| **People with temperature 105** | **588** |
| **People with temperature 98** | **586** |
| **People with temperature 108** | **567** |
| **People with temperature 97** | **557** |
| **People with temperature 107** | **556** |
| **People with temperature 99** | **541** |
| **People with temperature 106** | **538** |
| **People with temperature 100** | **536** |
| **People with temperature 103** | **533** |
| **People with temperature 101** | **529** |
| **People with temperature 102** | **519** |
| **People with temperature 104** | **510** |
| **People with temperature 96** | **544** |

A Python program is developed to process the data and predict a classifier using the decision tree and logistic regression model.

The process is shown below in Fig. 1:

The Data set(CSV file) is loaded

The target column is separated so that we can process it for cross validation

## We use Stratified k fold cross validation and split the data for training and testing

## Decision Tree model is developed

## Logistic regression model is developed

## Best of the two models is opted based on the precision, recall and f1 score value.

**Results:**

Using a Data set of size 7104 samples, the logistic regression gives an accuracy of 70.5% and decision tree gives an accuracy of 83%. By Hyper parameter tuning, we compare both the models and found that Decision tree as best one for the given data set.

**Conclusion:**

The study aimed at developing a machine learning model that predicts if a person is affected by Corona virus. It being in the field of medical science, we focus on minimizing the Type-2 error or false negative rate. The model is trained by giving data of patients regarding their symptoms. The fever temperature, cough, running nose and difficulty in breath. If a person is having any of these symptoms, it is indicated as “1” and without that symptoms is given by value “0” in Data set. The logistic regression model and decision tree models are used and accuracy is calculated. By doing the hyper parameter tuning, we compare both the models and found the decision as the best one for the Data set taken.

**References:**

1) Data sets from Kaggle-

<https://www.kaggle.com/sudalairajkumar/novel-corona-virus-2019-dataset>

<https://www.kaggle.com/therealcyberlord/coronavirus-covid-19-visualization-prediction>

2) Data from World Health Organization website-

<https://www.who.int/emergencies/diseases/novel-coronavirus-2019>

Appendix-A: Python code:

import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

df=pd.read\_csv('C:\\Users\\vaduguru\\Downloads\\corona\_prediction.csv')

df.head() #To check if data set is loaded successfully, by defaultit will return the first 5 rows.

df.shape #It returns a tuple of number of rows and number of columns, (rows, columns)

df.info() #Gives the name of the columns

df.infection\_Probability.value\_counts() #To check if the given Data set is balanced?

df.Fever.value\_counts()

condition\_1=(df['Fever']>100)&(df['BodyPains']==1)&(df['RunnyNose']==1)&(df['Difficulty\_in\_Breath'] == 1)

df['infection\_Probability'][condition\_1]=1

df[condition\_1]

condition\_2=(df['Age']>=60)&(df['Fever']>99)&(df['RunnyNose']==1)&(df['Difficulty\_in\_Breath'])|(df['BodyPains']==1)

df['infection\_Probability'][condition\_2]=1

df[condition\_2]

condition\_3=(df['Fever']>99)&(df['BodyPains']==0)&(df['RunnyNose']==0)&(df['Difficulty\_in\_Breath'] == 0)

df['infection\_Probability'][condition\_3]=1

df[condition\_3]

X = df.drop('infection\_Probability',axis=1).values #Removing the target column so that it contains whole Data set except Target columns

Y = df['infection\_Probability'].values #The target columns are copied here

from sklearn.model\_selection import StratifiedKFold,KFold,cross\_val\_score,ShuffleSplit,GridSearchCV

cv = StratifiedKFold(n\_splits=6,random\_state=11)

kf = KFold(n\_splits=6,random\_state=100)

for train\_index, test\_index in cv.split(X,Y):

#print("TRAIN:", train\_index, "TEST:", test\_index)

X\_train, X\_test = X[train\_index], X[test\_index]

y\_train, y\_test = Y[train\_index], Y[test\_index]

from sklearn.linear\_model import LogisticRegression

from sklearn.tree import DecisionTreeClassifier

from sklearn.metrics import accuracy\_score,confusion\_matrix

lr = LogisticRegression(class\_weight='balanced')

lr.fit(X\_train,y\_train)

lr\_pred = lr.predict(X\_test)

lr\_acc = accuracy\_score(y\_test,lr\_pred)

lr\_acc

cross\_val\_score(lr,X,Y,cv=3)

lr\_cm = confusion\_matrix(y\_test,lr\_pred)

lr\_cm

lr\_df = pd.DataFrame(data=lr\_cm,columns=['0','1'],index=['0','1'])

lr\_df

ax=sns.heatmap(lr\_df,annot=True)

bottom, top = ax.get\_ylim()

ax.set\_ylim(bottom+0.5, top-0.5)

plt.show()

#Decision Tree

dt=DecisionTreeClassifier(class\_weight='balanced')

dt.fit(X\_train,y\_train)

dt\_pred = dt.predict(X\_test)

dr\_acc = accuracy\_score(y\_test,dt\_pred)

dr\_acc

cross\_val\_score(dt,X,Y,cv=cv)

dt\_cm = confusion\_matrix(y\_test,dt\_pred)

dt\_cm

dt\_df = pd.DataFrame(data=dt\_cm,columns=['0','1'],index=['0','1'])

dt\_df

ax=sns.heatmap(dt\_df,annot=True)

bottom, top = ax.get\_ylim()

ax.set\_ylim(bottom+0.5, top-0.5)

plt.show()

#Choosing the apt model

def find\_best\_model(X,Y):

algos = {

'logistic\_reg':{

'model':LogisticRegression(class\_weight='balanced'),

'params' :{

'penalty':['l1','l2'],

'C':[0.0001,0.001,0.01,0.1,1.0,10,100,1000]

}

},

'DT\_clf':{

'model':DecisionTreeClassifier(),

'params':{

'criterion':['gini', 'entropy'],

'max\_depth': [2,4,6,8,12]

}

}

}

scores =[]

cv = ShuffleSplit(n\_splits=5,test\_size=0.2,random\_state=567)

for algo\_name,config in algos.items():

gd = GridSearchCV(config['model'],param\_grid=config['params'],cv=cv,return\_train\_score=False)

gd.fit(X,Y)

scores.append({

'model':algo\_name,

'best\_score':gd.best\_score\_,

'best\_params':gd.best\_params\_

})

return pd.DataFrame(scores,columns=['model','best\_score','best\_params'])

find\_best\_model(X\_train,y\_train)

dt1\_clf = DecisionTreeClassifier(criterion='entropy',max\_depth=6,class\_weight='balanced')

dt1\_clf.fit(X\_train,y\_train)

y\_pred = dt1\_clf.predict(X\_test)

accuracy\_score(y\_test,y\_pred)

tr\_cm = confusion\_matrix(y\_test,y\_pred)

tr\_cm

tr\_df = pd.DataFrame(data=tr\_cm,columns=['0','1'],index=['0','1'])

tr\_df

ax=sns.heatmap(tr\_df,annot=True,cbar=False)

bottom, top = ax.get\_ylim()

ax.set\_ylim(bottom+0.5, top-0.5)

plt.show()

from sklearn.metrics import f1\_score,precision\_score,recall\_score

f1\_score(y\_test,y\_pred)

precision\_score(y\_test,y\_pred)

recall\_score(y\_test,y\_pred)