PROJECT REPORT

ON

COMPARISON BETWEEN VARIOUS MACHINE LEARNING ALGORITHMS FOR COVID-19 DETECTION

Think big with big data

SUBMITTED BY -

Anuva Agarwal (A20494057)
Vismaya M (A20519405)
Abhishek Reddy De Reddy (A20501400)

Contents

I. Overview

In this project, we will explore the different machine learning algorithms like Logistic Regression, KNN, Neural Network, Random Forest, and Decision Tree and perform a comparison in terms of performance, speed and accuracy on the Covid-19 dataset. As TensorFlow offers greater accuracy, we have also added some observations about utilizing it.As an additional technique we have also built a pipeline using tpot classifier as it can automatically pick the method for best accuracy. Going forward we are going to refer to a dataset provided by the World Health Organization (WHO).

Note *- After assessing our models, you might find the predictions to be a bit harsh. But it's not like we can sugar-coat it.*

II. Introduction

The main idea of the project is to show which among the Logistic Regression, KNN, Neural Network, Random Forest, and Decision Tree is a better classifier in the machine learning set, and which performs the best on which type of data. We have also tried to include some observations upon using TensorFlow as it provides a better accuracy. The objective is to predict whether a patient has Covid-19 or not using these three Machine Learning models by factoring in independent variables such as Fever, Tiredness, Dry-Cough, Breathing Difficulty, Sore Throat, Symptoms or No symptoms, Pains, Nasal Congestion, Runny Nose, Age, Gender, Severity, Country.

The dataset we are planning to use for this project is a dataset available on Kaggle and provided by the World Health Organization (WHO). CDC reports national COVID-19 case surveillance data to the World Health Organization, as required under International Health Regulations and the data has been collected every year since January 2022. Our dataset contains about 316,801 records and we are choosing to go with this particular dataset because it has fewer missing values, and it contains only the related 27 features as compared to the other one which has a lot of irrelevant fields. For example - we would identify a person with fever with the value 1 and a 0 if they have no fever in our dataset. An Important aspect of this dataset is that we have both the physical health condition of the person like Fever, Tiredness, Pains and other symptoms of Covid-19 and the factors that mainly affect the spread of Covid-19 virus through contact from that patient. Here is the link to the dataset.

We chose this particular topic because the COVID-19 pandemic was declared a public health emergency of international concern by WHO on 30 January 2020 and a pandemic on 11 March 2020 and still continues to affect large parts of the population worldwide. Numerous vaccinations have been developed by US, India and other countries and distributed for free with health centers opened around the corner that can be accessed anytime but the WHO has estimated that over 100 million Americans have been infected by Coronavirus disease and over 1.09 million Americans have died of the 335 million approximate adults in the country-more than 1 in 3 have it.

We all know somebody who's struggling with Coronavirus disease, and we all know that this might have been avoided if only they had known about it and made the decision to take action. We are aware that, especially if there are no symptoms at this time, it is somewhat unrealistic to expect individuals to take time out of their busy schedules to visit a doctor for a routine check-up. However, it is realistic to anticipate that people will take a brief 5-minute health survey and utilize machine learning algorithms to determine whether they are at a high risk of the disease. We could convey our findings to them, which can literally save their life. We do understand that this is not an alternative for a visit to the Doctor, but given the ease of use, and cost being almost zero, this could be the next best thing.

These machine learning algorithms help in tracking the various causes responsible for the disease which might even change depending on the weather conditions and provide more insight to take a proactive approach in combating the disease for future reference. Pairwise correlations between predictor variables showed that wealthier economies had larger tourism industries and older populations, and that urbanization was greater in wealthier countries. We can also use ML to show correlation between different features, such as correlation between People Age and Severity of the disease, which could help users identify critical causes and make better life choices.

In our project we aim to take a deep dive into the following questions:

1. Can we accurately predict whether an individual has Covid-19 using the K-NN, Random Forest, and Decision Tree model?

2. Which of these 3 models is most accurate in its prediction for the two given datasets?

3. How to use python libraries to visualize which factors are closely correlated to each other?

4. Can we use just a subset of the features, thereby making the survey shorter, and still accurately predict whether an individual has Covid-19?

III Work Environment

For our project we have worked on the Google Colab platform by using the TensorFlow as it has a wide range of advantages over many environments

1) It has many pre-installed Libraries and does not require installation every time we work on it.

2) Everything we work on is stored on the cloud i.e., it is usually mounted through the google drive and can be easily operated on these sizes of datasets which are comparatively not so large from anywhere and at any time.

3) Multiple accesses on the same data and codes are possible i.e., it is easy to work and share code when working in a group and is a very user-friendly python-based platform.

4) This environment also provides us with free of cost GPUs and CPUs to work with as per our requirements.

5) Basically, TensorFlow is a Deep learning framework whereas MLlib is a machine learning framework.

Since we are working on deep learning frameworks i.e., using the neural networks we need GPU which is offered by the Google Colab free of cost as working on simple CPU would lead to running the code for comparatively longer times (speaking in terms of hours or even days). Furthermore, as we have different course structures and bad weather, it makes it easier to contribute in a WFH environment when compared with tools like Jupyter Notebook.

IV. Importing Libraries and Dataset

Dataset

```
[58] import numpy as np # linear algebra
     import pandas as pd
     import seaborn as sns
     import matplotlib.pyplot as plt
     from sklearn import preprocessing
     from sklearn.model_selection import train_test_split
     from sklearn.linear_model import LogisticRegression
     from sklearn.neighbors import KNeighborsClassifier
     from sklearn.tree import DecisionTreeClassifier
     from sklearn.ensemble import RandomForestClassifier
     import tensorflow
     from tpot import TPOTClassifier
     from sklearn.model_selection import train_test_split
     from sklearn.metrics import classification_report, accuracy_score
     from sklearn.preprocessing import Normalizer
     from keras.layers import Activation, Dense, Dropout, BatchNormalization, Input
     from keras.models import Model
     from tensorflow.keras.optimizers import Adam
     from keras.callbacks import ReduceLROnPlateau, EarlyStopping
     %matplotlib inline
     plt.style.use('fivethirtyeight')
```

The mount drive command will load the dataset from the local drive if the dataset is already uploaded

```
[3] from google.colab import drive
    drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

Now, we will convert the data in Excel file into a dataframe as we have many python libraries to support its analysis

```
[4] df = pd.read_csv("/content/drive/MyDrive/Cleaned-Data.csv")
    dataset = df.values
```

The very first step would be to import all the relevant libraries that we are going to use for this project followed by the dataset itself. After that, we imported our dataset using the read_csv function of the pandas library. Here's a quick brief on all the python libraries we have used so far.

- Pandas - Pandas is short for "Python Data Analysis Library". Pandas is the most commonly used library for data manipulation and analysis because it offers data structures and operations for manipulating numerical tables.
- Numpy - Numpy stands for 'Numerical Python' and it's a powerful library used for working with multi-dimensional arrays, linear algebra, and matrices.

- Matplotlib - Matplotlib is an extension of pyplot library and is used for creating static, animated, and interactive visualizations with ease.
- Seaborn - Seaborn is a data visualization library based on Matplotlib and is used to informative graphs and enhance visualizations.
- Train_test_split - This is from the Sklearn model selection library and it is used to split the data into two parts, one for training and the other for testing.
- Accuracy_score - As the name suggests, this helps us figure out the accuracy score of our data model.
- Tensorflow - Tensorflow is a free and open-source symbolic math library based on dataflow and differentiable programming. It can work with a lot of languages like JS, Python etc. so it's very flexible.

Here, we have also used TPOT classifier as it performs intelligent search over machine learning pipelines that contain supervised classification models, preprocessors, feature selection techniques, and any other estimator or transformer that follows the scikit-learn API . The TPOTClassifier will also search over the hyperparameters of all objects in the pipeline.

V. Data Description

```
[5] df.head()
```

	Fever	Tiredness	Dry-Cough	Difficulty-in-Breathing	Sore-Throat	None_Sympton	Pains	Nasal-Congestion	Runny-Nose	Diarrhea	...	Gender_Male	Gender_Transgender	Severity_
0	1	1	1	1	1	0	1	1	1	1	...	1	0	
1	1	1	1	1	1	0	1	1	1	1	...	1	0	
2	1	1	1	1	1	0	1	1	1	1	...	1	0	
3	1	1	1	1	1	0	1	1	1	1	...	1	0	
4	1	1	1	1	1	0	1	1	1	1	...	1	0	

5 rows × 27 columns

```
[6] df.shape
```

```
(316800, 27)
```

```
[6] df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 316800 entries, 0 to 316799
Data columns (total 27 columns):
 #   Column                    Non-Null Count   Dtype
---  ------                    --------------   -----
 0   Fever                     316800 non-null  int64
 1   Tiredness                 316800 non-null  int64
 2   Dry-Cough                 316800 non-null  int64
 3   Difficulty-in-Breathing   316800 non-null  int64
 4   Sore-Throat               316800 non-null  int64
 5   None_Sympton              316800 non-null  int64
 6   Pains                     316800 non-null  int64
 7   Nasal-Congestion          316800 non-null  int64
 8   Runny-Nose                316800 non-null  int64
 9   Diarrhea                  316800 non-null  int64
 10  None_Experiencing         316800 non-null  int64
 11  Age_0-9                   316800 non-null  int64
 12  Age_10-19                 316800 non-null  int64
 13  Age_20-24                 316800 non-null  int64
 14  Age_25-59                 316800 non-null  int64
 15  Age_60+                   316800 non-null  int64
 16  Gender_Female             316800 non-null  int64
 17  Gender_Male               316800 non-null  int64
 18  Gender_Transgender        316800 non-null  int64
 19  Severity_Mild             316800 non-null  int64
 20  Severity_Moderate         316800 non-null  int64
 21  Severity_None             316800 non-null  int64
 22  Severity_Severe           316800 non-null  int64
 23  Contact_Dont-Know         316800 non-null  int64
 24  Contact_No                316800 non-null  int64
 25  Contact_Yes               316800 non-null  int64
 26  Country                   316800 non-null  object
```

```
[7] df.describe()
```

	Fever	Tiredness	Dry-Cough	Difficulty-in-Breathing	Sore-Throat	None_Sympton	Pains	Nasal-Congestion	Runny-Nose	Di
count	316800.000000	316800.000000	316800.000000	316800.000000	316800.000000	316800.000000	316800.000000	316800.000000	316800.000000	316800.
mean	0.312500	0.500000	0.562500	0.500000	0.312500	0.062500	0.363636	0.545455	0.545455	0.
std	0.463513	0.500001	0.496079	0.500001	0.463513	0.242062	0.481046	0.497930	0.497930	0.
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.
25%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.
50%	0.000000	0.500000	1.000000	0.500000	0.000000	0.000000	0.000000	1.000000	1.000000	0.
75%	1.000000	1.000000	1.000000	1.000000	1.000000	0.000000	1.000000	1.000000	1.000000	1.
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.

8 rows × 26 columns

Head function returns top n rows of a DataFrame where n is a user input value. If it's left blank, it assumes the default value to be 5. It’s a good way to get an estimate of how your data looks, since you are able to see all the columns and what kind of values they contain. Shape function returns the dimension of your array, or your dataset that is the number of rows and columns. In our case it returns (316800,27) for database size. Info function prints a concise summary of the data frame such as number of columns, column data types, memory usage etc. The next method used here is the describe method, which computes and displays a table containing statistical data such as mean, median, mode, standard deviation, percentiles etc. Followed by that we have used the isnull function, which basically checks whether there is an empty or null value in our database which is used in combination with sum function, which does nothing but adds up all those null values.

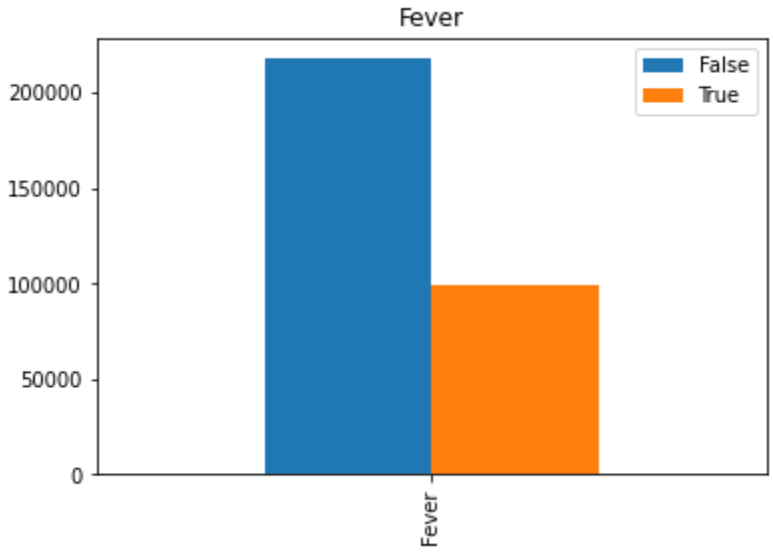
We also have to ensure that we do not have any missing values in our database, we do that by using isnull() command. Here, we did not find any, so it's a good idea to proceed with the current dataframe.

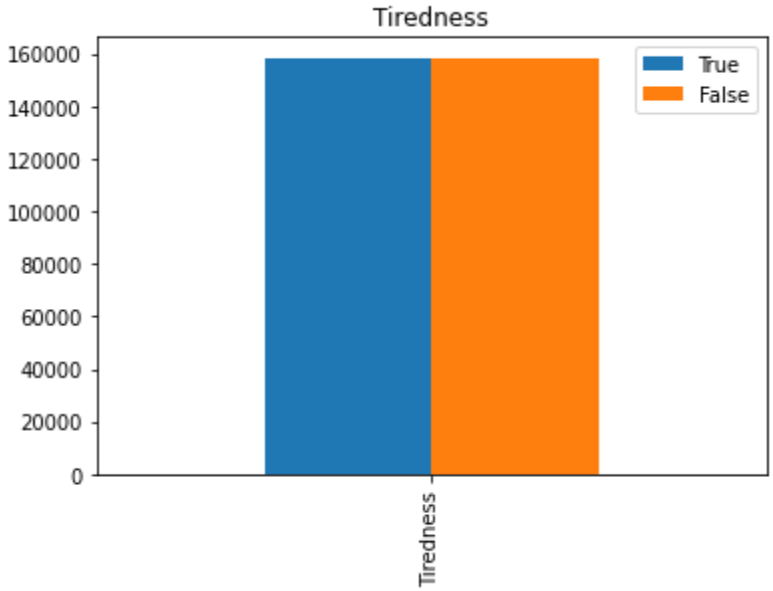
```
[8] df.isnull().sum()
```

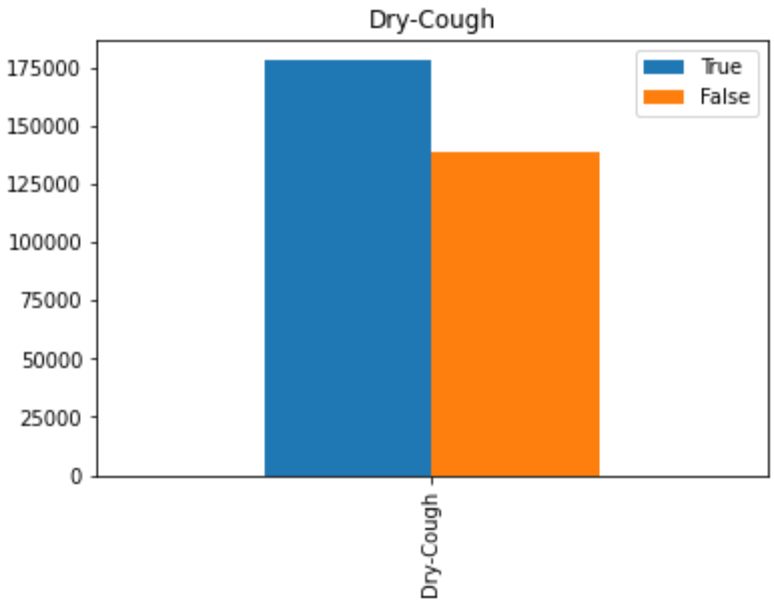
```
Fever                          0
Tiredness                      0
Dry-Cough                      0
Difficulty-in-Breathing        0
Sore-Throat                    0
None_Sympton                   0
Pains                          0
Nasal-Congestion               0
Runny-Nose                     0
Diarrhea                       0
None_Experiencing              0
Age_0-9                        0
Age_10-19                      0
Age_20-24                      0
Age_25-59                      0
Age_60+                        0
Gender_Female                  0
Gender_Male                    0
Gender_Transgender             0
Severity_Mild                  0
Severity_Moderate              0
Severity_None                  0
Severity_Severe                0
Contact_Dont-Know              0
Contact_No                     0
Contact_Yes                    0
Country                        0
dtype: int64
```

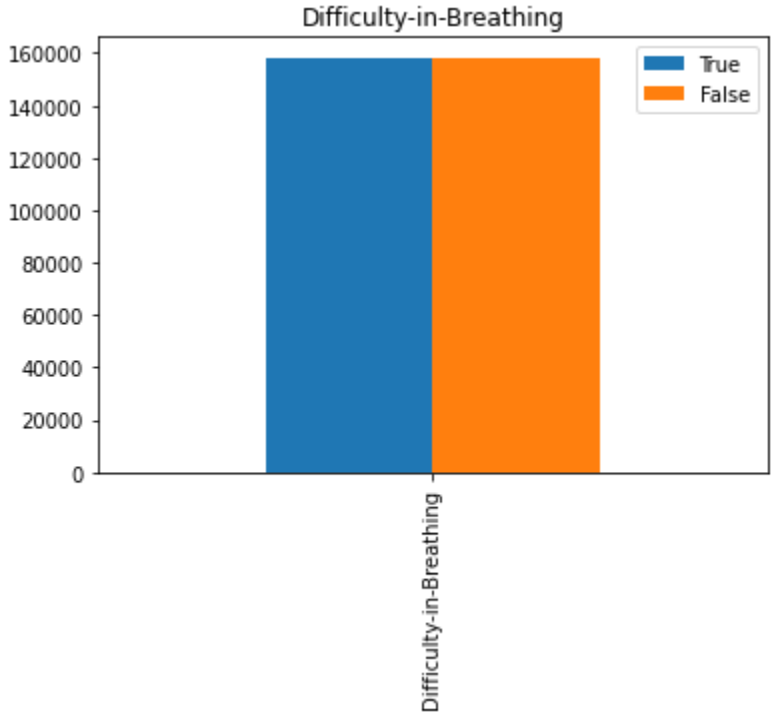
VI. Data Visualization

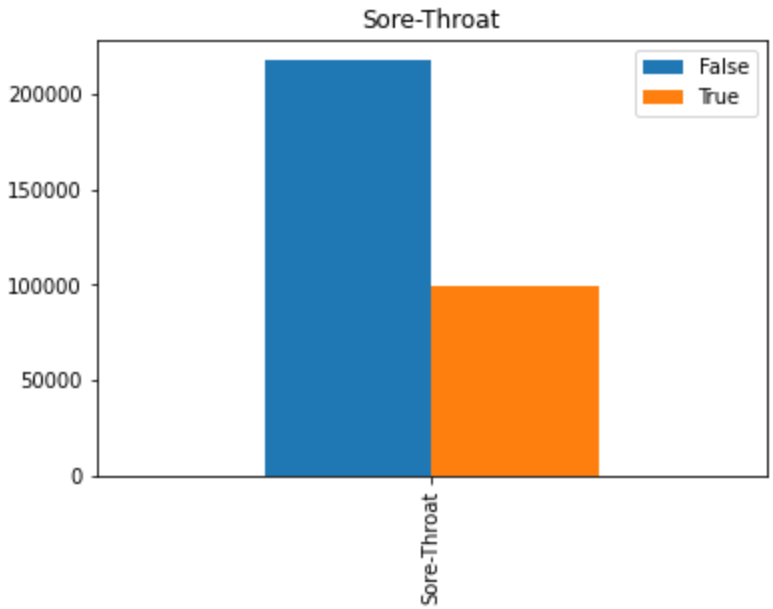
Through making these charts, we can better understand the values in each and every feature of both the datasets and their range simply by having a glance.Here, the visualization is not that clear so we have created bar graph to get a better idea of the values in different columns:

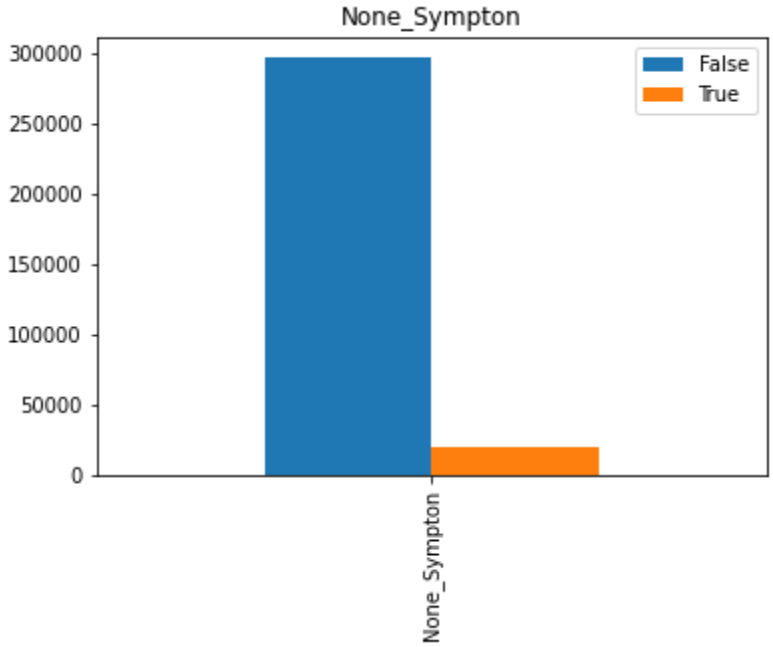

Fever
False
True
200000
150000
100000
50000
0
Fever

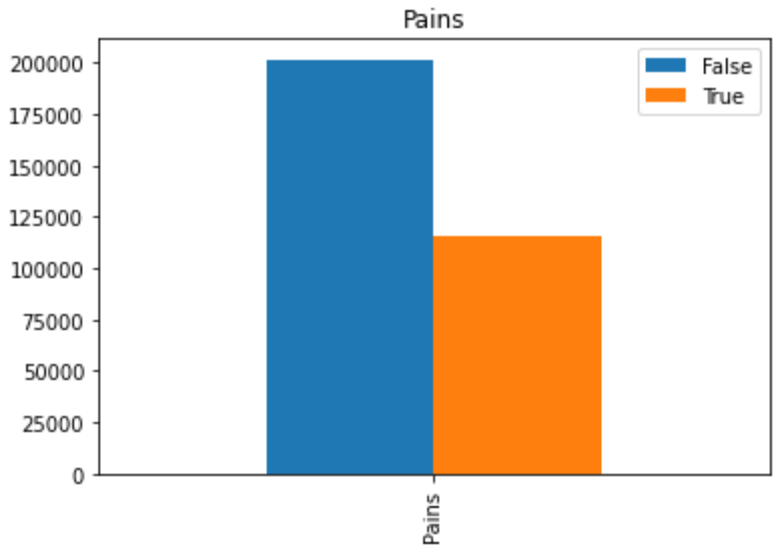

Tiredness
True
False
160000
140000
120000
100000
80000
60000
40000
20000
0
Tiredness

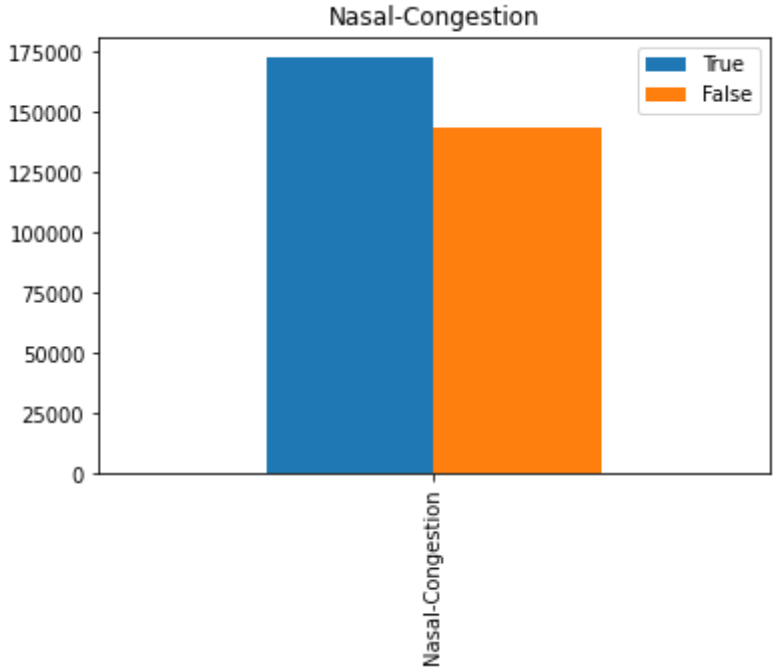

Dry-Cough
True
False
175000
150000
125000
100000
75000
50000
25000
0
Dry-Cough

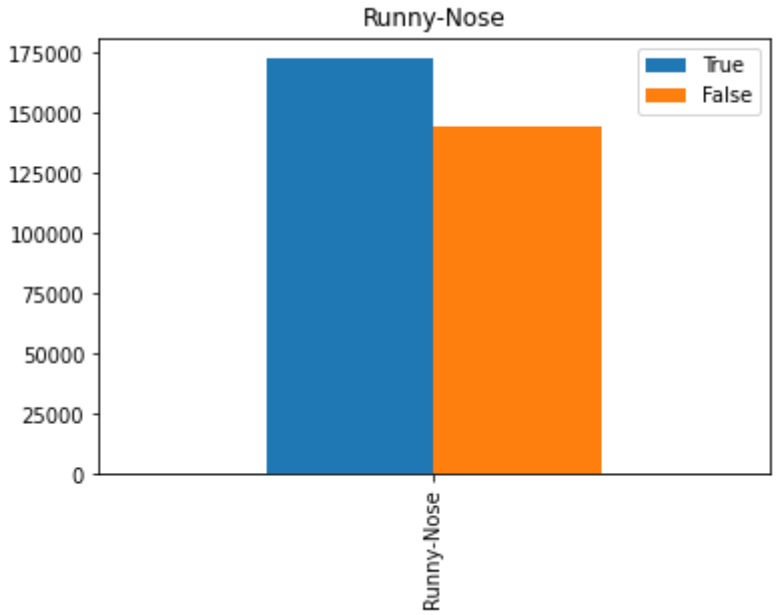

Difficulty-in-Breathing
True
False
160000
140000
120000
100000
80000
60000
40000
20000
0
Difficulty-in-Breathing

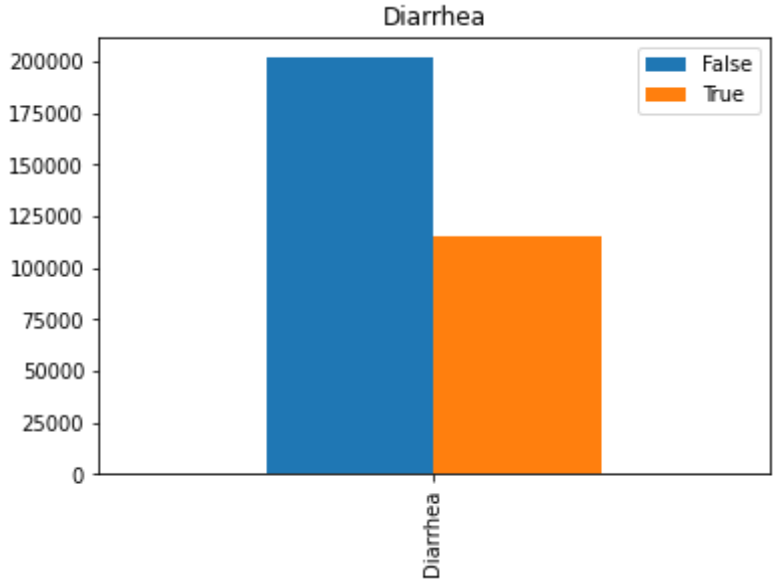

Sore-Throat
False
True
200000
150000
100000
50000
0
Sore-Throat

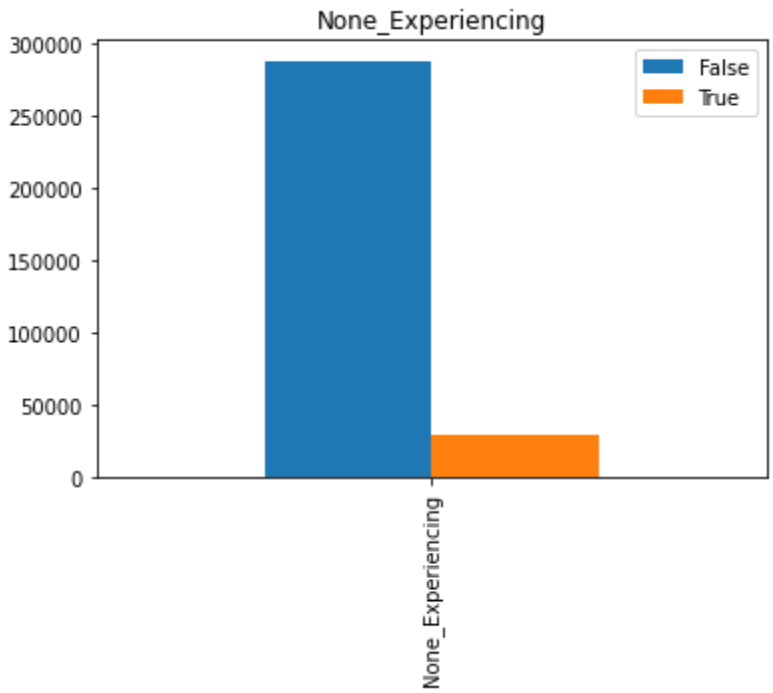

None_Sympton
False
True
300000
250000
200000
150000
100000
50000
0
None_Sympton

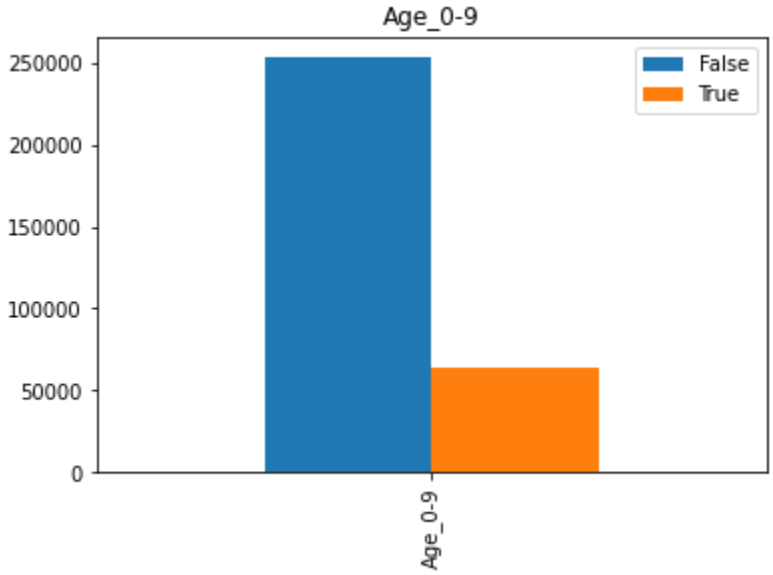

Pains
False
True
200000
175000
150000
125000
100000
75000
50000
25000
0
Pains

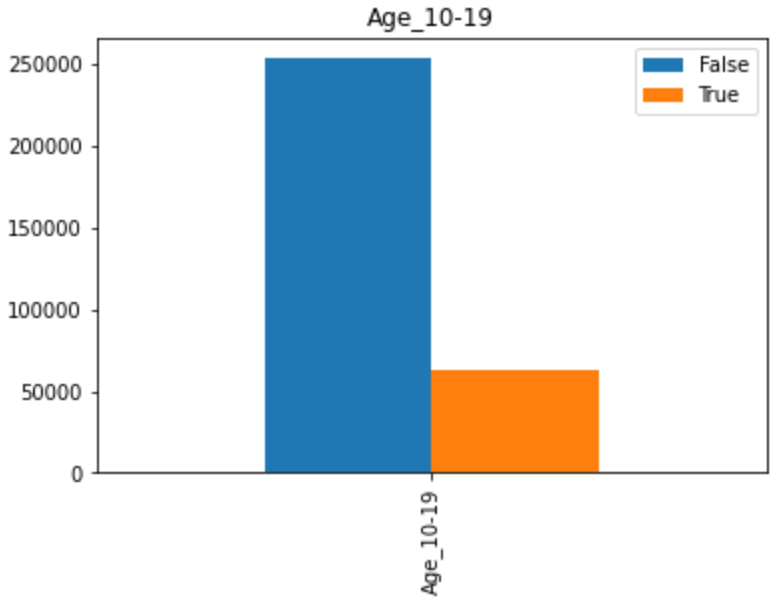

Nasal-Congestion
True
False
175000
150000
125000
100000
75000
50000
25000
0
Nasal-Congestion

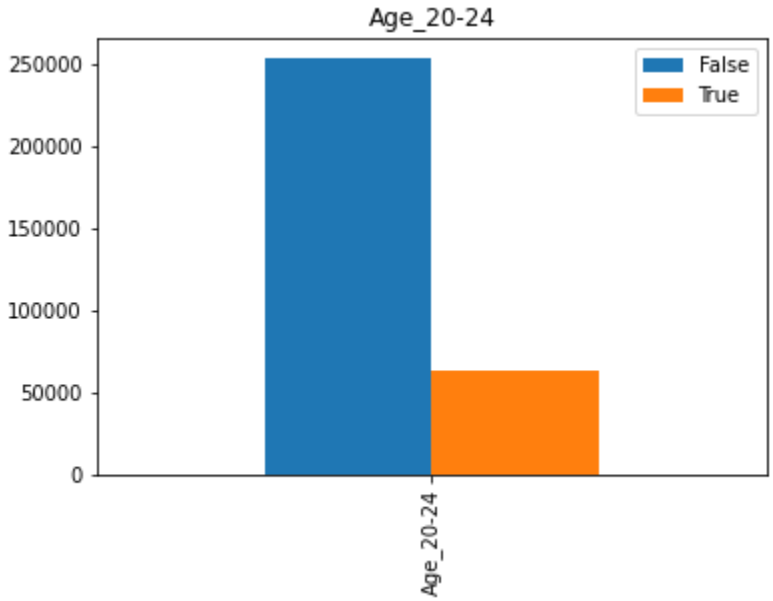

Runny-Nose
175000
150000
125000
100000
75000
50000
25000
0
True
False
Runny-Nose

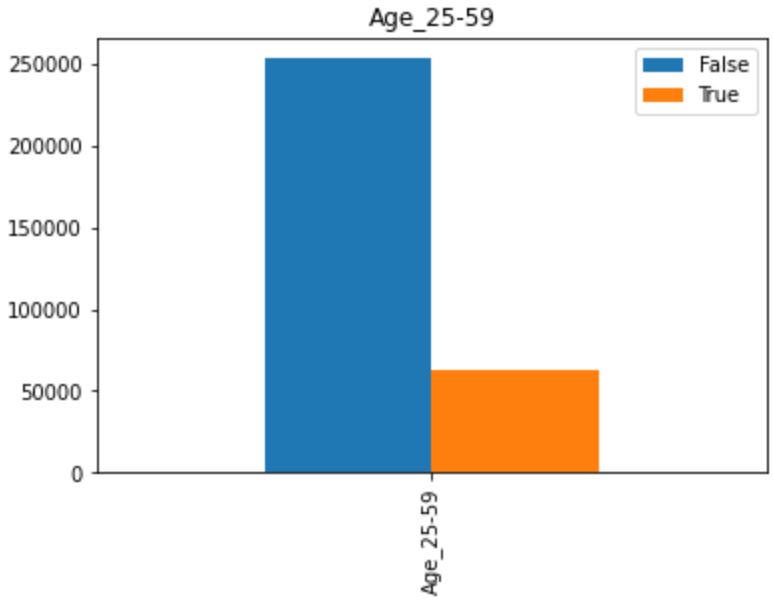

Diarrhea
200000
175000
150000
125000
100000
75000
50000
25000
0
False
True
Diarrhea

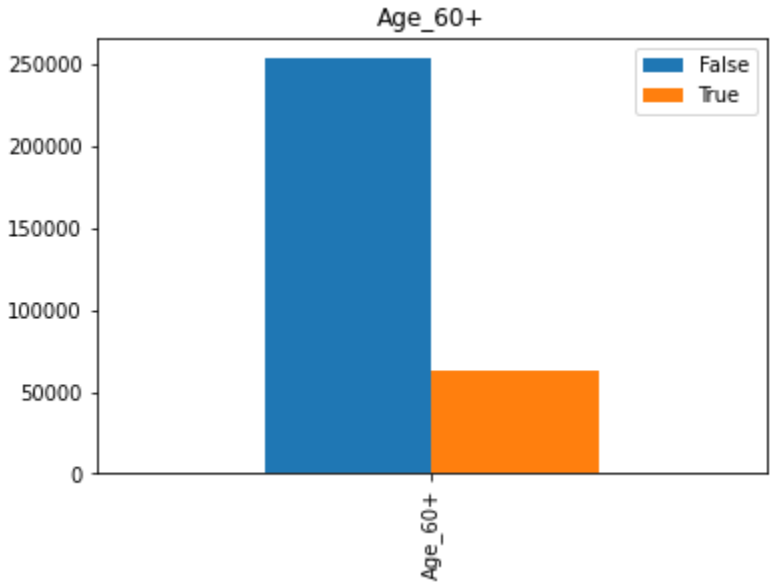

None_Experiencing
300000
250000
200000
150000
100000
50000
0
False
True
None_Experiencing

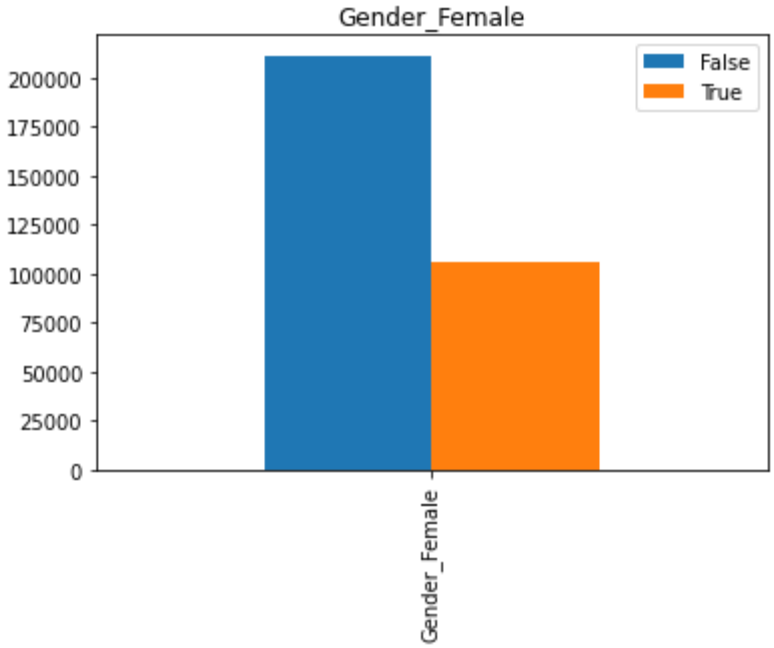

Age_0-9
250000
200000
150000
100000
50000
0
False
True
Age_0-9

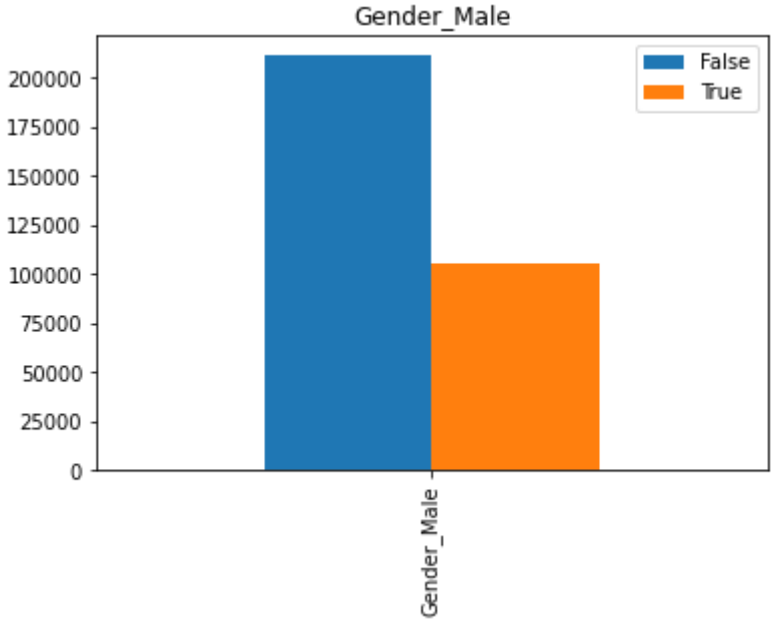

Age_10-19
False
True
250000
200000
150000
100000
50000
0
Age_10-19

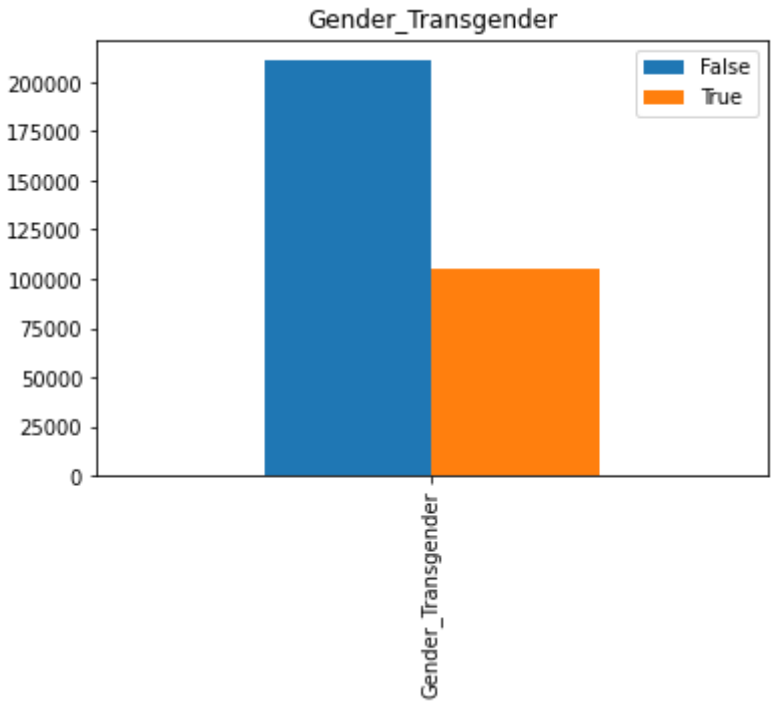

Age_20-24
False
True
250000
200000
150000
100000
50000
0
Age_20-24

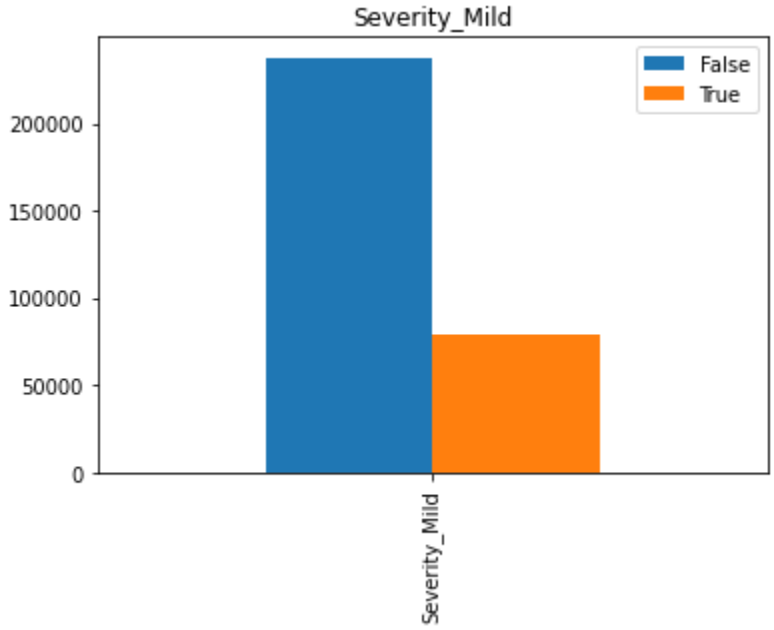

Age_25-59
False
True
250000
200000
150000
100000
50000
0
Age_25-59

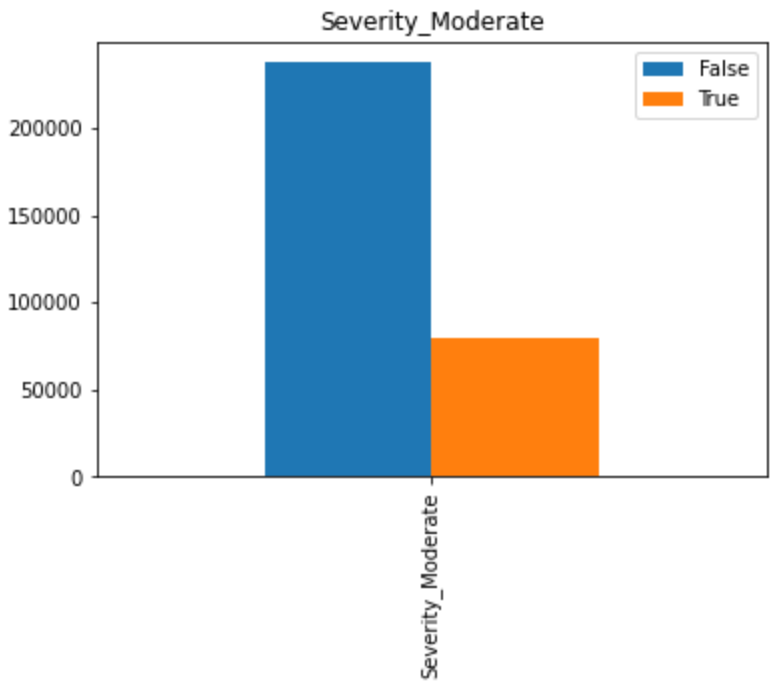

Age_60+
False
True
250000
200000
150000
100000
50000
0
Age_60+

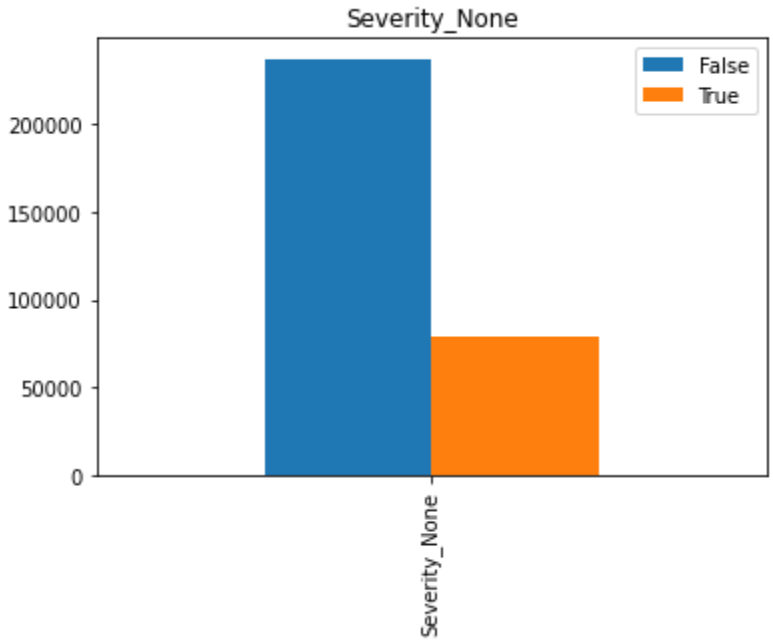

Gender_Female
False
True
200000
175000
150000
125000
100000
75000
50000
25000
0
Gender_Female

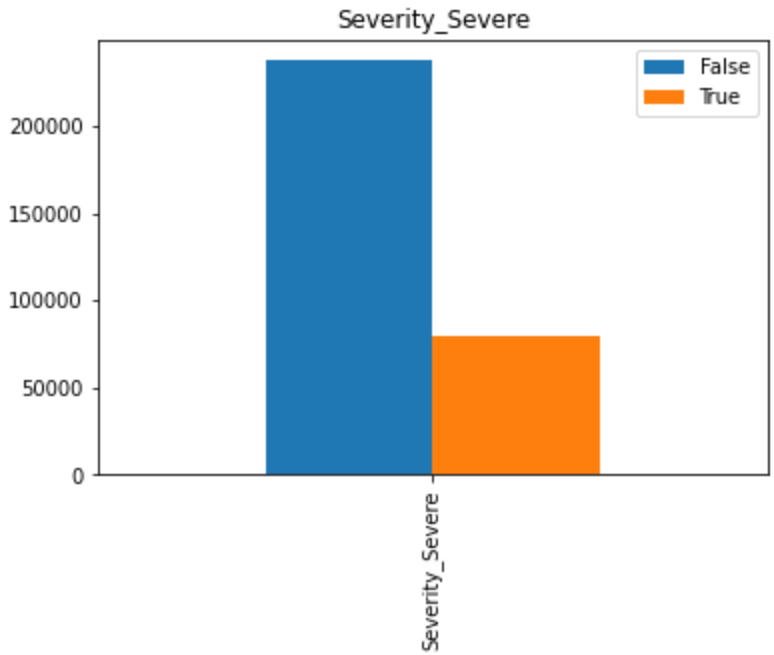

Gender_Male
False
True
200000
175000
150000
125000
100000
75000
50000
25000
0
Gender_Male

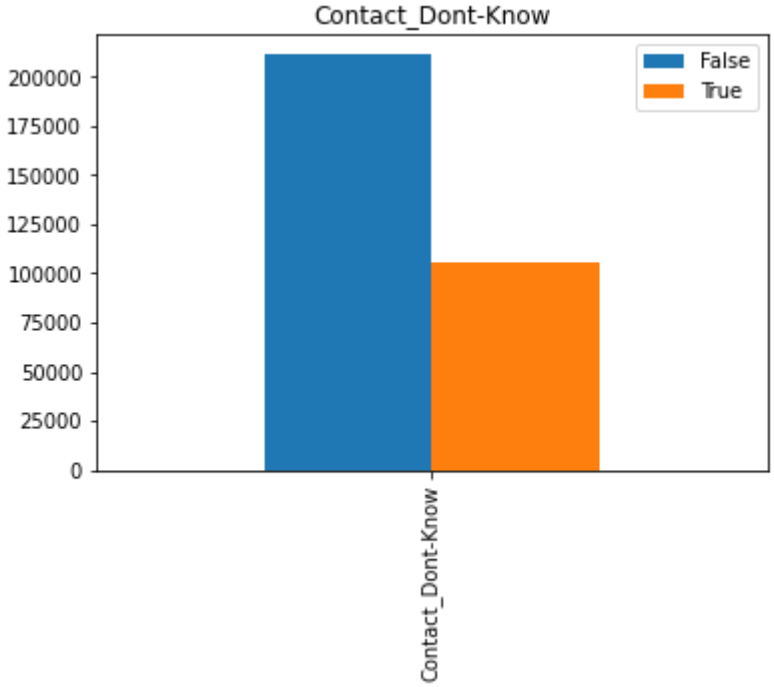

Gender_Transgender
False
True
200000
175000
150000
125000
100000
75000
50000
25000
0
Gender_Transgender

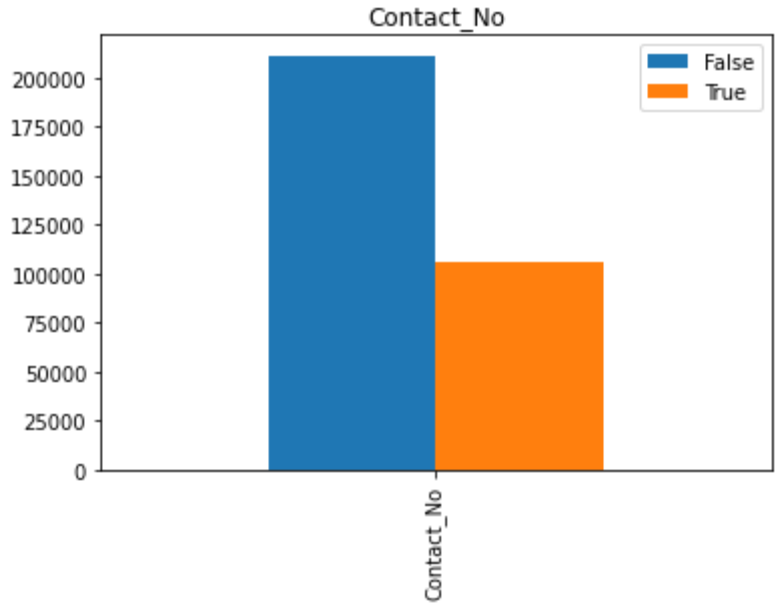

Severity_Mild
False
True
200000
150000
100000
50000
0
Severity_Mild

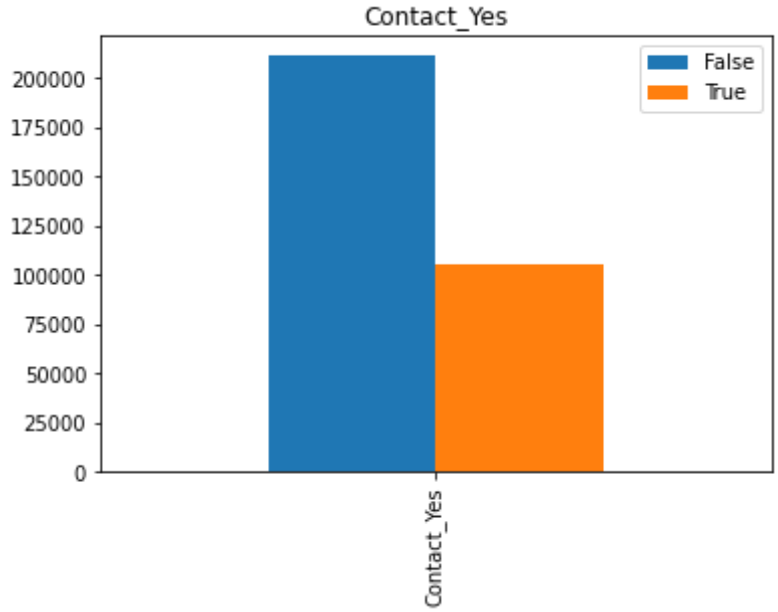

Severity_Moderate
False
True
200000
150000
100000
50000
0
Severity_Moderate


Severity_None
False
True
200000
150000
100000
50000
0
Severity_None

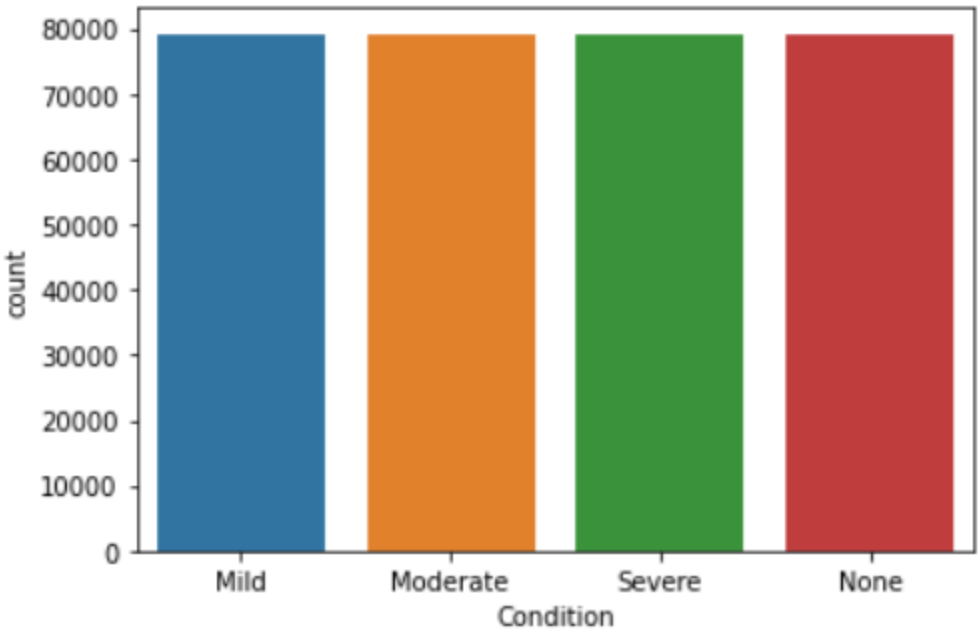

Severity_Severe
False
True
200000
150000
100000
50000
0
Severity_Severe


Contact_Dont-Know
False
True
200000
175000
150000
125000
100000
75000
50000
25000
0
Contact_Dont-Know

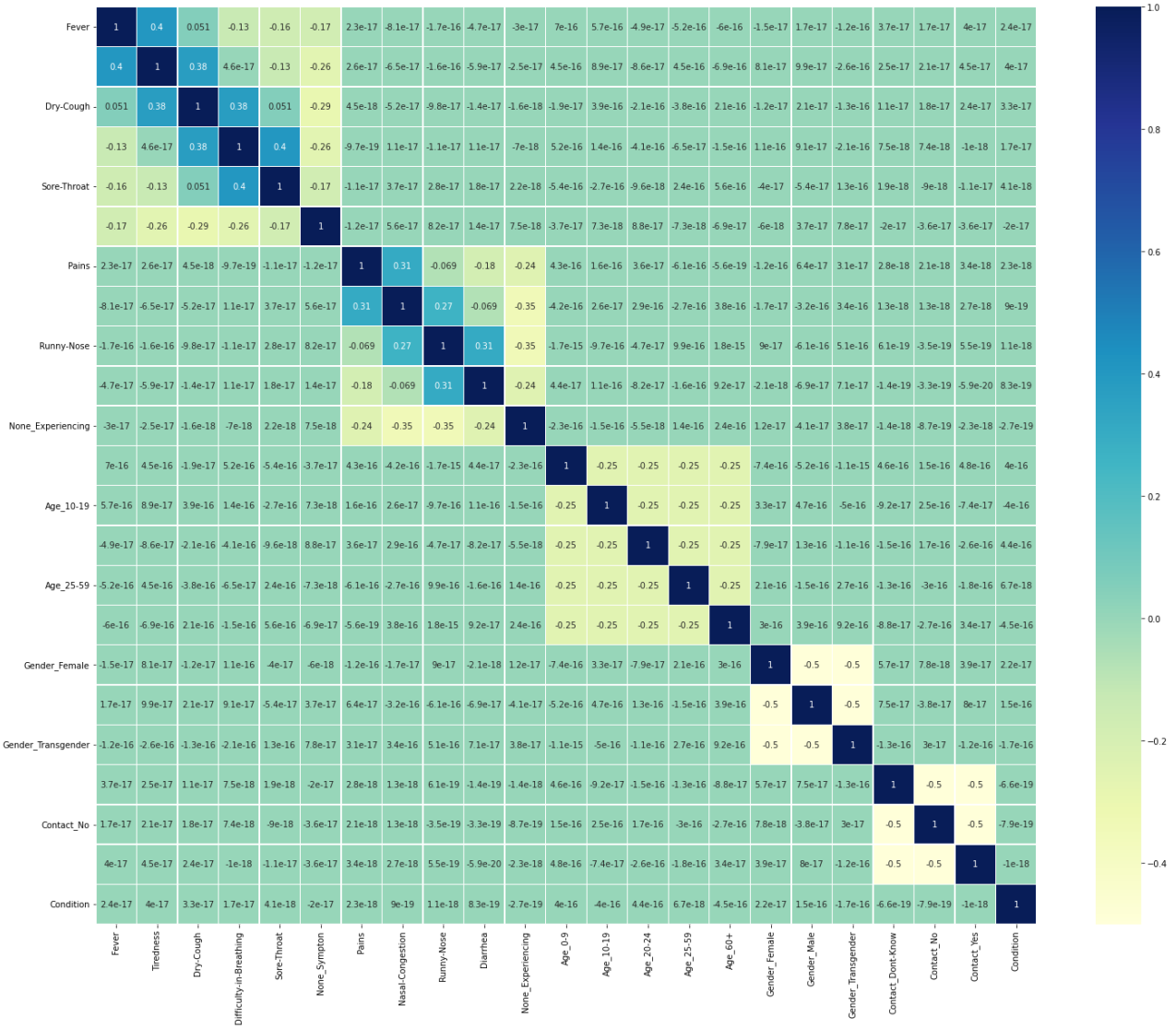




So, we can visualize from the graphs that the True value sum or 1 value sum is represented by the orange color in the graphs and false value sum or 0 value sum is represented by the blue color



In this graph we have visualized the cases of Mild, Moderate, Severe and None conditions and we can see that they are almost the same.



Here, we have created a heatmap to look at the correlation between features with a heatmap so that we can remove or substitute these as they might end up giving wrong results.

VII. Data Pre-processing

After creating various graphs and visualizing the data, we analyze the correlation between different features using the heatmap, and select the features that are strongly correlated to the outcome variable. For our dataset, we have selected these features – Condition, Age_0-9, 'Age_10-19', 'Age_20-24', 'Age_25-59', 'Age_25-59', 'Age_60+', 'Gender_Female', 'Gender_Male', 'Gender_Transgender', 'Contact_Dont-Know', 'Contact_No', 'Contact_Yes'. After selecting our features, we split the dataset in two parts, where one part is 70% and would be used to train our dataset and the rest 20% would be used as the test dataset.

VIII. Model Evaluation

```
# Accuracy on test set
print("Logistic Regression: " + str(accuracy_logreg * 100))
print("K Nearest neighbors: " + str(accuracy_knn * 100))
print("Decision tree: " + str(accuracy_dectree * 100))
print("Random Forest: " + str(accuracy_ranfor * 100))
```

```
Logistic Regression: 24.604377104377104
K Nearest neighbors: 24.813762626262626
Decision tree: 23.203914141414142
Random Forest: 23.464856902356903
```

This screenshot represents the accuracy of Logistic Regression, K Nearest neighbors, Decision tree and Random Forest

```
        Accuracy:0.250
        Classification Report
              precision    recall  f1-score   support

           0       0.00      0.00      0.00     63225
           1       0.25      1.00      0.40     63291
           2       0.00      0.00      0.00     63443
           3       0.00      0.00      0.00     63481

    accuracy                           0.25    253440
   macro avg       0.06      0.25      0.10    253440
weighted avg       0.06      0.25      0.10    253440

990/990 [==============================] - 3s 3ms/step
Dev set:
        Accuracy:0.250
        Classification Report
              precision    recall  f1-score   support

           0       0.00      0.00      0.00      7945
           1       0.25      1.00      0.40      7910
           2       0.00      0.00      0.00      7841
           3       0.00      0.00      0.00      7984

    accuracy                           0.25     31680
   macro avg       0.06      0.25      0.10     31680
weighted avg       0.06      0.25      0.10     31680
```

This screenshot represents the accuracy for Neural Network

```
from tpot import TPOTClassifier


tpot_classifier = TPOTClassifier(generations= 5, population_size= 50,
                                 verbosity= 2,
                                 n_jobs = -1 , random_state = 1 , early_stop = 12,
                                 cv = 5, scoring = 'accuracy')
tpot_classifier.fit(x_train,y_train)
```

```
Generation 1 - Current best internal CV score: 0.7502249053030303

Generation 2 - Current best internal CV score: 0.7502249053030303

Generation 3 - Current best internal CV score: 0.7502249053030303

Generation 4 - Current best internal CV score: 0.7502249053030303

Generation 5 - Current best internal CV score: 0.7502249053030303

Best pipeline: BernoulliNB(input_matrix, alpha=0.1, fit_prior=True)
TPOTClassifier(early_stop=12, generations=5, n_jobs=-1, population_size=50,
               random_state=1, scoring='accuracy', verbosity=2)
```

```
[62] print(accuracy)

0.2512521043771044
```

This screenshot represents the accuracy for TPOT

Logistic Regression

An approach used in supervised learning is logistic regression. It is mostly employed for categorical variable prediction using independent input-output mapping. The result can be classified as discrete or categorical, such as true or false, right or wrong, 0 or 1, etc. However, it doesn't provide precise values, a probabilistic value is provided. One

significant difference between Logistic Regression and Linear Regression is that Linear Regression predicts a regression line whereas Logistic Regression is used to solve classification problems.

Logistic Regression depends upon these 2 assumptions:

1) The dependent variable has to be categorical or discrete.
2) The independent variable shouldn't have multi-collinearity which means several other variables in the model should not be correlated.

Generally, Logistic Regression is used to produce these 2 kinds of results:

1) Binomial: Example: True or false.
2) Multinomial: Example: Pink or red or orange
3) Ordinal: Example: low, medium, high.

Logistic Regression can provide probabilities and categorize new data by utilizing continuous and discrete datasets. It can be used to classify the observations by using different types of data. Then, it can easily predict which variable is the most effective which is used for the categorization.

In our case, logistic regression model produced an accuracy of 24.6% on dataset

KNN

The k-nearest neighbors (KNN) is another supervised machine learning algorithm that can be used to solve both classification as well as regression problems. The KNN algorithm works on the assumption that similar data points exist in close proximity. KNN is really useful for the process of image recognition.

In our case, knn produced an accuracy of 24.81% on dataset

Decision Tree

Decision Tree is a one more type of Supervised learning technique which is primarily used to solve classification problems, but in cases, can be used to solve Regression problems as well. The model is a form of supervised learning, meaning that the model is trained and tested on a set of data that contains the desired categorization. The main idea of Decision Trees is to continuously make decisions like yes or no based on certain rules and split the dataset till the point each data point belonging to a different class is isolated. This phenomenon creates a Tree like structure, hence the name. In this tree structure, internal nodes represent features, branches represent decision rules and each leaf node represents the outcome.

Reasons that make decision tree a viable classification model:

1) Since we humans also make decisions based upon our own certain set of rules, it mimics our thinking process while making a decision, and hence, it's easy to understand.
2) The tree-like structure makes it easy for a user who's reading the logic for the first time to understand what is going on.

In our case, decision tree model produced an accuracy of 23.2% on dataset

Random Forest

Random forest is a Supervised Machine Learning Algorithm that combines the output of various decision trees to produce a single outcome. It works well for both regression and classification problems. Instead of focusing on just one decision tree, random forest considers all of the trees and aggregates them to forecast the most common outcome. A singular decision tree is more prone to problems of bias and overfitting, as compared to multiple trees ensembled together in the random forest algorithm. As a result, they are more accurate in predicting outcomes, particularly when decision trees are correlated with one another.

Steps in random forest algorithm are:

Step 1: In Random Forest n number of random records are taken from the data set having k number of records.

Step 2: Independent decision trees are constructed for every sample.

Step 3: An output will be given by every decision tree.

Step 4: Depending on Majority Voting or Averaging for Classification and regression respectively, concluding output is considered.

In our case, random forest model predicted results with 23.46% accuracy

Neural Network

They are a sequence of algorithms which are very similar to the functions and the way an animal brain operates, to identify relationships between a large amount of data. They, hence, duplicate the connections between the neurons and synapses found in the body of animals. Neural networks having many process layers are known as "deep" networks. These are used for deep learning algorithms.
In our work we have designed a neural network consisting of 3 dense layers and 1 input and 1 output layer where we have used the Adam optimizer and sigmoid activation functions to generate the best results.

In our case, neural network classifier model produced an accuracy of 25% on dataset

Pipeline Building using TPOT

The TPOTClassifier performs an intelligent search over machine learning pipelines that can contain supervised classification models, preprocessors, feature selection techniques, and any other estimator or transformer that follows the scikit-learn API. The TPOTClassifier will also search over the hyperparameters of all objects in the pipeline. Here, we have used these parameters: generations, population_size, verbosity, n_jobs, random_state, early_stop, cv and scoring where generations is the number of iterations to the run pipeline optimization process. In our case we are using 5 generations and as the RAM size is 16gb only so it took almost 6 hours to run the pipeline but the accuracy improved only by less than 1% which tells us that the time and resources used are not worth it for the above dataset.

In our case, tpot classifier model produced an accuracy of 25.13% on dataset

IX. Conclusion

The role of machine learning techniques in a regular day-to-day life plays a very critical role as it is used in each and every sector from industrial to healthcare. In our project we have worked on identifying Covid-19 disease in a person based on various factors through machine learning from the dataset provided by WHO. Initially, we have analyzed the dataset by Data Description and Data Visualization sections where we have presented a complete report of the data, then we have pre-processed the data and finally, using the machine learning techniques we have obtained the accuracy of all the 5 machine learning techniques we have used on both the datasets. The final accuracies or the best accuracies which we have obtained are 25.13% from Tpot but the accuracy from the Neural network of 25% seems better when taken time and resources into consideration.

X. Future Scope

Even though the accuracies are not high but still there is a huge scope in increasing these accuracies by using the Convolutional Neural Networks (CNN) where instead of us selecting the features used to classify data the CNN is a deep learning approach which extracts the useful and most important features by itself and classifies the data for better accuracy and proper classification.

References

1) *Charlyn N. V., Jyh-Horng J.J. H., Julio J. E., Xavier A. I. et al., 2021. "COVID-19 Prediction Applying Supervised Machine Learning Algorithms with Comparative Analysis Using WEKA"*
2) *Anit N. R., Jais J., Aswin S., ,Neha G., Deepa N., Arjun S.. et al., 2020. "Prediction and Spread Visualization of Covid-19 Pandemic Using Machine Learning"*
3) *A Dairi, Y Sun, F Harrou. "Deep learning methods for forecasting COVID-19 time series data: A Comparative Study"*
4) *MRH Mondal, S Bharati, P Podder. " Diagnosis of COVID-19 using machine learning and deep learning techniques.*
5) *T Han, FNB Gois, R Oleveira, LR Prates. "Modeling the progression of COVID-19 deaths using Kalman filter AutoML"*
6) *P Wang, X Zheng, J Li, B Zhu - Chaos, Solitons & Fractals. "Prediction of epidemic trends in COVID-19 with logistic model and machine learning technics".*

7) *VK Gupta, A Gupta, D Kumar. "Prediction of COVID-19 confirmed, death, and cured cases in India using random forest model".*
8) *https://www.kaggle.com/*

Contributions:

1. First Page Proposal – Vismaya M
2. Dataset selection - Anuva Agarwal
3. Literature Review Draft – Abhishek Reddy De Reddy
4. Google Colab Code – Anuva Agarwal
5. Documentation – Segment I to VI Data Preprocessing – Vismaya M
6. Documentation – Segment VIII Model Evaluation- Anuva Agarwal
7. Conclusion and future scope – Abhishek Reddy De Reddy