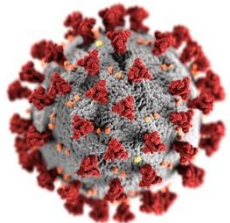
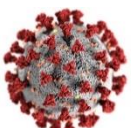
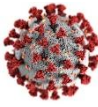


Image source: <https://images.unsplash.com>, Free to use under the [Unsplash License](#)



## CSAI 801 Project



## COVID-19 Outcome Prediction



# Group 1

## Names:

Mohamed Makram Abo Elsood

Osama Ali El Awadia

Ragia Hisham Abutaleb

# Contents:

## ➔ Introduction

1. defining the problem
2. Objectives
3. Methodology
  - a. Data exploration
  - b. Data preprocessing
    - i. One hot encoding
    - ii. Feature scaling (additional preprocessing)
    - iii. Removing rare values (additional preprocessing)
  - c. Applying classifiers and evaluating performance
    - i. KNN
    - ii. Naïve Bayes
    - iii. Logistic Regression
    - iv. SVM
    - v. Decision Tree
4. Comparison between different approaches
5. Conclusion
6. References

## → Introduction

Influenza (flu) and COVID-19 are both contagious respiratory illnesses and you cannot tell the difference between them just by looking at the symptoms alone because they almost have the same symptoms. However, compared to flu, COVID-19 can show more severe symptoms and can cause more serious illness in some people, or it can take longer before some people show symptoms despite being infected, and other people remain contagious for longer periods of time. Serious COVID-19 illness resulting in hospitalization and death can occur even in healthy people. That's why if you feel sick, testing is thoroughly needed to tell what the illness is and to confirm a diagnose because it can also reveal if someone have both the flu and COVID-19 at the same time. In all cases, you should get the needed medical care to help relieve any symptoms or complications.[1]

---

### 1. Defining the problem

Although most people with COVID-19 get better within weeks of illness, some people experience post-COVID conditions (known as long COVID/ or chronic COVID).

Despite the fact that more is learned every day about COVID-19, the virus that causes it, how it spreads, the varying degrees of signs and symptoms, from mild or no symptoms to severe symptoms, and even possible treatments and vaccines, there are still things, such as post-COVID conditions, that are unknown. These conditions can present as different types and combinations of health problems for different lengths of time and can cause death. [1]

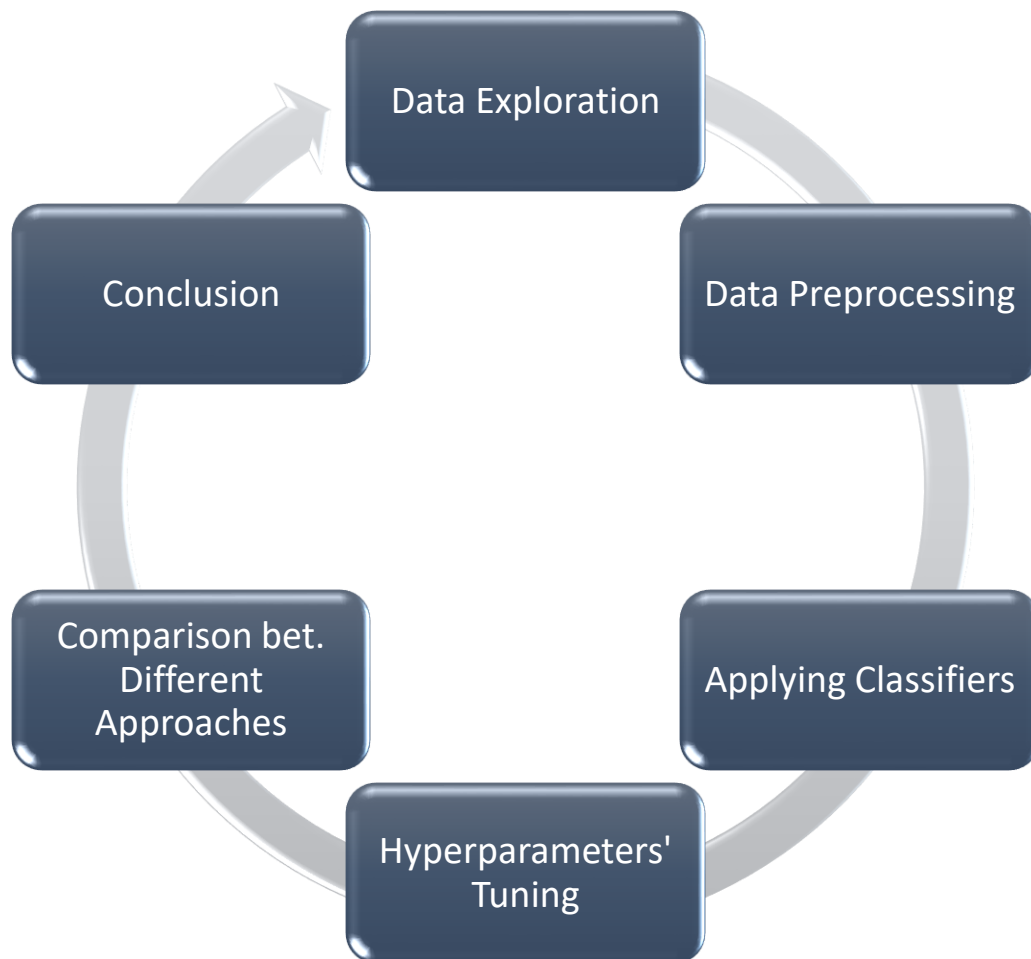
---

## 2. Objectives

There was an alarming need for an algorithm to gather all the pre-defined data from early symptoms to post conditions and the overall health data of patients during the recovery period. So it can be possible to accurately predict how severe the symptoms are, and the chances of the patient to recover and get back their health abilities, or to die; so the medical intervention is seriously and extremely needed. [1]

---

## 3. Methodology



## a. Data Exploration

1. At first, the data was misleading, so we applied pair plot and heatmap data visualization techniques, to be able to get insights from the data, see Fig.1 and Fig.2 below.

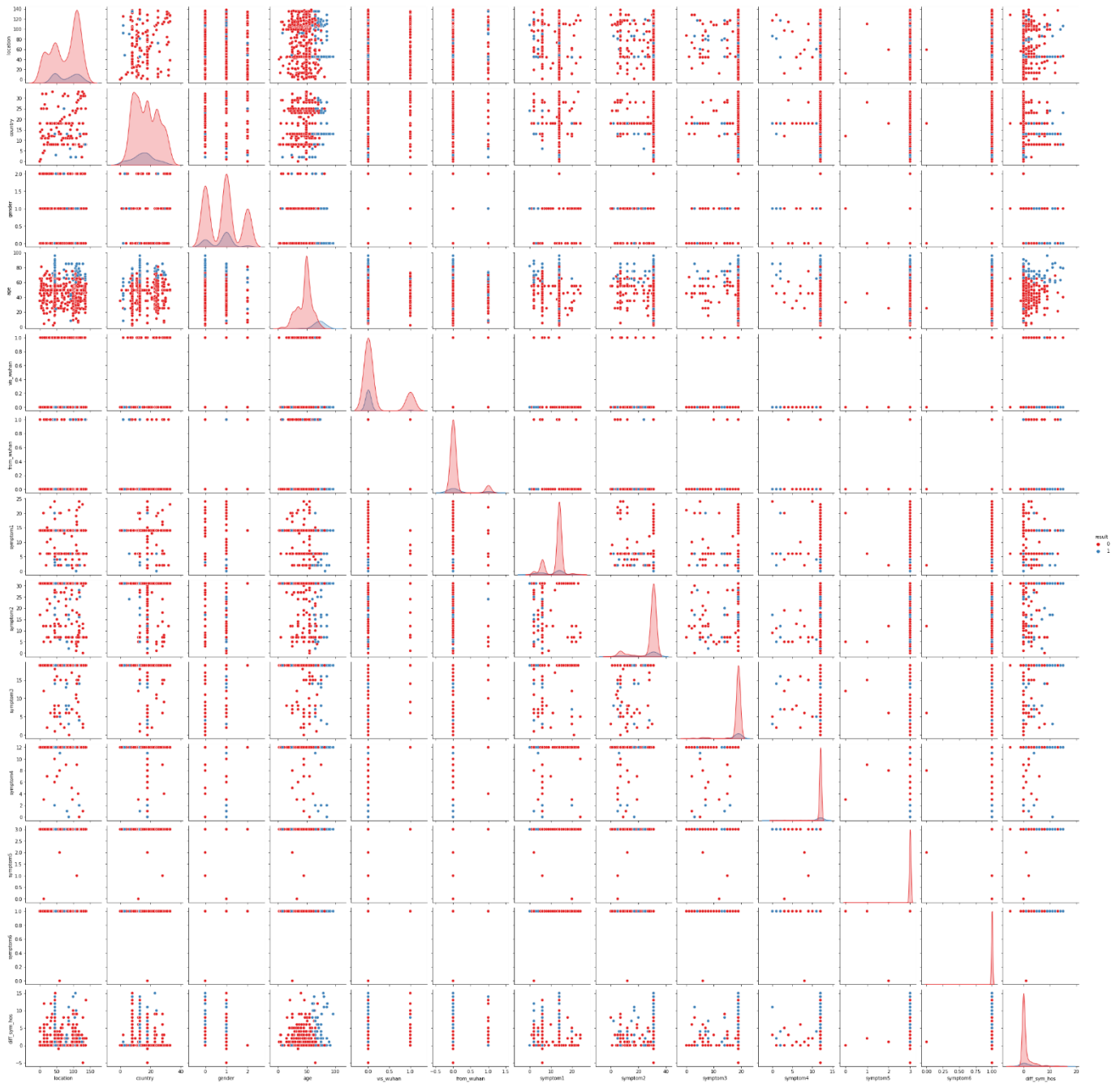


Fig.1: Pair plot

“Pair plot is used to understand the best set of features to explain a relationship between two variables or to form the most separated clusters. It also helps to form some simple classification models by drawing some simple lines or make linear separation in our data-set.”[2]

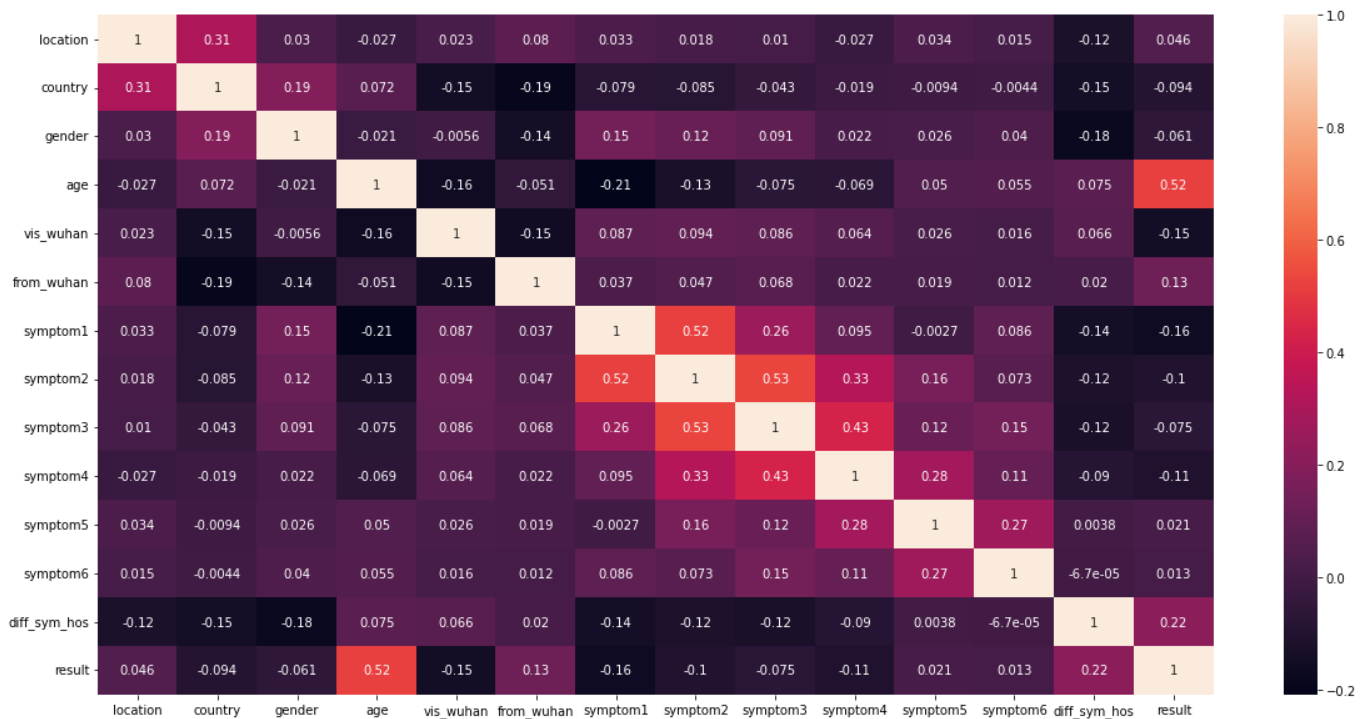


Fig. 2: Heatmap

As seen above in Fig.2, the heatmap shows that there is a quiet high correlation between "age" and "result" so we tried to investigate more to get insights that might help the decision maker.

### >> Further Data Exploration:

```
df[df['age'] > 40]['result'].value_counts()
```

|   |     |
|---|-----|
| 0 | 542 |
| 1 | 103 |

Fig. 3: Recoveries vs. deaths for age above 40

As shown in the python code above, approximately 85% of cases above 40 years old have been recovered and about 15% haven't. The age above 40 plays a crucial role in the data we deal with, the next code explains.

```
df['result'].value_counts()
```

|   |     |
|---|-----|
| 0 | 755 |
| 1 | 108 |

Fig. 4: Recoveries vs. deaths

Here, you can see that out of all 863 cases, there are 645 (approximately 75%) are above 40, so you should consider this age seriously.

```
df.pivot_table('result', index='gender', aggfunc='sum')
```

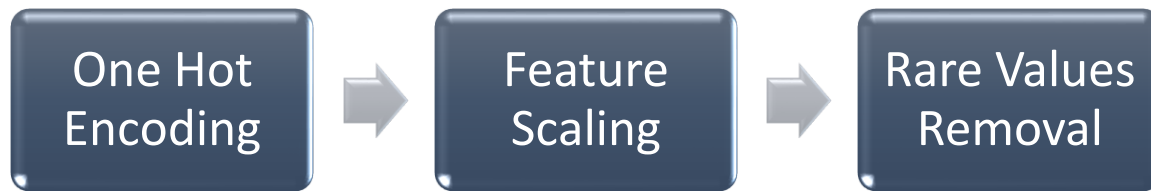
| result |    |
|--------|----|
| gender |    |
| 0      | 34 |
| 1      | 69 |
| 2      | 5  |

Fig. 5: Analysis of no. of deaths

The pivot table shown above explains the number of deaths regarding the 3 genders (female/ male/not mentioned).



## b. Data preprocessing



### i. One hot encoding

Since it is considered to be categorical data, we had to apply one hot coding to most of the features we have. Features like 'location', 'country', 'gender', 'symptoms 1-6', and 'diff\_sym\_hos' were encoded, while 'age', 'vis\_wuhan', 'from\_wuhan', and 'result' weren't.

### ii. Feature scaling

We applied MinMaxScaler from sklearn to let the 'age' values lie in between 0 and 1, without scaling; the classifier may be biased towards the 'age' feature due to its high values rather than other columns after applying 'One Hot Encoding'.

```
Feature Scaling

In [44]: from sklearn.preprocessing import MinMaxScaler, StandardScaler
         scaler = MinMaxScaler()
         #scaler1 = StandardScaler()
         df[['age']] = scaler.fit_transform(df[['age']])

In [45]: df[['age']]

Out[45]:
```

|     | age      |
|-----|----------|
| 0   | 0.680851 |
| 1   | 0.574468 |
| 2   | 0.468085 |
| 3   | 0.617021 |
| 4   | 0.595745 |
| ... | ...      |
| 852 | 0.504255 |
| 853 | 0.504255 |
| 854 | 0.504255 |
| 855 | 0.504255 |
| 856 | 0.504255 |

Fig. 5: Feature Scaling

### iii. Drop rare values

After observing the data, we found that there are lots of values in 'symptoms' columns and 'country' column are very small and removing them will approximately not affect the result but on the other hand the data will be more clean and the number of columns due to one hot encoding will be pretty smaller than before.

NOTE: We already applied the classifiers to the data without removing rare values. You'll also see a comparison between the two approaches.

```
In [20]: df['symptom1'].value_counts()
```

```
Out[20]: 14    644
         6    143
         2     37
         4      7
        20      7
        21      3
        18      2
        22      2
        24      2
        15      1
         0      1
         5      1
        23      1
        16      1
        17      1
         7      1
         9      1
        19      1
        13      1
        11      1
         8      1
        12      1
         1      1
        10      1
         3      1
```

Fig. 6: Example on rare values

## c. Applying classifiers and evaluating performance

- i. KNN
- ii. Naïve Bayes
- iii. Logistic Regression
- iv. SVM
- v. Decision Tree

## i. KNN

→ As it is, on “<https://scikit-learn.org>” [3]

### **sklearn.neighbors.KNeighborsClassifier**

```
class sklearn.neighbors.KNeighborsClassifier(n_neighbors=5, *, weights='uniform', algorithm='auto', leaf_size=30, p=2, metric='minkowski', metric_params=None, n_jobs=None)
```

**n\_neighbors**int, **default=5**

Number of neighbors to use by default for **kneighbors** queries.

→ We’ve worked on ‘n\_neighbors’ hyperparameter.

→ Classification metrics:

| Classification Metrics | First Trial | After Hyperparameters’ Tuning @ n=10&5 | Second Trial | After Hyperparameters’ Tuning @ n=10&5 |
|------------------------|-------------|--|--------------|--|
| Accuracy               | 0.9711      | 0.9191 – 0.9517                        | 0.9655       | 0.9241 – 0.9517                        |
| Precision              | 0.9474      | 1 – 1                                  | 0.9231       | 1 – 1                                  |
| Recall                 | 0.8182      | 0.3636 – 0.5625                        | 0.7500       | 0.3125 – 0.5625                        |
| F1 Score               | 0.8780      | 0.5333 – 0.7200                        | 0.8276       | 0.48 – 0.7200                          |
| AUC                    | 0.995       | 0.998 – 0.887                          | 0.897        | 0.929 – 0.887                          |

Best Hyperparameters: {'n\_neighbors': 3 (manual tuning) & 10 (using GridSearchCV function)}

## Classification Report

```
In [15]:  from sklearn.metrics import classification_report  
          print(classification_report(y_test, y_pred))
```

|              | precision | recall | f1-score | support |
|--------------|-----------|--------|----------|---------|
| 0            | 0.97      | 0.99   | 0.98     | 151     |
| 1            | 0.95      | 0.82   | 0.88     | 22      |
| accuracy     |           |        | 0.97     | 173     |
| macro avg    | 0.96      | 0.91   | 0.93     | 173     |
| weighted avg | 0.97      | 0.97   | 0.97     | 173     |

Fig. 7: KNN classification report

```
Out[8]: <sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay  
        at 0x207b0fc8430>
```

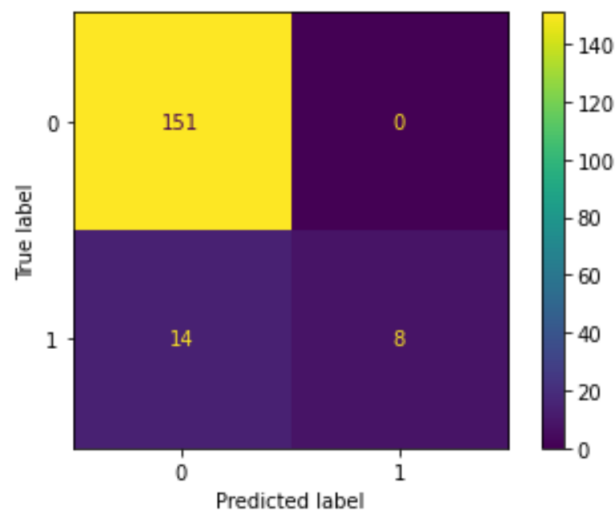


Fig. 8: KNN confusion matrix

## ii. Naïve Bayes

→ As it is, on “<https://scikit-learn.org/>” [4]

### `sklearn.naive_bayes.GaussianNB`

```
class sklearn.naive_bayes.GaussianNB(*, priors=None, var_smoothing=1e-09)
```

**`var_smoothing`***float, default=1e-9*

Portion of the largest variance of all features that are added to variances for calculation stability.

→ We’ve worked on 'var\_smoothing' hyperparameter.

→ Classification metrics:

| Classification Metrics | First Trial | After Hyperparameters' Tuning | Second Trial | After Hyperparameters' Tuning |
|------------------------|-------------|-------------------------------|--------------|-------------------------------|
| Accuracy               | 0.9942      | 1                             | 0.9931       | 1                             |
| Precision              | 0.9565      | 1                             | 0.9412       | 1                             |
| Recall                 | 1           | 1                             | 1            | 1                             |
| F1 Score               | 0.9778      | 1                             | 0.9697       | 1                             |
| AUC                    | 0.997       | 1                             | 0.996        | 1                             |

Best Hyperparameters: {'var\_smoothing': 1e-20}

### iii. Logistic Regression

→ As it is, on “<https://scikit-learn.org/>” [5]

## **sklearn.linear\_model.LogisticRegression**

```
class sklearn.linear_model.LogisticRegression(penalty='l2', *, dual=False, tol=0.0001, C=1.0, fit_intercept=True, intercept_scaling=1, class_weight=None, random_state=None, solver='lbfgs', max_iter=100, multi_class='auto', verbose=0, warm_start=False, n_jobs=None, l1_ratio=None)
```

**penalty{'l1', 'l2', 'elasticnet', 'none'}, default='l2'**

Specify the norm of the penalty:

- 'none': no penalty is added;
- 'l2': add a L2 penalty term and it is the default choice;
- 'l1': add a L1 penalty term;
- 'elasticnet': both L1 and L2 penalty terms are added.

→ We've worked on 'C' and 'penalty' hyperparameters.

→ Classification metrics:

| Classification Metrics | First Trial | After Hyperparameters' Tuning @ n=10&5 | Second Trial | After Hyperparameters' Tuning @ n=10&5 |
|------------------------|-------------|--|--------------|--|
| Accuracy               | 1           | 1                                      | 1            | 1                                      |
| Precision              | 1           | 1                                      | 1            | 1                                      |
| Recall                 | 1           | 1                                      | 1            | 1                                      |
| F1 Score               | 1           | 1                                      | 1            | 1                                      |
| AUC                    | 1           | 1                                      | 1            | 1                                      |

Best Hyperparameters: {'C': 1, 'penalty': 'l2'}

#### iv. SVM

→ As it is, on “<https://scikit-learn.org/>” [6]

### sklearn.svm.SVC

```
class sklearn.svm.SVC(*, C=1.0, kernel='rbf', degree=3, gamma='scale', coef0=0.0, shrinking=True, probability=False, tol=0.001, cache_size=200, class_weight=None, verbose=False, max_iter=-1, decision_function_shape='ovr', break_ties=False, random_state=None)
```

#### **Cfloat, default=1.0**

Regularization parameter. The strength of the regularization is inversely proportional to C. Must be strictly positive. The penalty is a squared l2 penalty.

#### **kernel{'linear', 'poly', 'rbf', 'sigmoid', 'precomputed'} or callable, default='rbf'**

Specifies the kernel type to be used in the algorithm. If none is given, 'rbf' will be used. If a callable is given it is used to pre-compute the kernel matrix from data matrices; that matrix should be an array of shapes (n\_samples, n\_samples).

→ We've worked on 'C' and 'penalty' hyperparameters.

→ Classification metrics:

| Classification Metrics | First Trial | After Hyperparameters' Tuning @ n=10&5 | Second Trial | After Hyperparameters' Tuning @ n=10&5 |
|------------------------|-------------|--|--------------|--|
| Accuracy               | 1           | 1                                      | 1            | 1                                      |
| Precision              | 1           | 1                                      | 1            | 1                                      |
| Recall                 | 1           | 1                                      | 1            | 1                                      |
| F1 Score               | 1           | 1                                      | 1            | 1                                      |
| AUC                    | 1           | 1                                      | 1            | 1                                      |

Best Hyperparameters: {'C': 1, 'kernel': 'linear'}



## v. Decision Tree

→ As it is, from “<https://scikit-learn.org/>” [7]

### **sklearn.tree.DecisionTreeClassifier**

```
class sklearn.tree.DecisionTreeClassifier(*, criterion='gini', splitter='best', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features=None, random_state=None, max_leaf_nodes=None, min_impurity_decrease=0.0, class_weight=None, ccp_alpha=0.0)
```

**criterion{"gini", "entropy"}, default="gini"**

The function to measure the quality of a split. Supported criteria are “gini” for the Gini impurity and “entropy” for the information gain.

**max\_depthint, default=None**

The maximum depth of the tree. If None, then nodes are expanded until all leaves are pure or until all leaves contain less than min\_samples\_split samples.

→ We’ve worked on ‘C’ and ‘penalty’ hyperparameters.

→ Classification metrics:

| Classification Metrics | First Trial | After Hyperparameters' Tuning @ n=10&5 | Second Trial | After Hyperparameters' Tuning @ n=10&5 |
|------------------------|-------------|--|--------------|--|
| Accuracy               | 1           | 1                                      | 1            | 1                                      |
| Precision              | 1           | 1                                      | 1            | 1                                      |
| Recall                 | 1           | 1                                      | 1            | 1                                      |
| F1 Score               | 1           | 1                                      | 1            | 1                                      |
| AUC                    | 1           | 1                                      | 1            | 1                                      |

Best Hyperparameters: {'criterion': 'entropy', 'max\_depth': 1}

#### 4. Conclusion

- Although the accuracy is very high, it's not our first concern. We are more concerned about the recall. Four out of five classifiers (all except KNN) show 100% accuracy and recall which is not reasonable at all and not applicable in real life. So we decided to use KNN which is more reasonable and performs pretty well on the positive (1) class as shown in Fig. 7.

- After removing rare values from each feature, there was a slight change in the results, but the big advantage is that the computing power has definitely been declined as the dimensions of the data has reduced from 863 rows to 724 rows, at first, we've worked on the data without removing anything ,we just did that as an extra work.

---

## 5. References

- [1] CDC
- [2] medium.com
- [3] sklearn - KNN
- [4] sklearn - NB
- [5] sklearn - LR
- [6] sklearn - SVM
- [7] sklearn - DT

## Additional resources

- YouTube channel: "Data Professor"
- YouTube channel: "Python Engineer"
- datacamp
- SuperDataScience
- machinelearningmastery