CSAI 801 Project: COVID-19 Outcome Prediction

Mohamed Abdelrazek Rezaik Salim

ID: 20399127

Supervised by:

Dr. Marwa Elsayed

Eng. Amr Zaki

Project Description

- The data used in this project will help to identify whether a person is going to recover from coronavirus symptoms or not based on some pre-defined standard symptoms. These symptoms are based on guidelines given by the World Health Organization (WHO).
- This dataset has daily level information on the number of affected cases, deaths and recovery from 2019 novel coronavirus. Please note that this is a time series data and so the number of cases on any given day is the cumulative number.
- The data is available from 22 Jan, 2020. Data is in “data.csv”.

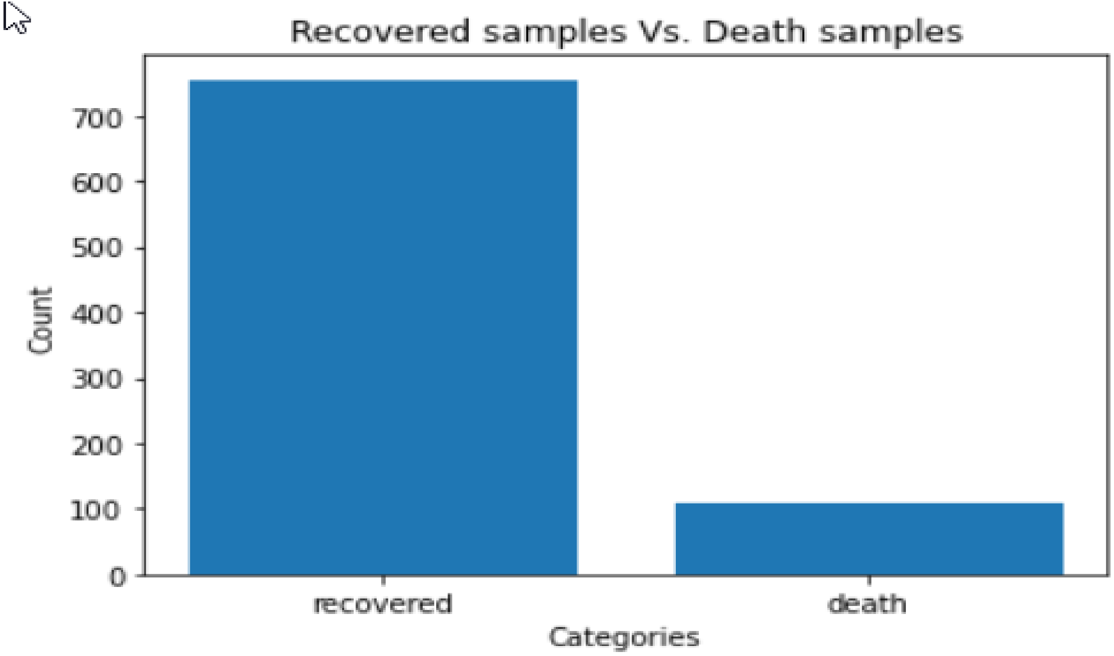
Dataset Properties

The dataset contains 14 major variables that will be having an impact on whether someone has recovered or not, the description of each variable are as follows,

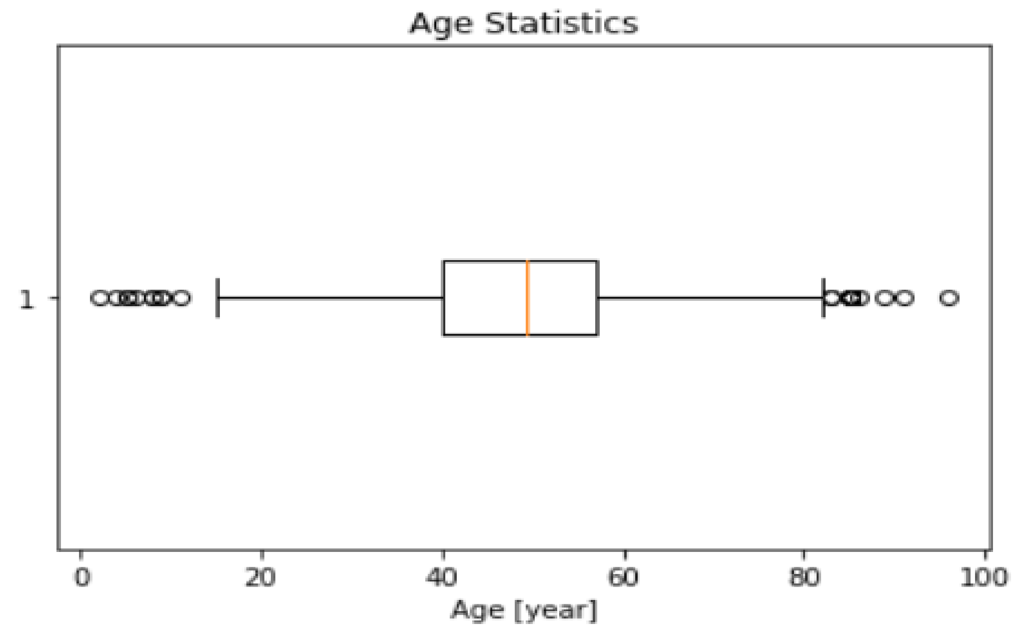
1) **Country:** where the person resides
2) **Location:** which part in the Country
3) **Age:** Classification of the age group for each person, based on WHO Age Group Standard
4) **Gender:** Male or Female
5) **Visited_Wuhan:** whether the person has visited Wuhan, China or not
6) **From_Wuhan:** whether the person is from Wuhan, China or not
7) **Symptoms:** there are six families of symptoms that are coded in six fields.
8) **Time_before_symptoms_appear:**
9) **Result:** death (1) or recovered (0)

Exploratory Data Analysis (EDA)

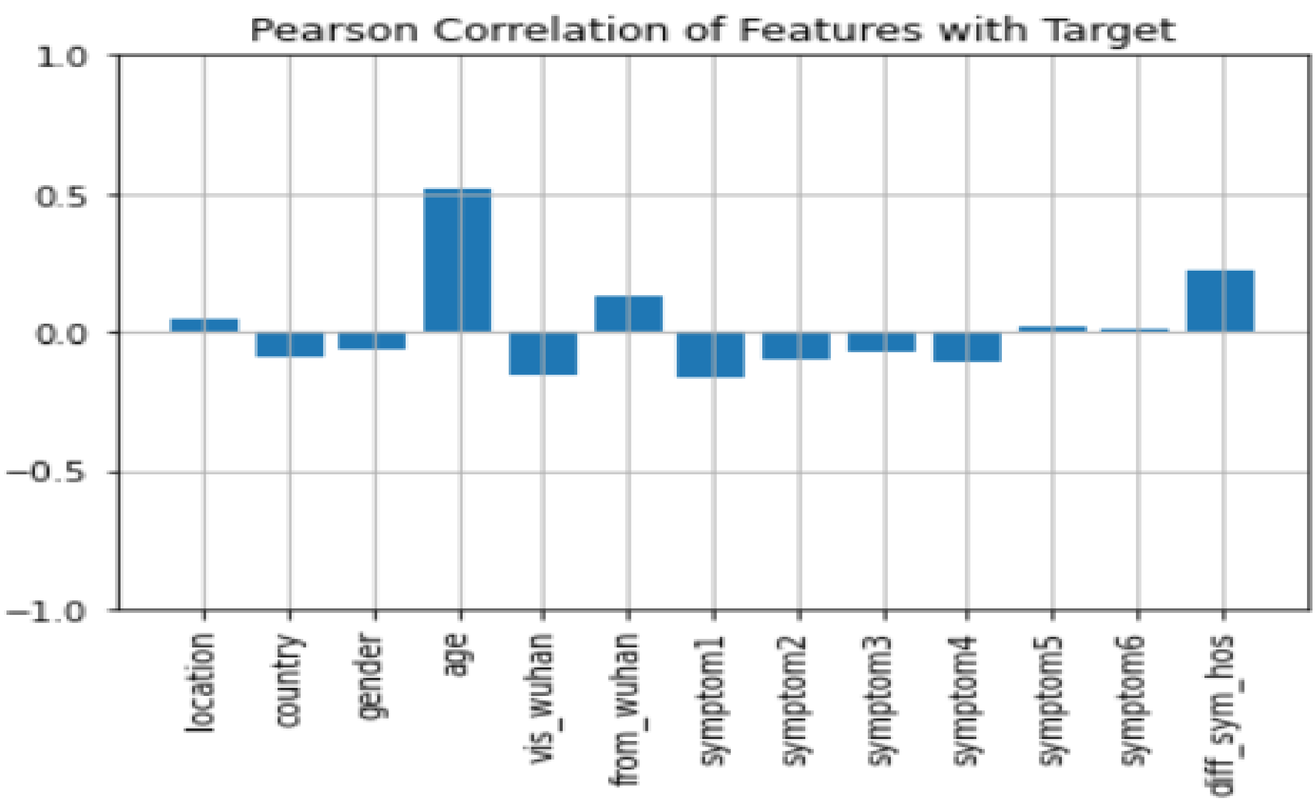
- The data is already cleaned as it is mentioned in the description.
- The data has 13 features and one target column.
- The data has no null values, no outliers, and columns data types are correct.
- The data is already ordinally encoded.
- The data has 863 Records.
- The data is imbalanced, where the percentage of class 1 (recovered) is 13%, and class 0 (death) is 87%.



- The age feature shows that the range from 40 to 60 of age represents most patients.



- Some features have a strong correlation with the target column, and others do not.



- As we can see the “age” column has the strongest relationship with the target it’s about +0.5 and “diff_sym_hos” is about +0.22.
- So, we can assume the age column has a big effect on the target column.

Splitting and Preparing the Data for Training

- We will split the data into training and testing 0.8 to 0.2 respectively with a stratified distribution.

```
# get samples and target
X = df[features]
y = df["result"]
# Get train and test data
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, np.ravel(y),
                                                    test_size=0.2,
                                                    shuffle=True,
                                                    random_state=1,
                                                    stratify=y)
```

- Because the data is imbalanced, we will use the oversampling technique to reduce the imbalance effect, but we will not use it with all types of algorithms.

```
from collections import Counter
from imblearn.over_sampling import SMOTE
# summarize class distribution
print("Before oversampling: ",Counter(y_train))
```

```
Before oversampling:  Counter({0: 604, 1: 86})
```

```
SMOTE = SMOTE()
# fit and apply the transform
X_train_SMOTE, y_train_SMOTE = SMOTE.fit_resample(X_train, y_train)
```

```
# summarize class distribution
print("After oversampling: ",Counter(y_train_SMOTE))
```

```
After oversampling:  Counter({0: 604, 1: 604})
```

- We will split the training data into cross validation sets using “GridSearchCV” object, where number of folds is 10.

```python
def grid_search_hyper(model, params, X_train, y_train, X_test):
    """This function for tuning a model and train it using GridSearchCV object
        inputs:
            model: the model you want to tune and train
            params: the hyperparameters that will be tuned
             X_train, y_train, X_test: data for train test
        outputs:
            grid.best_params_[1]: a dictioinary of best hyperparameters for the model
            y_train_pred: the predicted train y
            y_test_pred: the predicted test y
            predicted_test_proba: the predicted probabilty test y
            grid: the trained GridSearchCV with best hyperparameters accessable"""

    grid = GridSearchCV(estimator=model,
                        param_grid=params,
                        scoring='accuracy',
                        cv=10,
                        return_train_score=False,
                        refit=True,
                        verbose=1)

    # training and Prediction
    grid.fit(X_train, y_train)
    # using the best estimator(the model with best hyperparameters) to predict the target using training and testing data
    y_train_pred = grid.best_estimator_.predict(X_train)
    y_test_pred = grid.best_estimator_.predict(X_test)
    # predict the probabilty of each sample to make smooth curve
    predicted_test_proba = grid.best_estimator_.predict_proba(X_test)

    return grid.best_params_, y_train_pred, y_test_pred, predicted_test_proba, grid
```

Building Models and the Objective

- We train our models on a medical data that require a specific type of evaluation metrics such as Recall metric to be high on both two classes 0 (recovered) and 1 (death). So, our focus will be on this metric more than others to enhance it. The models we will use are:
    (1) K-Nearest Neighbors (Without SMOTE)
    (2) Logistic Regression (With SMOTE)
    (3) Naïve Bayes (With SMOTE)
    (4) Decision Trees (With SMOTE)
    (5) Support Vector Machines (Without SMOTE)

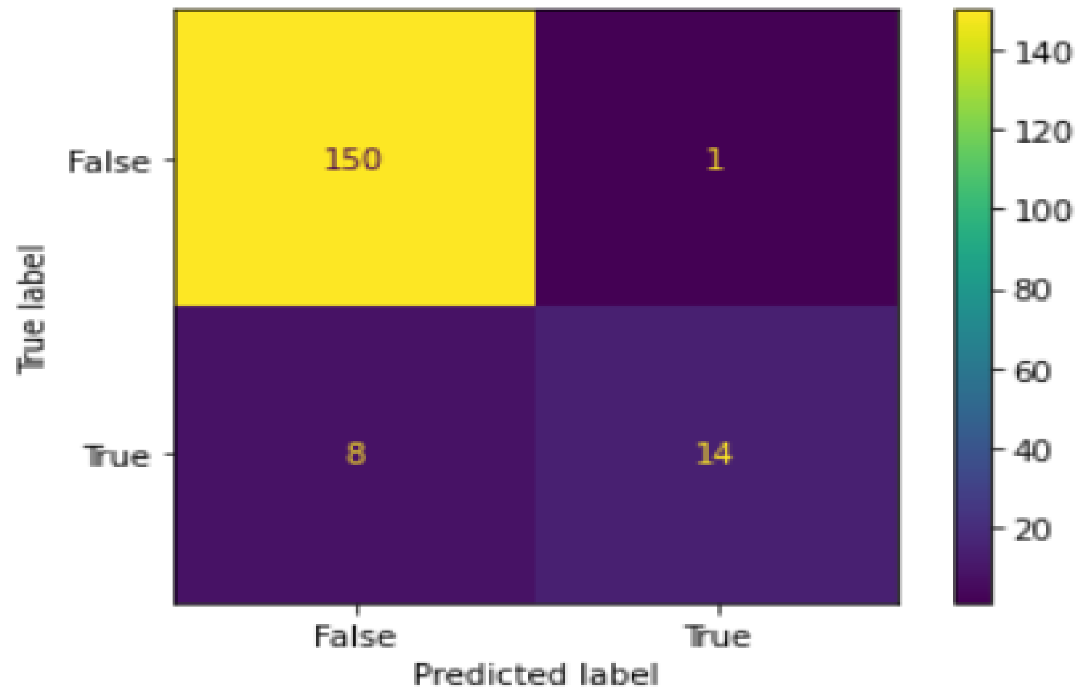
- We noticed that when we use a data augmentation technique such as oversampling to make the dataset balanced with some algorithms the Recall metric becomes better at testing data.
- We will use SMOTE to oversample the data.

1. K-Nearest Neighbors (Without SMOTE)

- When we trained the model with GridSearchCV we got that the best hyperparameter is:
  - The best n_neighbors is: 5
- **Recall:** in class 0 it's good, it's about 0.99, but in class 1 it's not good, it's about 0.64.
- **Precision:** in class 0 it's good, it's about 0.95, and in class 1, it's about 0.93.
- **F1-score:** in class 0 it's good, it's about 0.97, but in class 1 it's not good, it's about 0.76.
- **ROC/AUC:** it's good, it's about 0.99.

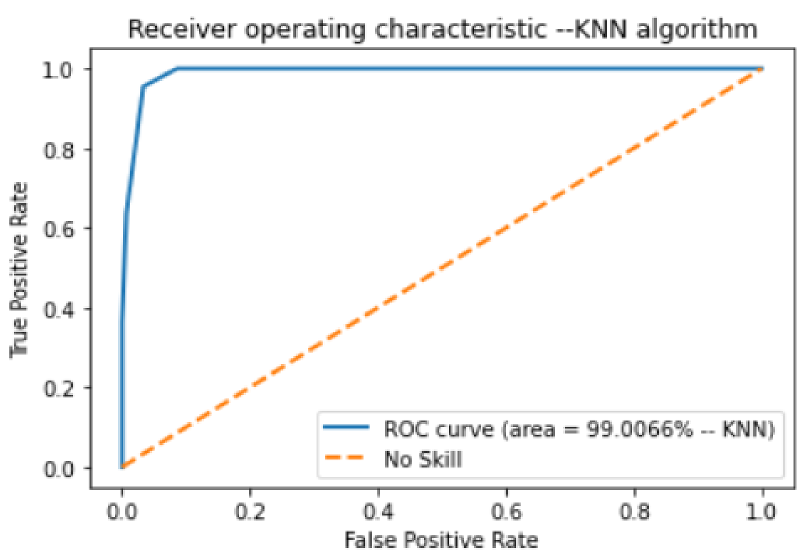
```
-Testing evaluation metrics:

        1- The Recall is: 64%
        1- The Precision is: 93%
        1- The F1-score is: 76%
-----------------------------------------------------------------------
- The calssification report for testing data

              precision    recall  f1-score   support

           0       0.95      0.99      0.97       151
           1       0.93      0.64      0.76        22

    accuracy                           0.95       173
   macro avg       0.94      0.81      0.86       173
weighted avg       0.95      0.95      0.94       173
```

```
-----------------------------------------------------------------
The confusion matrix of Testing predicted data
```




```
# plotting the ROC curve for KNN algorithm
plot_roc(y_test,knn_test_proba_pred[:,1], title="KNN algorithm", algorithm="KNN")
```

```
AUC: 0.9901
No Skill AUC: 0.5000
```



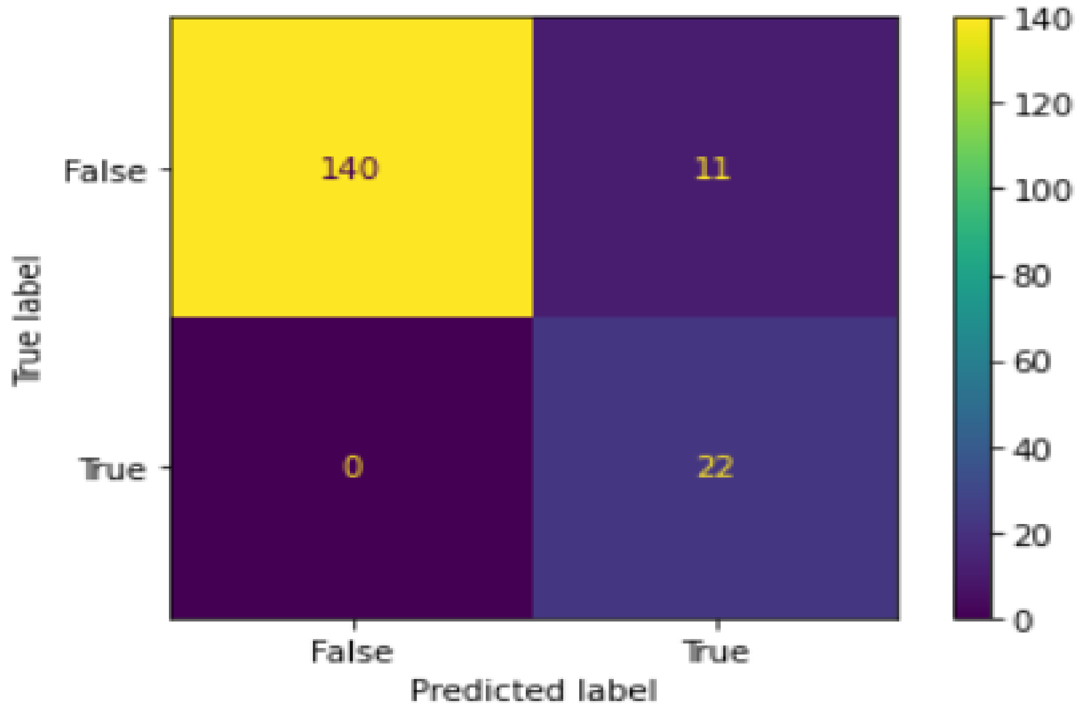
- This model is doing well overall, but the Healthcare application requires a high Recall metric in two classes, where this model is high in class 0, but low in class 1.

2. Logistic Regression (With SMOTE)

- When we trained the model with GridSearchCV we got that the best hyperparameters are:
  - The best C is: 1.0
  - The best penalty is: l2
  - The best solver is: lbfgs
- **Recall:** in class 0 it's good, it's about 0.93, and in class 1, it's about 1.0.
- **Precision:** in class 0 it's good, it's about 1.0, but in class 1 it's not good, it's about 0.67.
- **F1-score:** in class 0 it's good, it's about 0.96, but in class 1 it's not good, it's about 0.80.
- **ROC/AUC:** it's good, it's about 0.98.

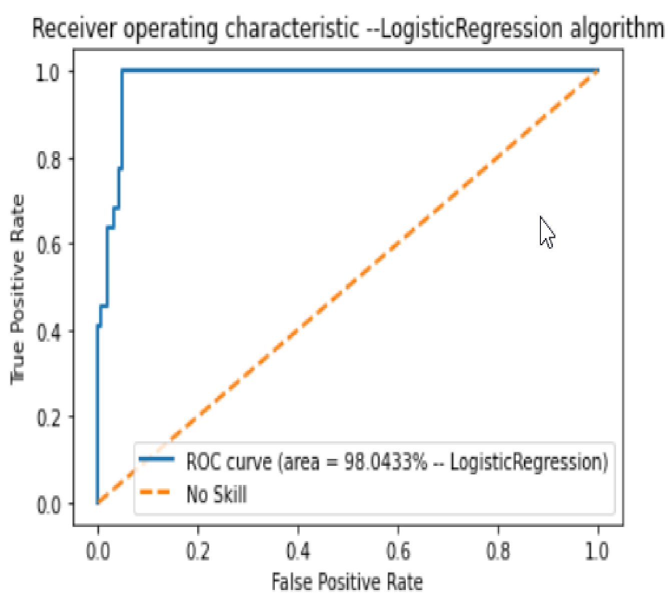
```
-Testing evaluation metrics:

        1- The Recall is: 100%
        1- The Precision is: 67%
        1- The F1-score is: 80%
-----------------------------------------------------------------------
- The calssification report for testing data

              precision    recall  f1-score   support

           0       1.00      0.93      0.96       151
           1       0.67      1.00      0.80        22

    accuracy                           0.94       173
   macro avg       0.83      0.96      0.88       173
weighted avg       0.96      0.94      0.94       173
```

```
The confusion matrix of Testing predicted data
```




```python
# plotting the ROC curve for Logistic Regression algorithm
plot_roc(y_test,logr_test_proba_pred[:,1], title="LogisticRegression algorithm", algorithm="LogisticRegression")
```

```
AUC: 0.9804
No Skill AUC: 0.5000
```



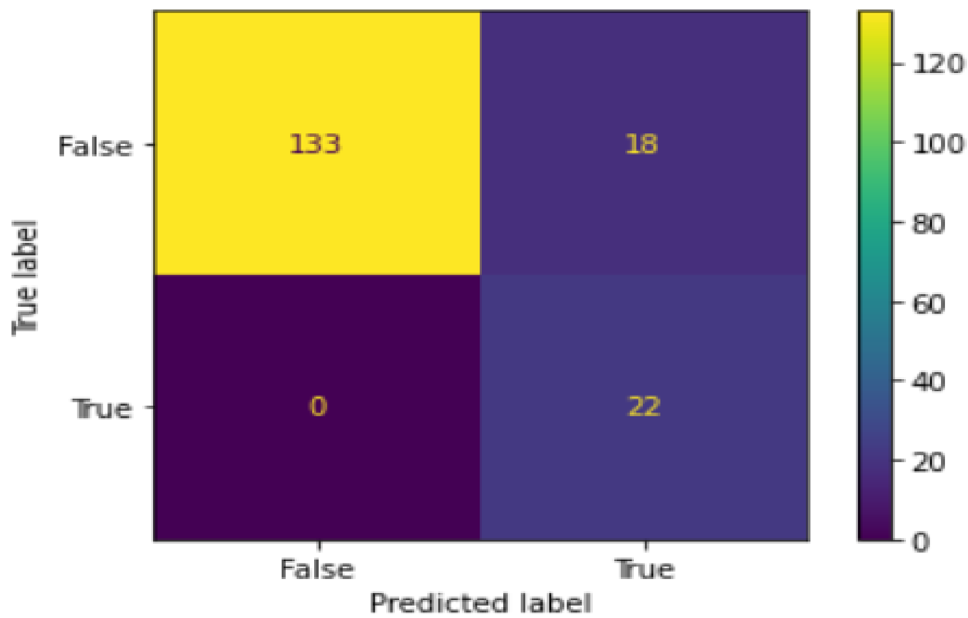
- This model is doing well overall, where this model is high in Recall metric with two classes.

3. Naïve Bayes (With SMOTE)

- We will use Gaussian Naïve Bayes.
- When we trained the model with GridSearchCV we got that the best hyperparameter is:
  - The best var_smoothing is: 0.05542664520663104
- **Recall:** in class 0 it's good, it's about 0.88, and in class 1, it's about 1.0.
- **Precision:** in class 0 it's good, it's about 1.0, but in class 1 it's not good, it's about 0.55.
- **F1-score:** in class 0 it's good, it's about 0.94, but in class 1 it's not good, it's about 0.71.
- **ROC/AUC:** it's good, it's about 0.98.

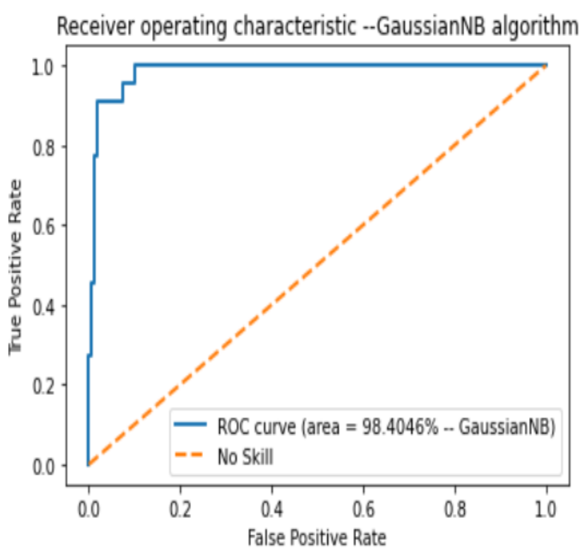
```
----------------------------------------------------------------------
-Testing evaluation metrics:

        1- The Recall is: 100%
        1- The Precision is: 55%
        1- The F1-score is: 71%
----------------------------------------------------------------------
- The calssification report for testing data

              precision    recall  f1-score   support

           0       1.00      0.88      0.94       151
           1       0.55      1.00      0.71        22

    accuracy                           0.90       173
   macro avg       0.78      0.94      0.82       173
weighted avg       0.94      0.90      0.91       173
```

```
The confusion matrix of Testing predicted data
```




```python
# plotting the ROC curve for GaussianNB algorithm
plot_roc(y_test,gauNB_test_proba_pred[:,1], title="GaussianNB algorithm", algorithm="GaussianNB")
```

```
AUC: 0.9840
No Skill AUC: 0.5000
```



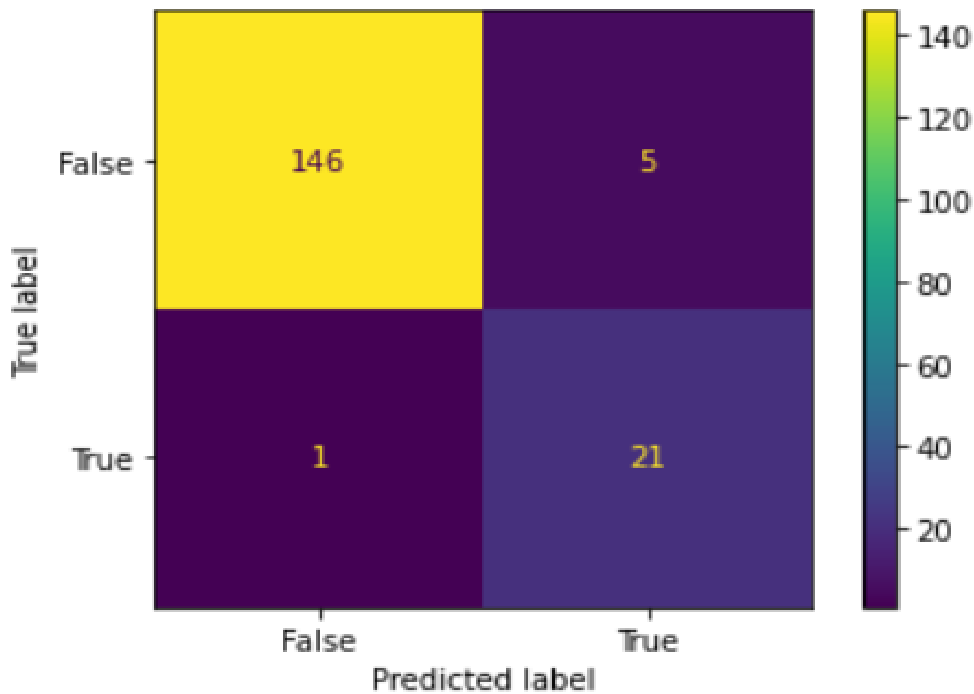
- This model is doing well overall, where this model is high in Recall metric with two classes.

2. Decision Trees (With SMOTE)

- When we trained the model with GridSearchCV we got that the best hyperparameters are:
  - `The best criterion is: gini`
  - `The best max_depth is: 8`
  - `The best min_samples_leaf is: 5`
- **Recall:** in class 0 it's good, it's about 0.97, and in class 1, it's about 0.95.
- **Precision:** in class 0 it's good, it's about 0.99, and in class 1, it's about 0.81.
- **F1-score:** in class 0 it's good, it's about 0.98, and in class 1, it's about 0.88.
- **ROC/AUC:** it's good, it's about 0.96.

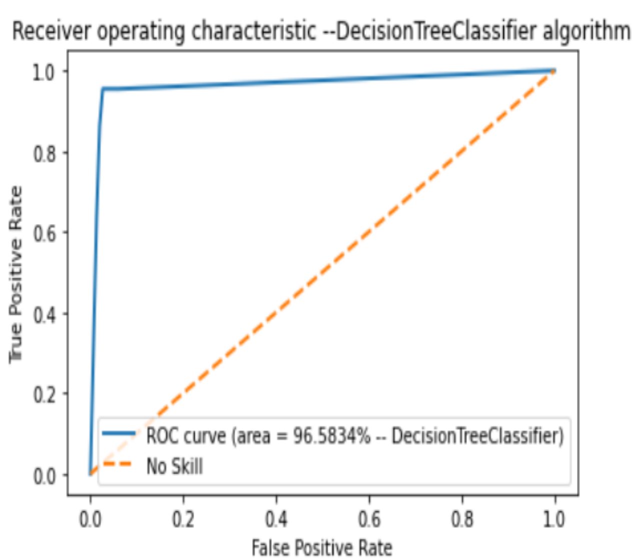
```
-Testing evaluation metrics:

       1- The Recall is: 95%
       1- The Precision is: 81%
       1- The F1-score is: 88%
-----------------------------------------------------------------------
- The calssification report for testing data

              precision    recall  f1-score   support

           0       0.99      0.97      0.98       151
           1       0.81      0.95      0.88        22

    accuracy                           0.97       173
   macro avg       0.90      0.96      0.93       173
weighted avg       0.97      0.97      0.97       173
```

The confusion matrix of Testing predicted data




```
# plotting the ROC curve for DecisionTreeClassifier algorithm
plot_roc(y_test,decT_test_proba_pred[:,1], title="DecisionTreeClassifier algorithm", algorithm="DecisionTreeClassifier")
```

```
AUC: 0.9658
No Skill AUC: 0.5000
```



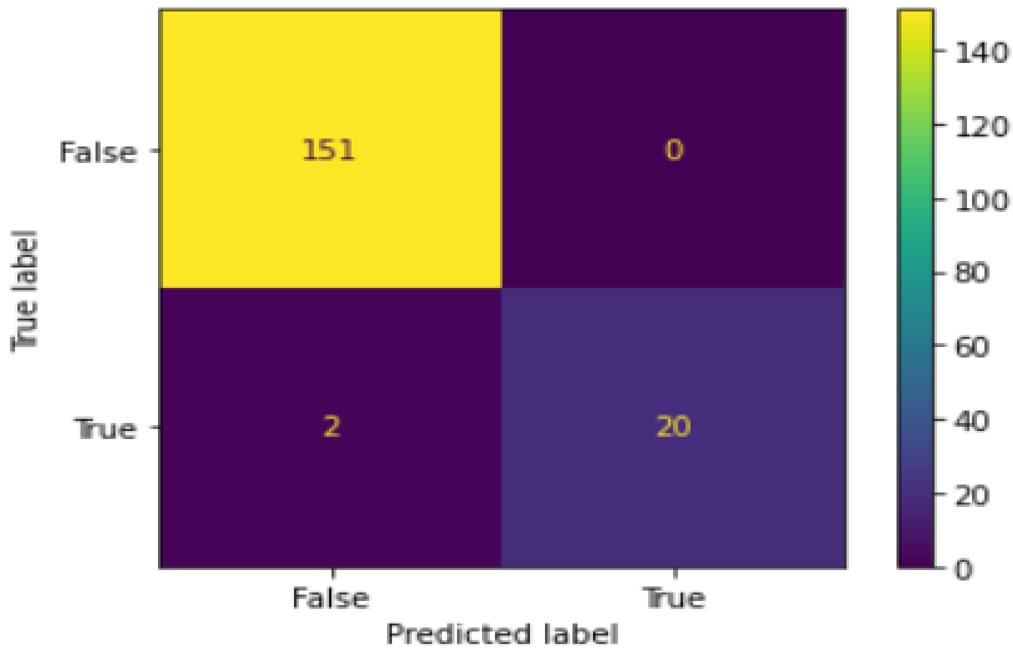
- This model is doing well overall, where this model is high in Recall metric with two classes.

3. Support Vector Machines (Without SMOTE)

- When we trained the model with GridSearchCV we got that the best hyperparameters are:
  - The best C is: 1000
  - The best gamma is: 0.0001
- **Recall:** in class 0 it's good, it's about 1.0, and in class 1, it's about 0.91.
- **Precision:** in class 0 it's good, it's about 0.99, and in class 1, it's about 1.0.
- **F1-score:** in class 0 it's good, it's about 0.99, and in class 1, it's about 0.95.
- **ROC/AUC:** it's good, it's about 1.0.

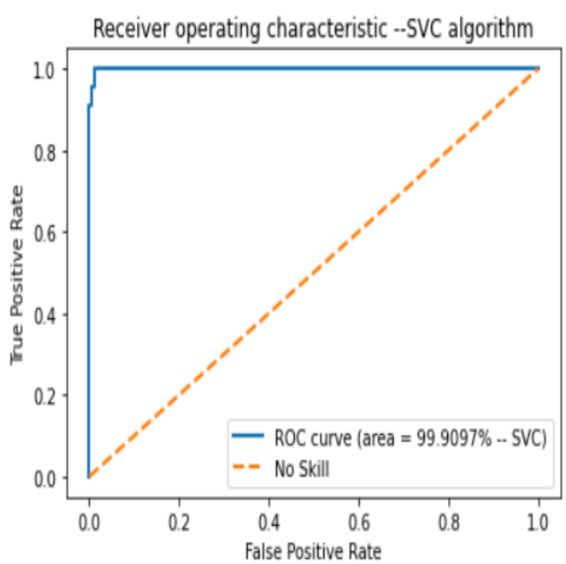
```
-Testing evaluation metrics:

        1- The Recall is: 91%
        1- The Precision is: 100%
        1- The F1-score is: 95%
-------------------------------------------------------------------------
- The calssification report for testing data

              precision    recall  f1-score   support

           0       0.99      1.00      0.99       151
           1       1.00      0.91      0.95        22

    accuracy                           0.99       173
   macro avg       0.99      0.95      0.97       173
weighted avg       0.99      0.99      0.99       173
```

```
The confusion matrix of Testing predicted data
```




```python
# plotting the ROC curve for SVC algorithm
plot_roc(y_test,svc_test_proba_pred[:,1], title="SVC algorithm", algorithm="SVC")
```

```
AUC: 0.9991
No Skill AUC: 0.5000
```

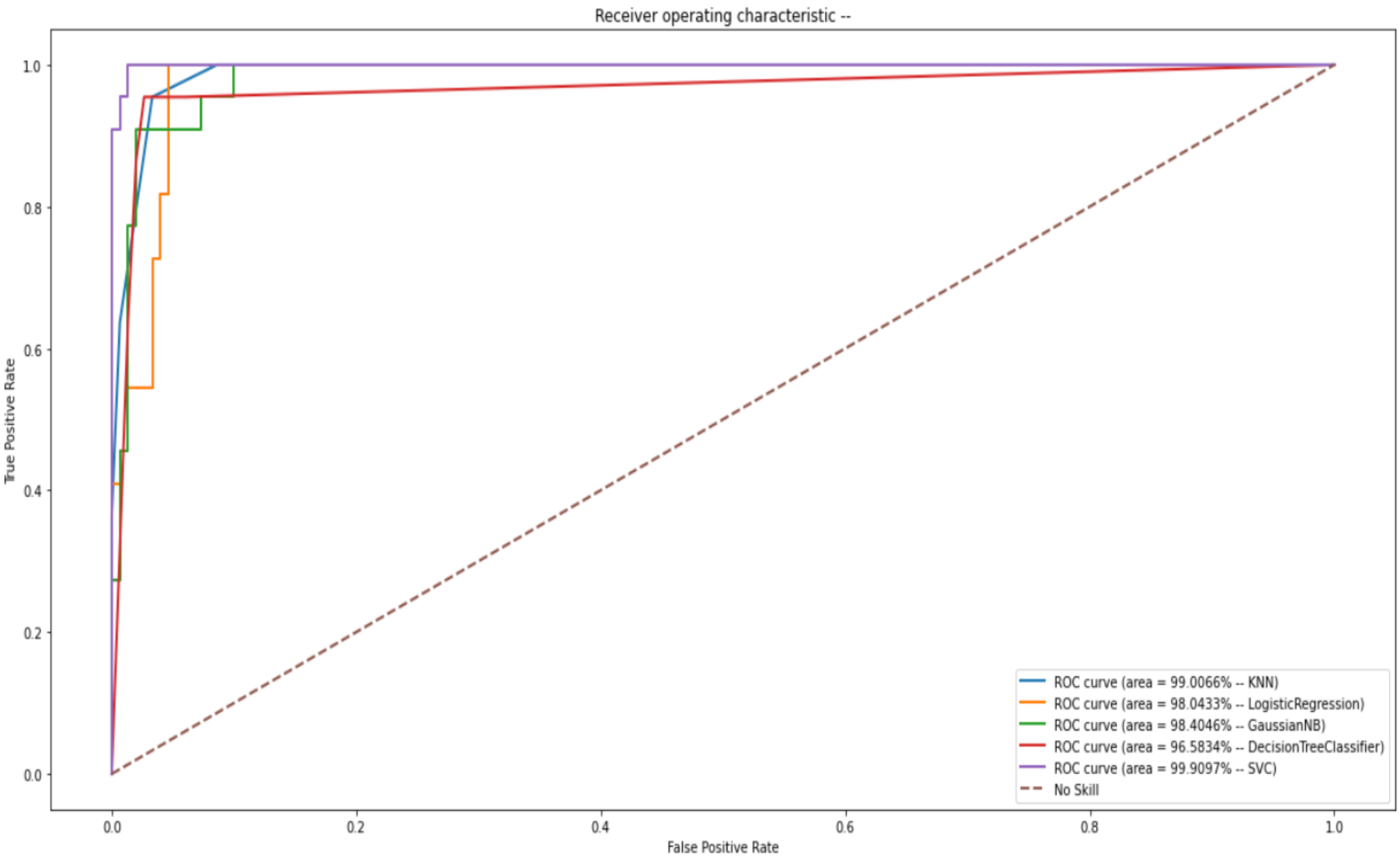


- This model is doing well overall, where this model is high in Recall metric with two classes.

Conclusion

All models are doing well overall, but in healthcare applications that require a high recall metric on the class 1 most of the time, and in our case class 1 (death class), so, the best model should have a high Recall value on this class, and be balanced with other metrics, as shown above Support Vector Machine is suitable for this.

- Best model with each evaluation metric:
    - Recall: Logistic Regression, Decision Tree, and Support vector Machines.
    - Precision: K-nearest Neighbor, and Support Vector Machines.
    - F1-score: Support Vector Machines.
    - ROC/AUC: Support Vector Machines.



Receiver operating characteristic --
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
ROC curve (area = 99.0066% -- KNN)
ROC curve (area = 98.0433% -- LogisticRegression)
ROC curve (area = 98.4046% -- GaussianNB)
ROC curve (area = 96.5834% -- DecisionTreeClassifier)
ROC curve (area = 99.9097% -- SVC)
No Skill