Problem Description:

Covid 19 is considered the biggest problem that has appeared in our society recently, which led to many human losses, so most experiments and studies are directed towards it now so that we can deal with it and make decisions based on accurate results.

Since the artificial body has become one of the most important tools in decision-making, we will use various models to predict the death/survival of patients. based on provided data

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).

Dataset:

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”.
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,

- Country: where the person resides
- Location: which part in the Country
- Age: Classification of the age group for each person, based on WHO Age Group Standard
- Gender: Male or Female
- Visited_Wuhan: whether the person has visited Wuhan, China or not
- From_Wuhan: whether the person is from Wuhan, China or not

- Symptoms: there are six families of symptoms that are coded in six fields.
- Time_before_symptoms_appear:
- Result: death (1) or recovered (0)

Required classifiers:

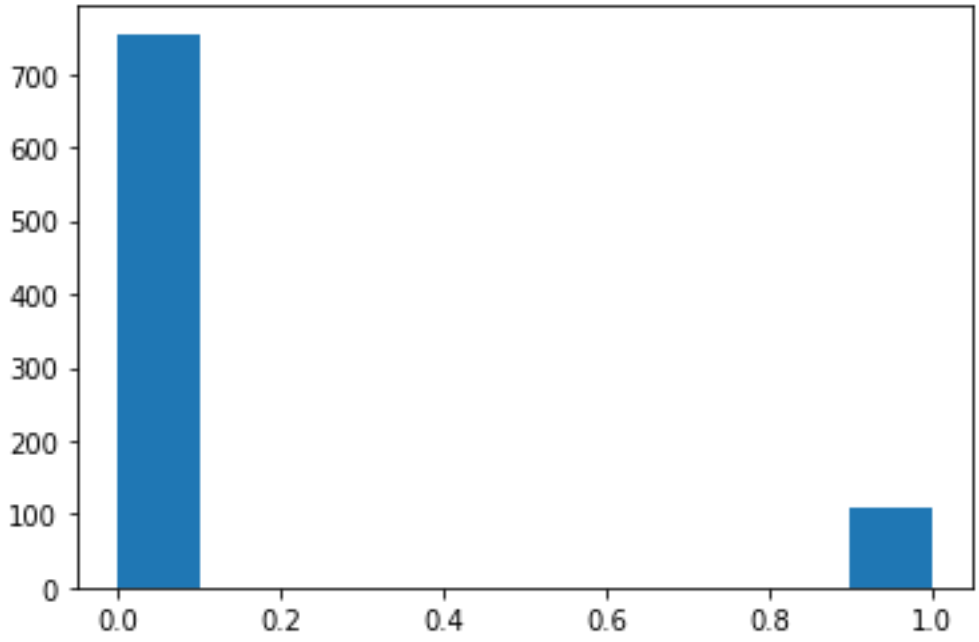
- K-Nearest Neighbors
- Logistic Regression
- Naïve Bayes
- Decision Trees
- Support Vector Machines

Project Walkthrough:

- This project was performed using a Colab notebook.
- Python language was used.
- Each algorithm in a separate notebook.
- The code for each algorithm consists of:
  1- Importing libraries.
  2- Loading data.
  3- Scaling the data.
  4- Oversampling the fewer class in data using SMOTE or class_weight according to the algorithm because there are algorithms that do not support the class_weight argument such as KNN and naïve bayes.
  5- Splitting data using train test split.
  6- Train model.
  7- Calculate the confusion matrix and classification report for train data
  8- Tune the hyperparameters according to each algorithm using grid search or validation data
  9- Test model.
  10- Evaluate the model by calculate the confusion matrix and classification report for test data.

Data exploration:

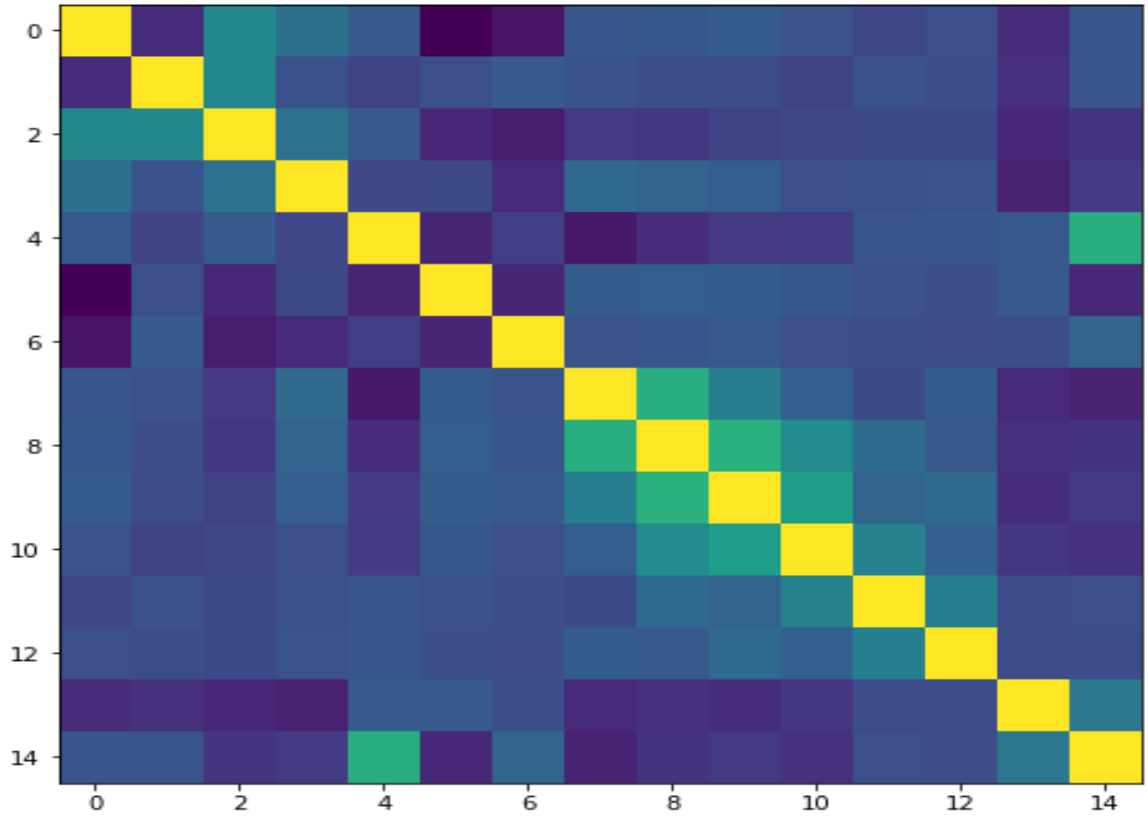
	Unnamed: 0	location	country	gender	age	vis_wuhan	from_wuhan	symptom1	symptom2	symptom3	symptom4	symptom5	symptom6	diff_sym_hos	result
0	0	104	8	1	66.0	1	0	14	31	19	12	3	1	8	1
1	1	101	8	0	56.0	0	1	14	31	19	12	3	1	0	0
2	2	137	8	1	46.0	0	1	14	31	19	12	3	1	13	0
3	3	116	8	0	60.0	1	0	14	31	19	12	3	1	0	0
4	4	116	8	1	58.0	0	0	14	31	19	12	3	1	0	0

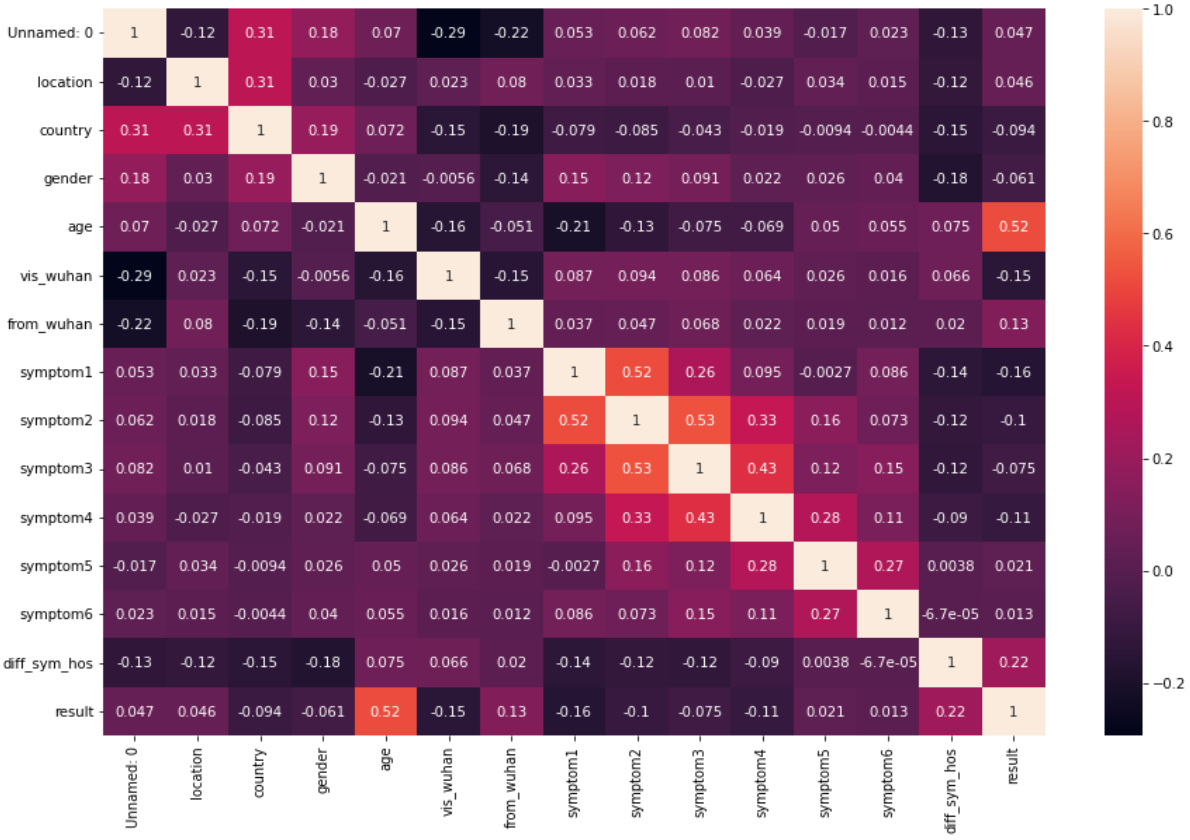


findings and discussion:

- the first figure explains that features values are in different ranges. Therefore, I used StandardScaler() to scale the values in the same range. As a matter of fact, I tried both ways scaled features and unscaled but scaled features did well rather than unscaled regarding metrics.
- Second figure explains that there is an unbalanced two classes in the data so I performed oversampling on the fewer class in data using SMOTE or class_weight according to the algorithm because there are algorithms that do not support the class_weight argument such as KNN and naïve bayes.

Features correlation:





Models

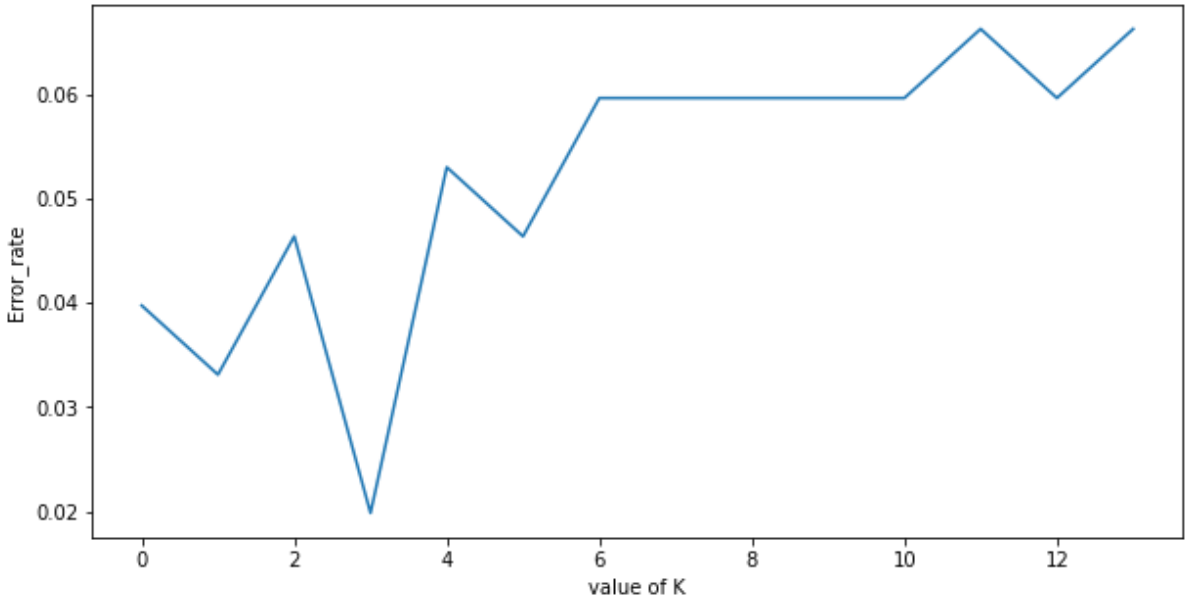
1- K-Nearest Neighbors:

K-nearest neighbors (KNN) is a type of supervised learning algorithm and nonparametric algorithm. KNN tries to predict the correct class for the test data by calculating the distance between the test data and all the training points. Then select the K number of points which is closest to the test data.

Classification report for train data:

	precision	recall	f1-score	support
0	0.99	0.92	0.95	607
1	0.92	1.00	0.96	601
accuracy			0.96	1208
macro avg	0.96	0.96	0.96	1208
weighted avg	0.96	0.96	0.96	1208

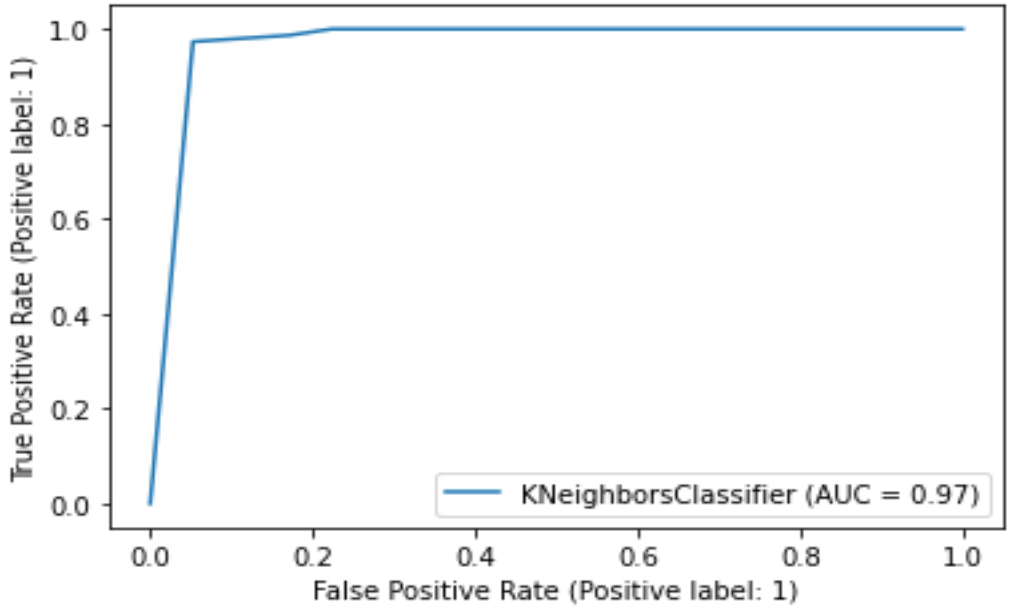
Try to find the best value of K:



Classification report for test data:

	precision	recall	f1-score	support
0	0.98	0.83	0.90	76
1	0.85	0.99	0.91	75
accuracy			0.91	151
macro avg	0.92	0.91	0.91	151
weighted avg	0.92	0.91	0.91	151

Roc curve:



findings and discussion:

- at the first, the algorithm did not take any parameters. However, I started with K=5 and trained the model. First figure reflects the measurements for training data.
- second I tried to find the best value for k and as observed the best value was 3 with a minimum error rate.
- the third figure explains the metrics for testing data and it looks well with regarding the training data.

- At this model I used SMOTE for oversampling because it does not support class_weight argument

2- Logistic Regression:

Logistic Regression is a supervised classification algorithm. One of its most uses is in binary classification (two classes only) such as this project.

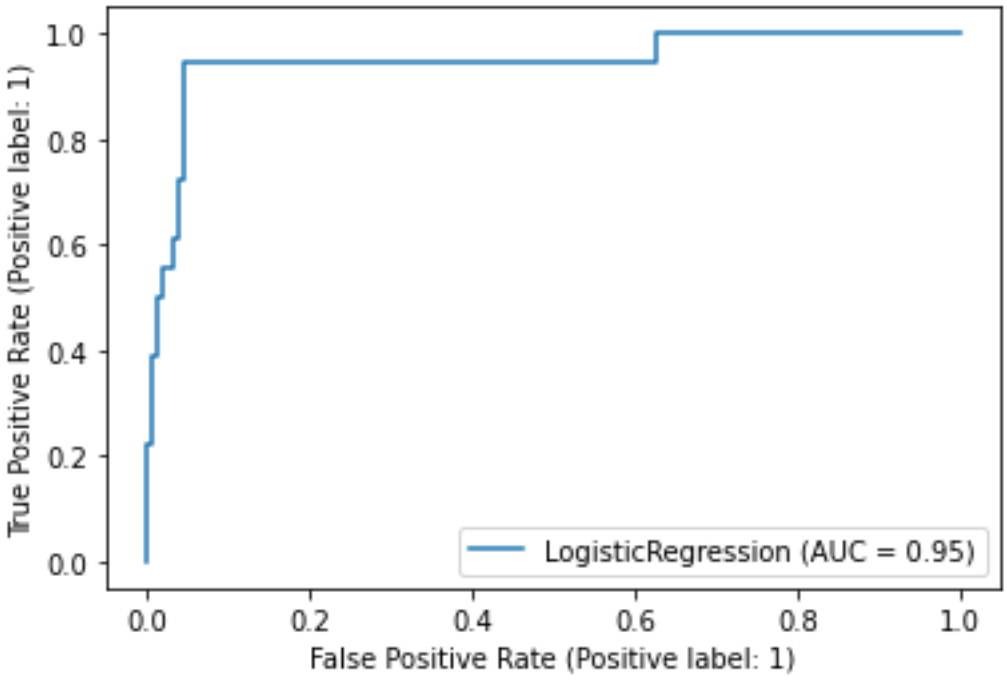
Classification report for train data:

	precision	recall	f1-score	support
0	0.99	0.91	0.95	600
1	0.62	0.96	0.75	90
accuracy			0.92	690
macro avg	0.81	0.93	0.85	690
weighted avg	0.94	0.92	0.93	690

Classification report for test data:

	precision	recall	f1-score	support
0	0.99	0.90	0.95	155
1	0.53	0.94	0.68	18
accuracy			0.91	173
macro avg	0.76	0.92	0.81	173
weighted avg	0.94	0.91	0.92	173

Roc curve:



findings and discussion:

- according to the above figures the model did well in both training and testing considering accuracy, and recall. While, precision and f1 score do not high enough especially for the 1 class.
- At this model I used class_weight for oversampling.

3- Naive Bayes:

Naïve Bayes (NB) is a probabilistic supervised algorithm that works using conditional probabilities. It assumes that all features are independent. However, due to the independency assumption, its work is limited to real data.

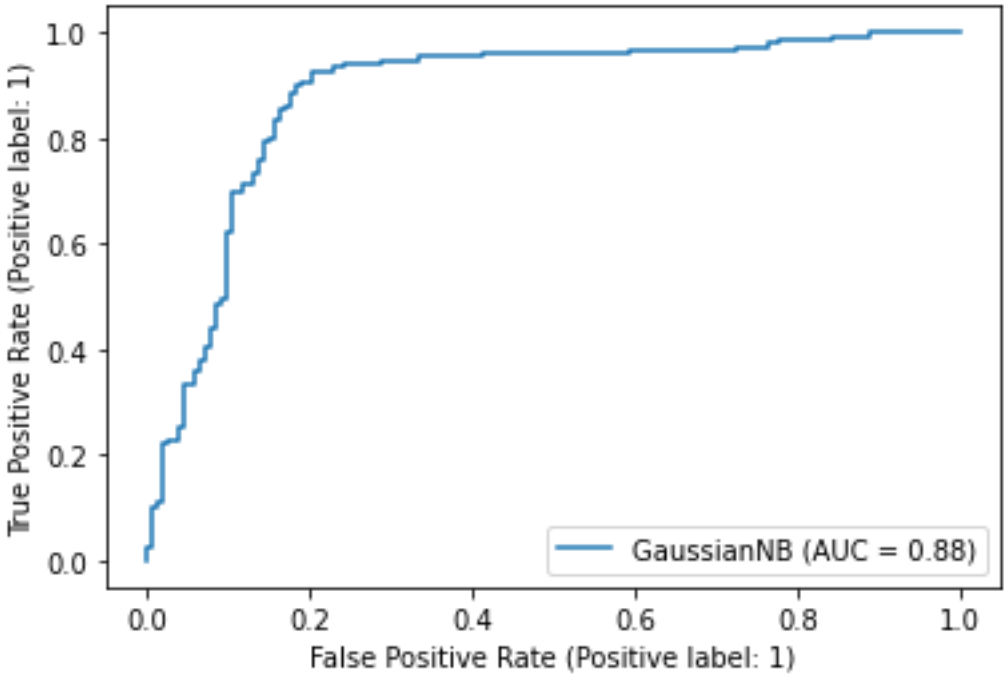
Classification report for train data:

	precision	recall	f1-score	support
0	0.95	0.76	0.84	603
1	0.80	0.96	0.87	605
accuracy			0.86	1208
macro avg	0.88	0.86	0.86	1208
weighted avg	0.88	0.86	0.86	1208

Classification report for test data:

	precision	recall	f1-score	support
0	0.93	0.70	0.80	152
1	0.76	0.95	0.84	150
accuracy			0.82	302
macro avg	0.84	0.82	0.82	302
weighted avg	0.84	0.82	0.82	302

Roc curve:



findings and discussion:

- As shown in the above figures all measurements of this model are not good enough because of consideration of independency of the features.
- At this model I used SMOTE for oversampling because it does not support class_weight argument.
- This model is not suitable for this type of problems.

4- Decision Trees:

Decision Tree is a supervised algorithm. It has a hierarchical structure. It works with either Gini or Entropy to check the best attribute to start with as a root node.

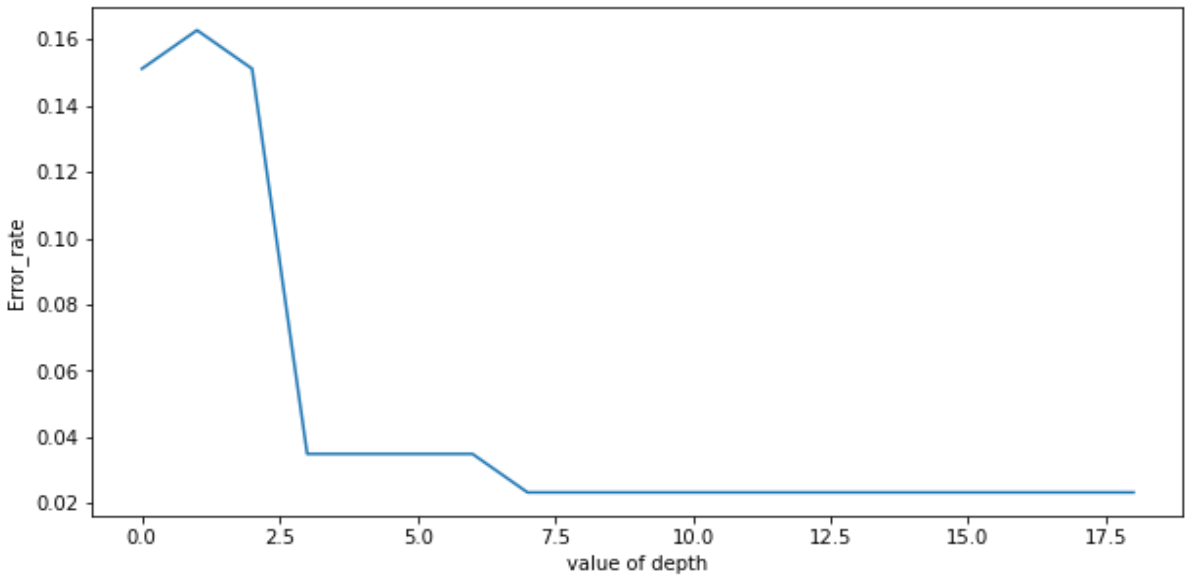
Classification report for train data:

	precision	recall	f1-score	support
0	1.00	0.96	0.98	601
1	0.79	1.00	0.89	89
accuracy			0.97	690
macro avg	0.90	0.98	0.93	690
weighted avg	0.97	0.97	0.97	690

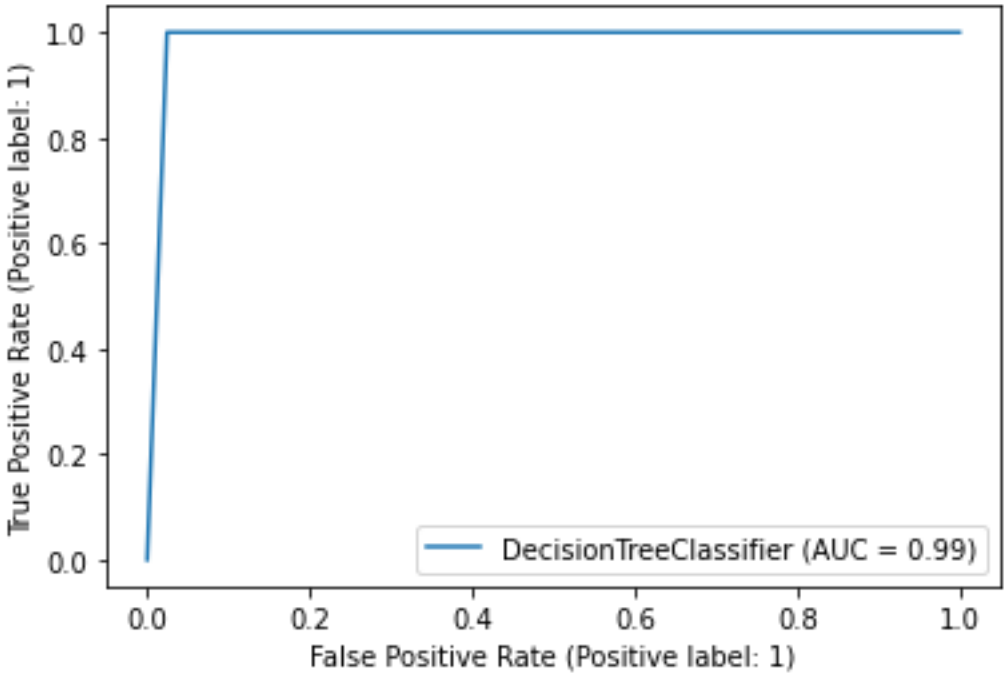
Classification report for test data:

	precision	recall	f1-score	support
0	1.00	0.95	0.97	82
1	0.56	1.00	0.71	5
accuracy			0.95	87
macro avg	0.78	0.98	0.84	87
weighted avg	0.97	0.95	0.96	87

Try to find the best value of max depth:



Roc curve:



findings and discussion:

- the model was traind by using criterion = ‘entropy’ and max depth = 6
- the first figure reflects the training metrics and it looks good for accuracy and recall
- the second figure when trying to find the best max depth and it was 7
- the third figure reflects the testing metrics and it looks good for accuracy and recall as train but not good for persicion and f1 score
- decision tree tends to overfit
- At this model I used class_weight for oversampling.

5- Support Vector Machines:

SVM is a supervised learning model with associated learning algorithms that analyze data for regression and classification.
In addition to performing linear classification, SVMs can efficiently perform a non-linear classification using what is called the kernel trick, implicitly mapping their inputs into high-dimensional feature spaces**.**

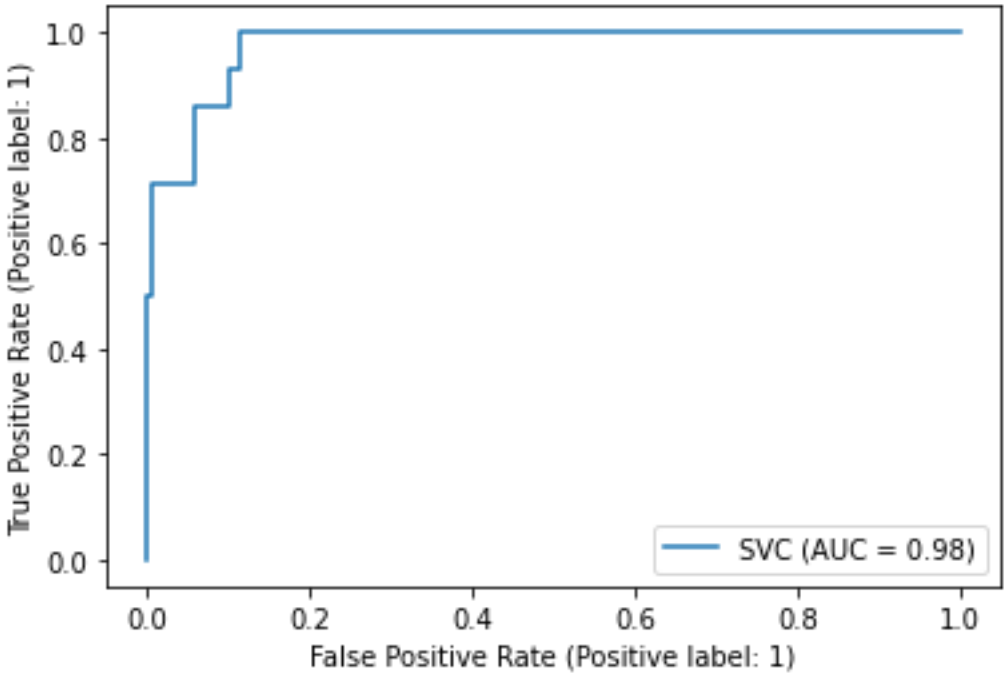
Classification report for train data:

	precision	recall	f1-score	support
0	0.99	0.94	0.97	596
1	0.73	0.97	0.83	94
accuracy			0.95	690
macro avg	0.86	0.96	0.90	690
weighted avg	0.96	0.95	0.95	690

Classification report for test data:

	precision	recall	f1-score	support
0	0.97	0.97	0.97	159
1	0.71	0.71	0.71	14
accuracy			0.95	173
macro avg	0.84	0.84	0.84	173
weighted avg	0.95	0.95	0.95	173

Roc curve:



findings and discussion:

- At first the model was trained by kernel = 'rbf', class_weight='balanced', C=5, gamma=0.01. But by applying grid search the model could reach to optimal values kernel = 'rbf', class_weight='balanced', C=10, gamma=.1
- According to above figures and measurements which reflect that algorithm works really well.
- At this model I used class_weight for oversampling.

Conclusion:

According to the previous output for each algorithm, it is obvious that SVM and decision tree have higher accuracy both have 95% in the test set. However, the decision tree prone to overfit SVM consider the best model. In addition, KNN and logistic regression almost have the same accuracy. Whereas, naïve bayes considers the worst model.