COVID-19 Outcome Prediction

Dr. Hazem Abbas

Names

- Ahmed Mohamed Gaber Abdelaziz ID : 21amga
- Karim Gamal Mahmoud Mohamed ID : 21kgmm
- Ahmed Ibrahim Salem Abdelhamid ID : 21aisa

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.
- The data is available from 22 Jan,2020.

Data Preprocessing:

- At first, we had discovered our data and we found that the shape of the data is Row 863 Column 14.
- There are 13 features and 1 label column (target data).
- By trying several ways to find the best data split ratio.
- We found that the feature (symptom 5 & 6 = 3 &1) Respectively .
- Except for two rows which had the value 1 and 1 respectively and the other one had its value 2 and 0 respectively, so they don't have much change in the training model, but we keep them in the training model as well .

Data Preprocessing (cont.)

- We made our models in two ways.
- The first way is without one hot encoder and the second one is with one hot encoder.
- Because some models have high performance with one hot encoding and some other models have high performance without one hot encoding, we will discuss that in detail .
- we rescale the value of the all data

The Optimal Hyperparameters

- We used GridSearchCV to find the optimal hyperparameters .
- What is grid search?
  - It is the process of performing hyperparameter tuning in order to determine the optimal values for a given model .
- How did we use it ?
  - we pass predefined values for hyperparameters to the GridSearchCV function. We do this by defining a dictionary in which we mention a particular hyperparameter along with the values it can take.

Optimal Hyperparameters (cont.)

- GridSearchCV tries all the combinations of the values passed in the dictionary and evaluates the model for each combination using the Cross-Validation method.
- After using this function, we get accuracy / loss for every combination of hyperparameters, and we can choose the one with the best performance.

1. K-Nearest Neighbors

	KNN without One Hot Encoding	**KNN with** One Hot Encoding
Data Splitting	Training = 90 % / Testing = 10 %	Training = 90 % / Testing = 10 %
Hyperparameters	N_Neighbors = 3	N_Neighbors = 3
	Metric = 'minkowski'	Metric = 'minkowski'
	P = 2	P = 2
	weights='distance'	weights='distance'

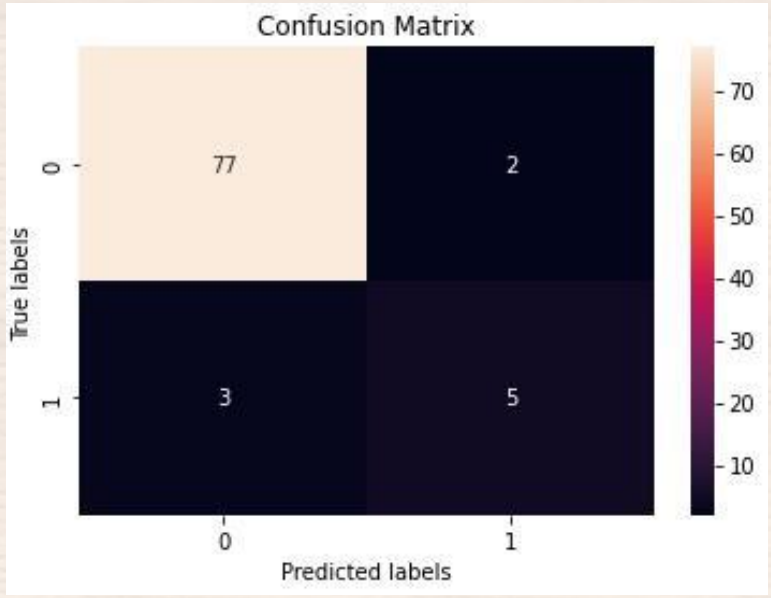
1. K-Nearest Neighbors

Performance	KNN without One Hot Encoding	KNN with One Hot Encoding
Accuracy for training	100 %	100 %
Accuracy for testing	94 %	94 %
Precision	0 => 96 % / 1 => 71 %	0 => 96 % / 1 => 71 %
Recall	0 = > 97 % / 1 => 62 %	0 = > 97 % / 1 => 62 %
F1-score	0 => 97 % / 1 => 67 %	0 => 97 % / 1 => 67 %
ROC/AUC	98 %	91 %

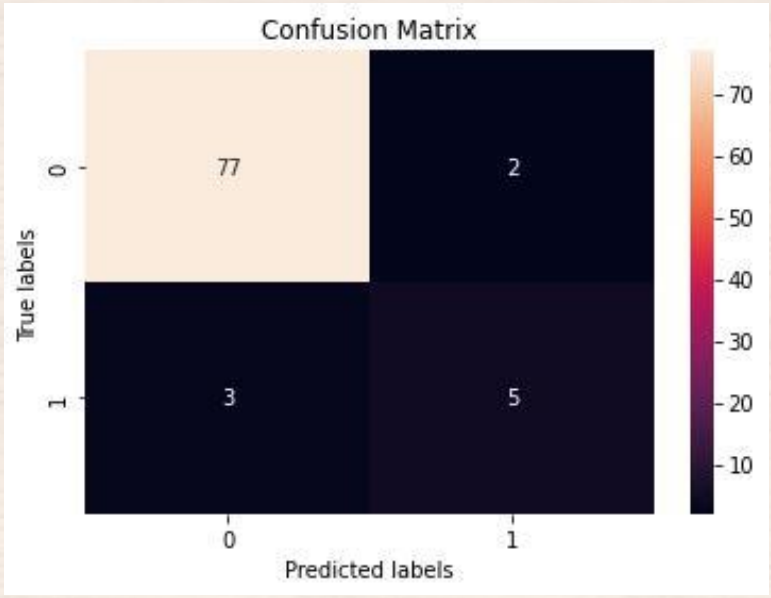
The Best Model : KNN without One Hot Encoding

Confusion Matrix

KNN without One Hot Encoding

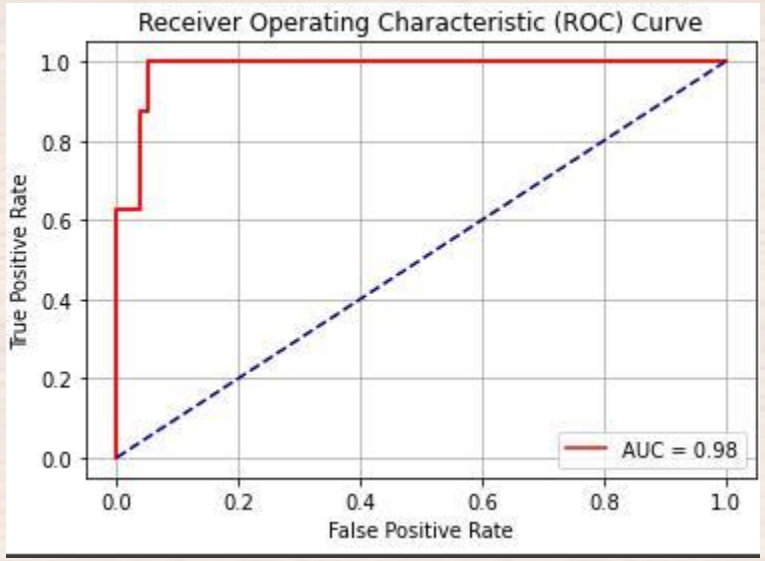


KNN with One Hot Encoding

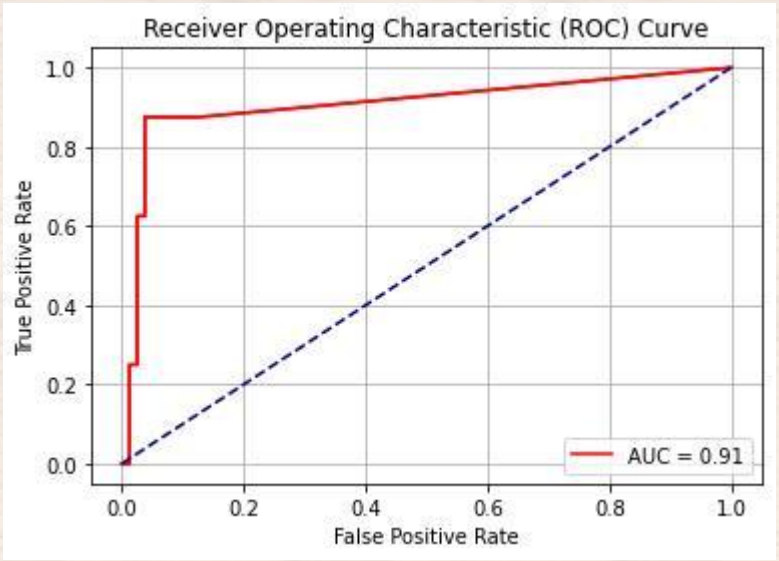


The ROC Curve

KNN without One Hot Encoding



KNN with One Hot Encoding



2. Logistic Regression

	LR without One Hot Encoding	**LR with** One Hot Encoding
Data Splitting	Training = 90 % / Testing = 10 %	Training = 84 % / Testing = 16 %
Hyperparameters	penalty = 'l1'	penalty = 'l2'
	solver = 'liblinear'	solver = 'newton-cg'
	C = 5	C = 1
	random_state=1	random_state=3
	max_iter = 500	max_iter = 500

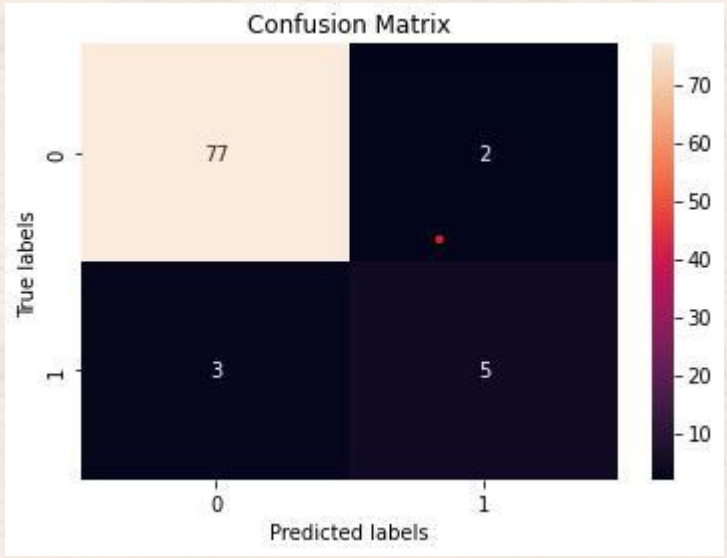
2. Logistic Regression

Performance	LR without One Hot Encoding	LR with One Hot Encoding
Accuracy for training	95 %	100 %
Accuracy for testing	94 %	96 %
Precision	0 => 96 % / 1 => 71 %	0 => 98 % / 1 => 81 %
Recall	0 = > 97 % / 1 => 62 %	0 = > 98 % / 1 => 87 %
F1-score	0 => 97 % / 1 => 67 %	0 => 98 % / 1 => 84 %
ROC/AUC	97 %	96 %

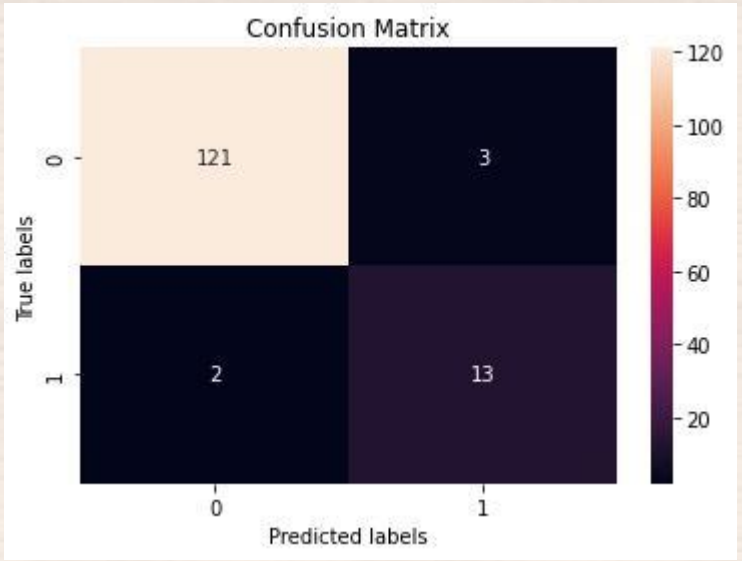
The Best Model : LR with One Hot Encoding

Confusion Matrix

LR without One Hot Encoding

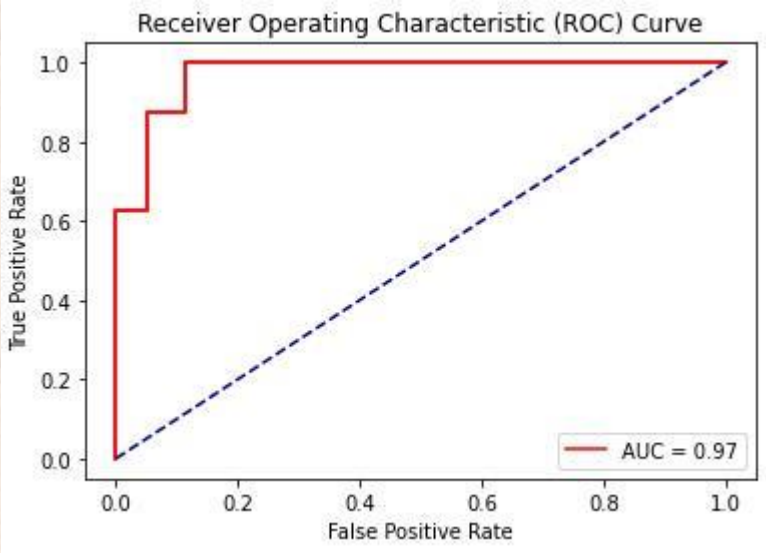


LR with One Hot Encoding

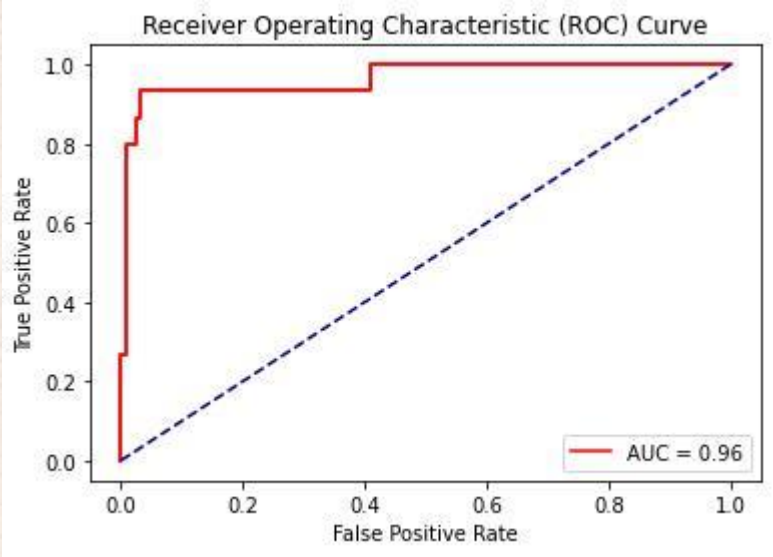


The ROC Curve

LR without One Hot Encoding



LR with One Hot Encoding



3. Naïve Bayes

	NB without One Hot Encoding	**NB with** One Hot Encoding
Data Splitting	Training = 90 % / Testing = 10 %	Training = 90 % / Testing = 10 %
Hyperparameters	alpha = 5	alpha = 1
	binarize=1,	binarize=1
	fit_prior = True	fit_prior = True

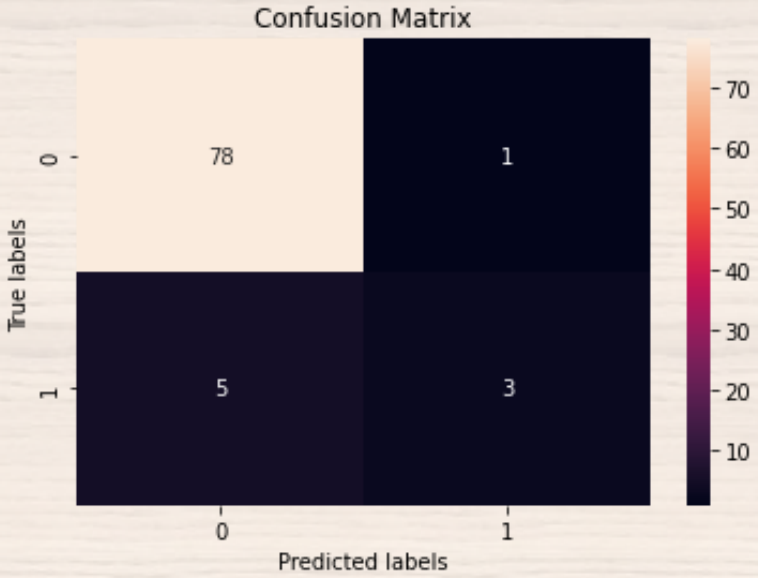
3. Naïve Bayes

Performance	NB without One Hot Encoding	NB with One Hot Encoding
Accuracy for training	90 %	94.2 %
Accuracy for testing	93 %	91 %
Precision	0 => 94 % / 1 => 75 %	0 => 93 % / 1 => 67 %
Recall	0 = > 99 % / 1 => 38 %	0 = > 97 % / 1 => 40 %
F1-score	0 => 96 % / 1 => 50 %	0 => 95 % / 1 => 50 %
ROC/AUC	95 %	90 %

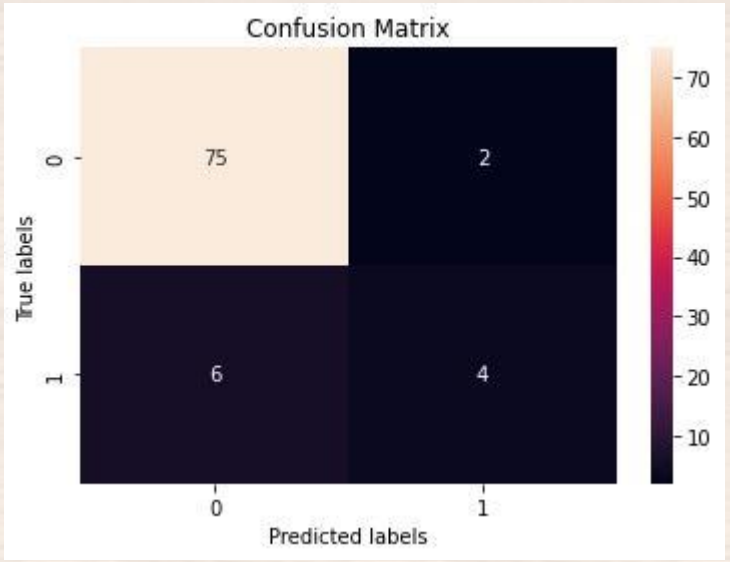
The Best Model : NB without One Hot Encoding

Confusion Matrix

NB without One Hot Encoding

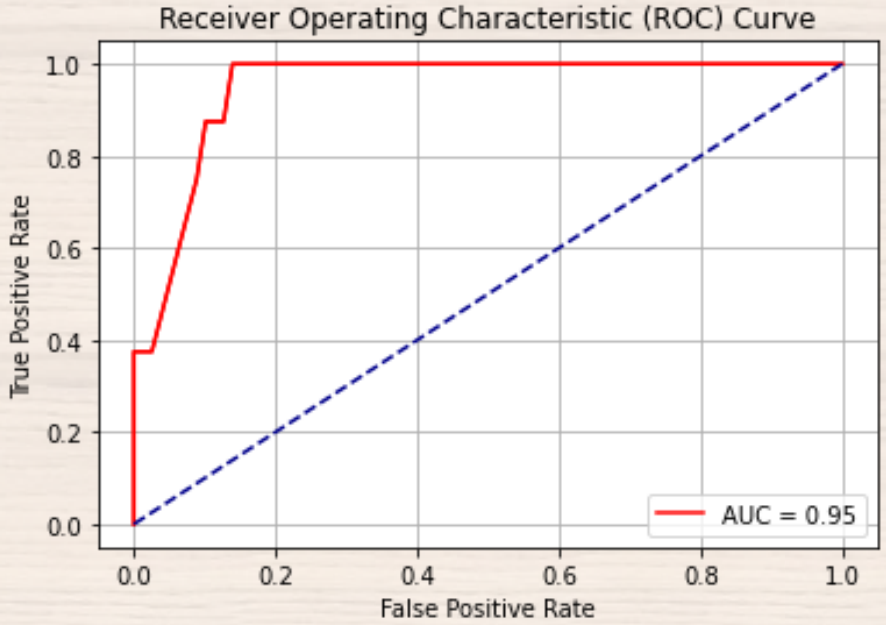


NB with One Hot Encoding

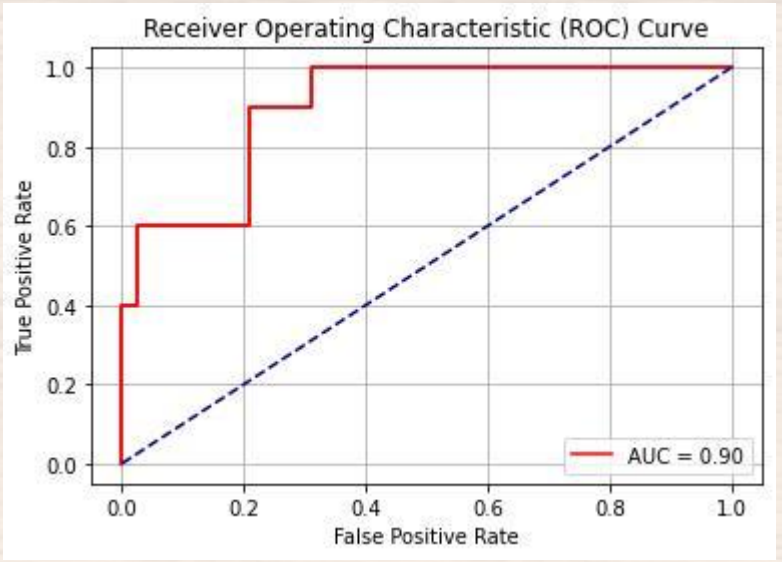


The ROC Curve

NB without One Hot Encoding



NB with One Hot Encoding



4. Decision Trees

	DT without One Hot Encoding	**DT with** One Hot Encoding
Data Splitting	Training = 90 % / Testing = 10 %	Training = 90 % / Testing = 10 %
Hyperparameters	criterion= "gini"	criterion= “entropy"
	max_depth= 9	max_depth= 16
	min_samples_leaf= 2	min_samples_leaf= 1
	random_state= 3	random_state= 3

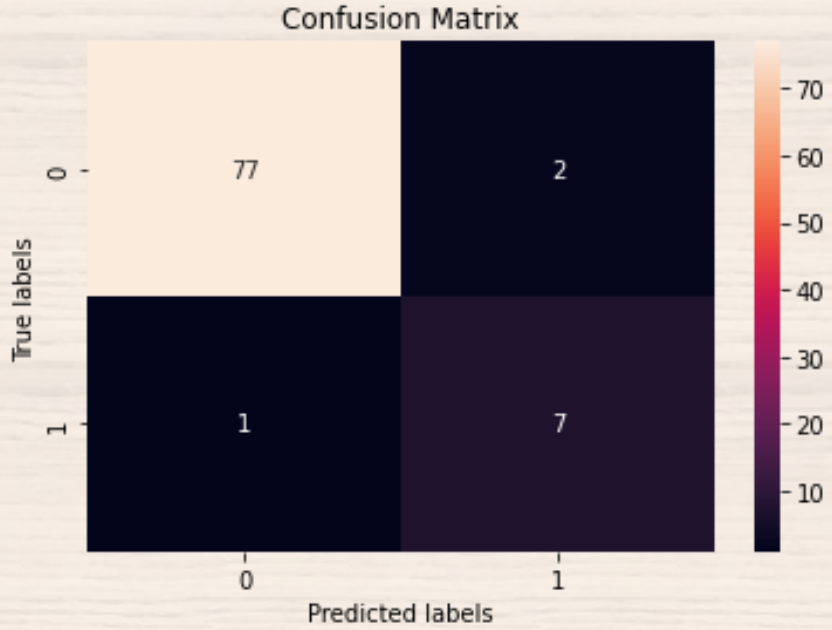
4. Decision Trees *

Performance	DT without One Hot Encoding	DT with One Hot Encoding
Accuracy for training	98 %	98 %
Accuracy for testing	97 %	97 %
Precision	0 => 99 % / 1 => 78 %	0 => 97 % / 1 => 86 %
Recall	0 = > 97 % / 1 => 88 %	0 = > 99 % / 1 => 75 %
F1-score	0 => 98 % / 1 => 82 %	0 => 98 % / 1 => 80 %
ROC/AUC	98 %	98 %

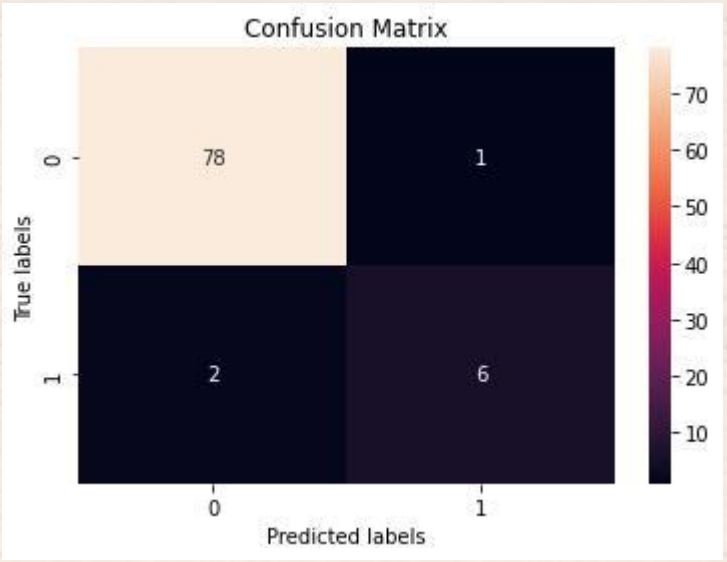
The Best Model : DT without One Hot Encoding

Confusion Matrix

DT without One Hot Encoding

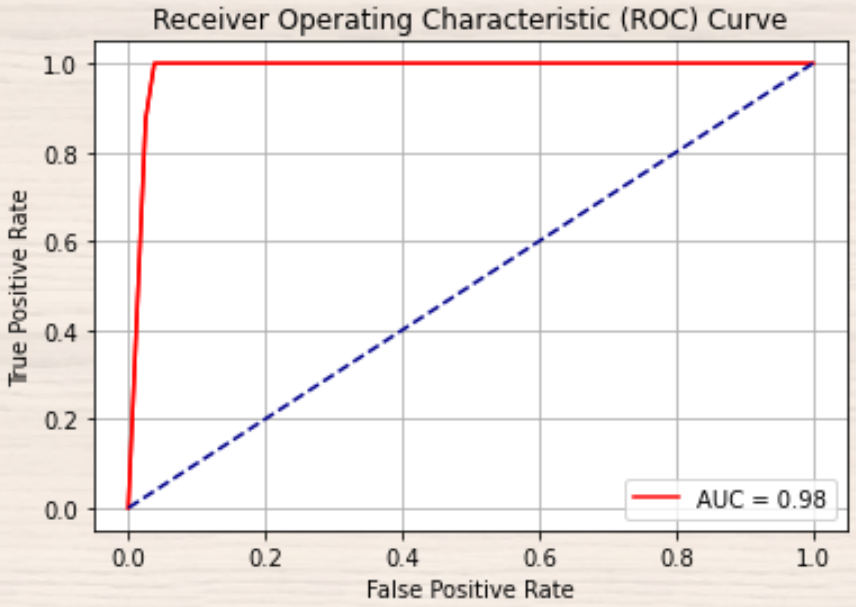


DT with One Hot Encoding

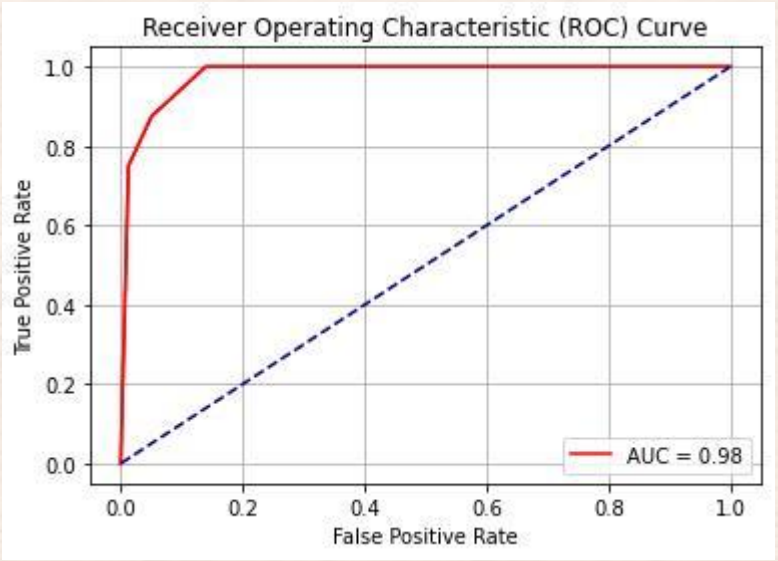


The ROC Curve

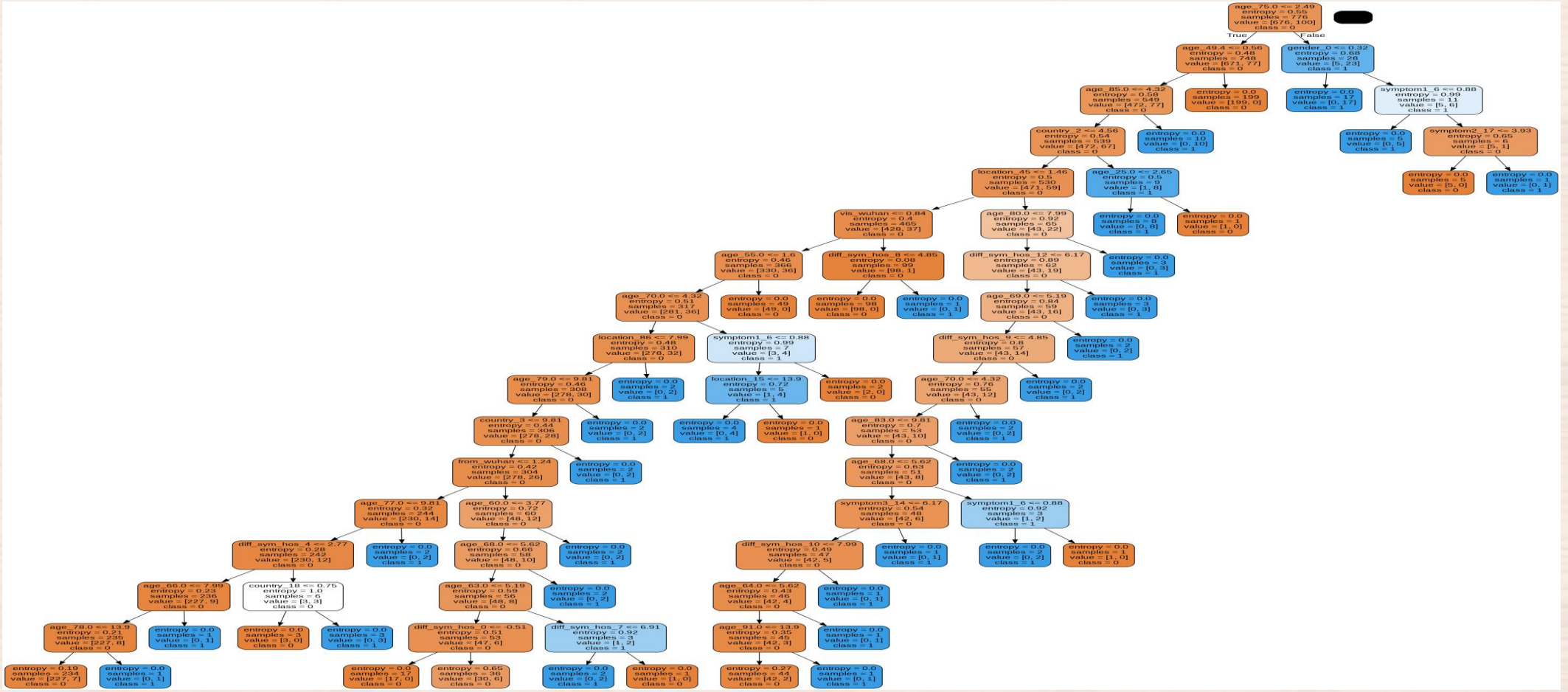
DT without One Hot Encoding



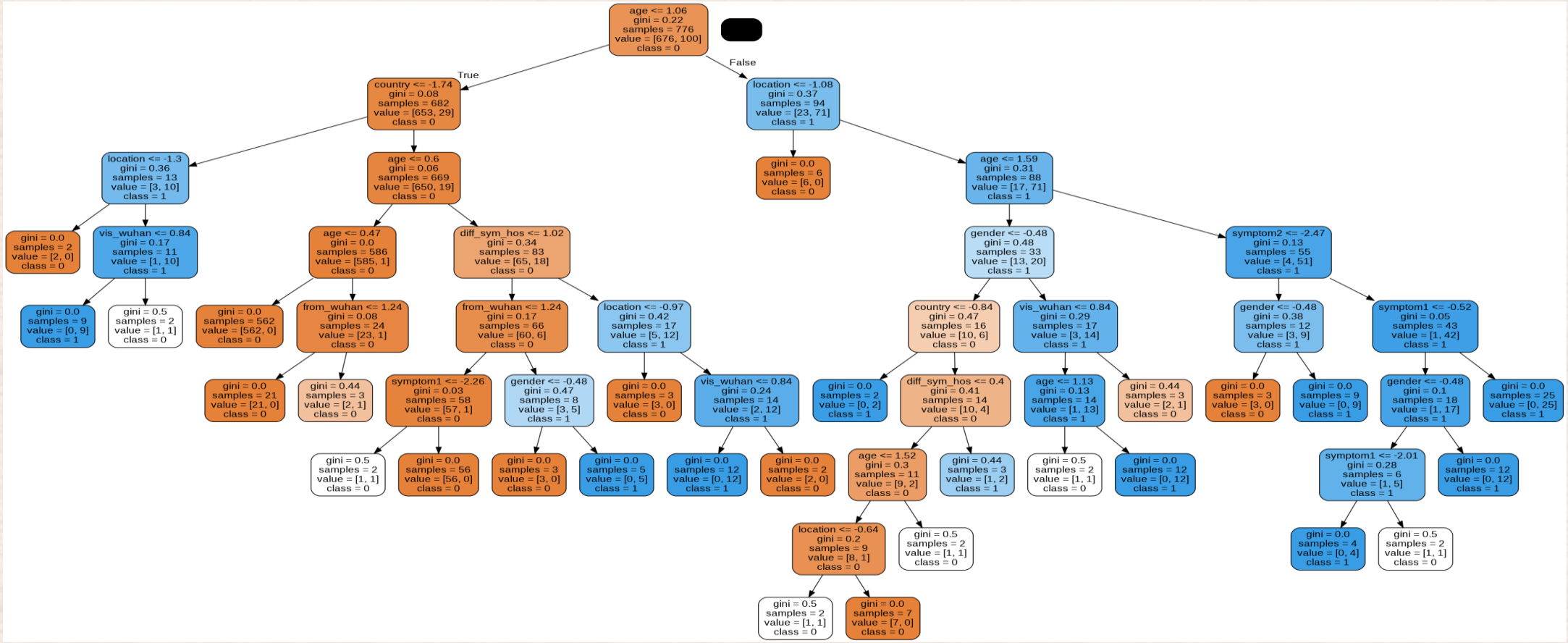
DT with One Hot Encoding



The Visualization for the Tree with Hot Encoding



The Visualization for the Tree without Hot Encoding



5. Support Vector Machines

	SVM without One Hot Encoding	**SVM with** One Hot Encoding
Data Splitting	Training = 90 % / Testing = 10 %	Training = 85 % / Testing = 15 %
Hyperparameters	kernel = 'rbf'	kernel = ‘rbf'
	C=100	C=100
	gamma=0.01	gamma=0.001
	random_state = 0	random_state = 3
	probability=True	probability=True

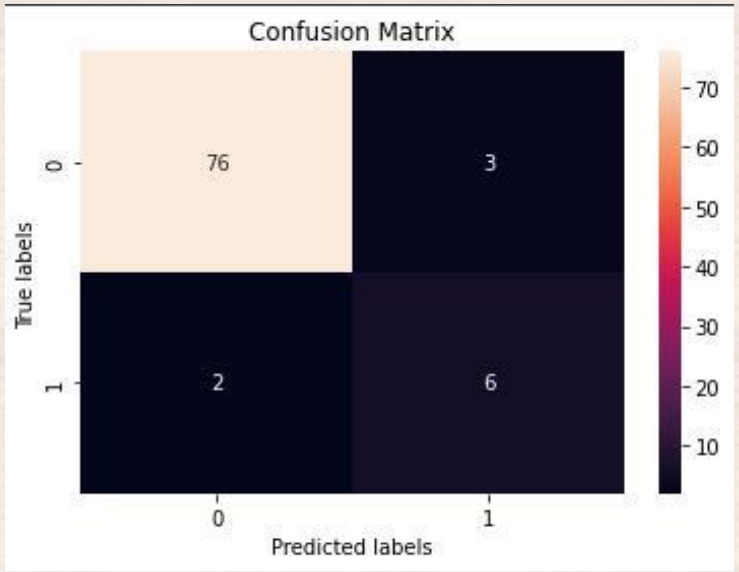
5. Support Vector Machines

Performance	SVM without One Hot Encoding	SVM with One Hot Encoding
Accuracy for training	98 %	100 %
Accuracy for testing	94 %	96 %
Precision	0 => 97 % / 1 => 67 %	0 => 98 % / 1 => 81 %
Recall	0 = > 96 % / 1 => 75 %	0 = > 97 % / 1 => 87 %
F1-score	0 => 97 % / 1 => 71 %	0 => 98 % / 1 => 84 %
ROC/AUC	98 %	98 %

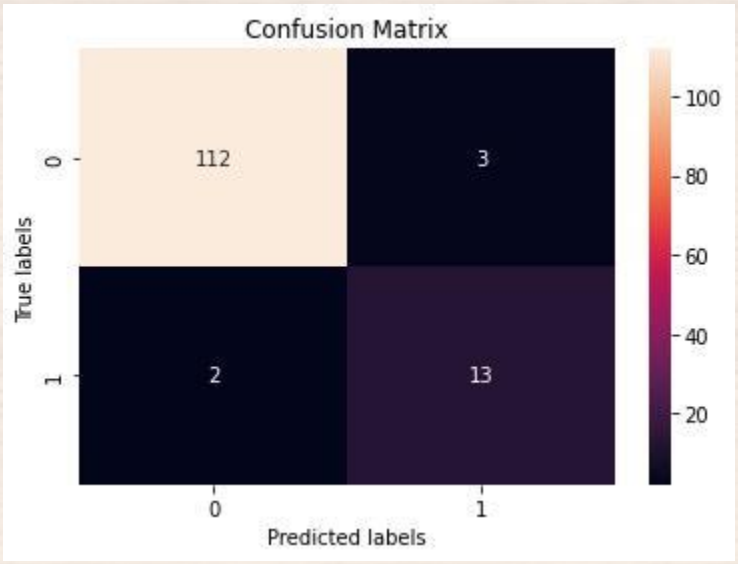
The Best Model : SVM with one hot encoding

Confusion Matrix

SVM without One Hot Encoding

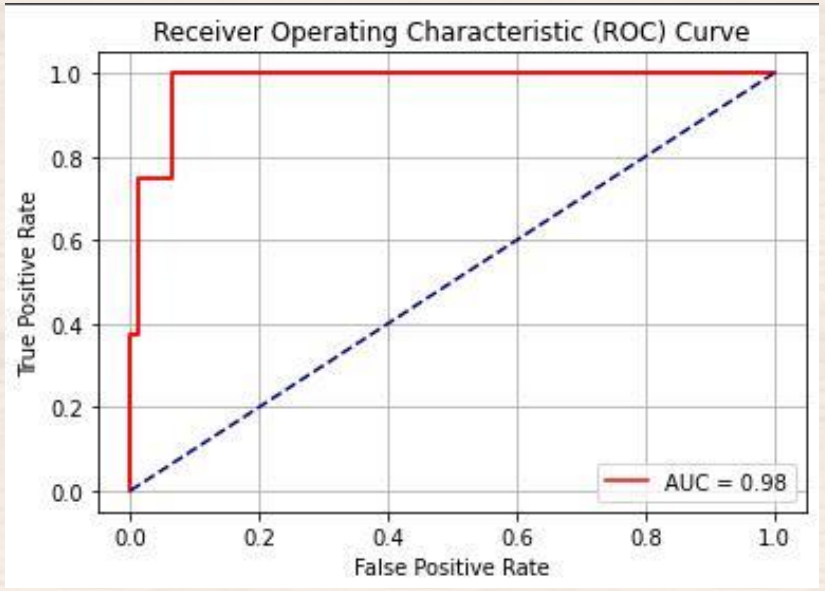


SVM with One Hot Encoding

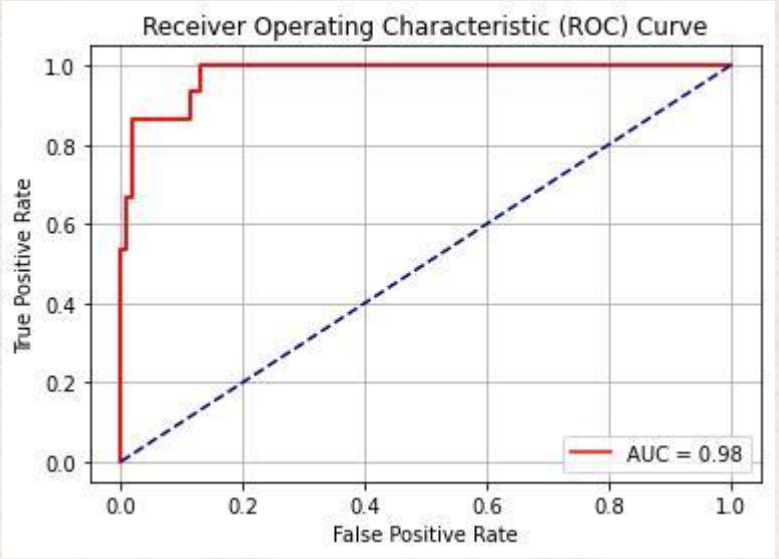


The ROC Curve

SVM without One Hot Encoding



SVM with One Hot Encoding



The Best Model

The good news is that all models are above 90% accuracy.

but we found that the **Decision Tree (without one hot encoding)** model was the highest one.

The Models From Scratch

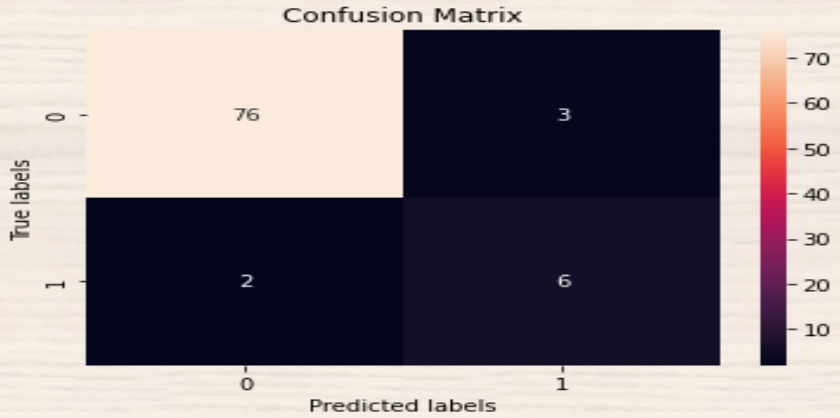
Unfortunately in this part we were not able to build the SVM model from scratch but we mentioned the source of the code that we found.

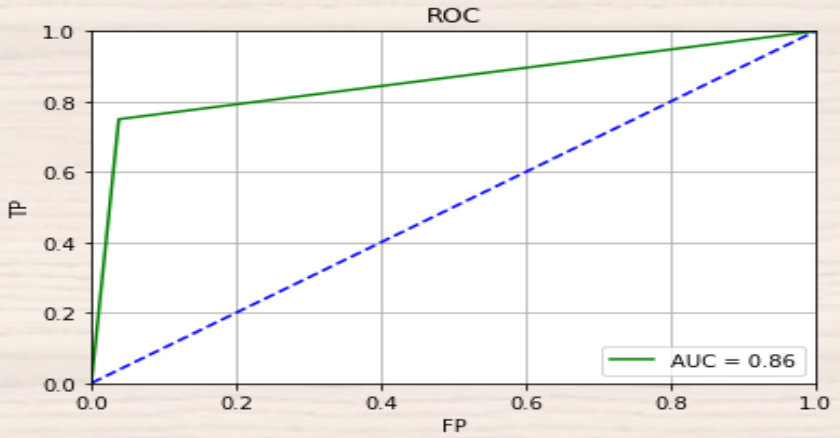
But the good news is that we built the rest of the four models from scratch.

This is done with the help of some of the resources we mentioned in the Resources slide.

1. KNN From Scratch

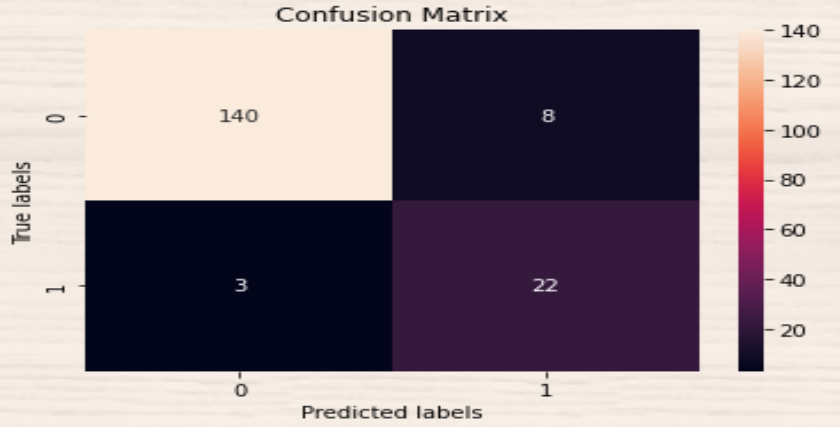
Performance	KNN without One Hot Encoding
Accuracy for testing	94 %
Precision	0 => 97 % / 1 => 67 %
Recall	0 = > 96 % / 1 => 75 %
F1-score	0 => 97 % / 1 => 71 %
ROC/AUC	86 %

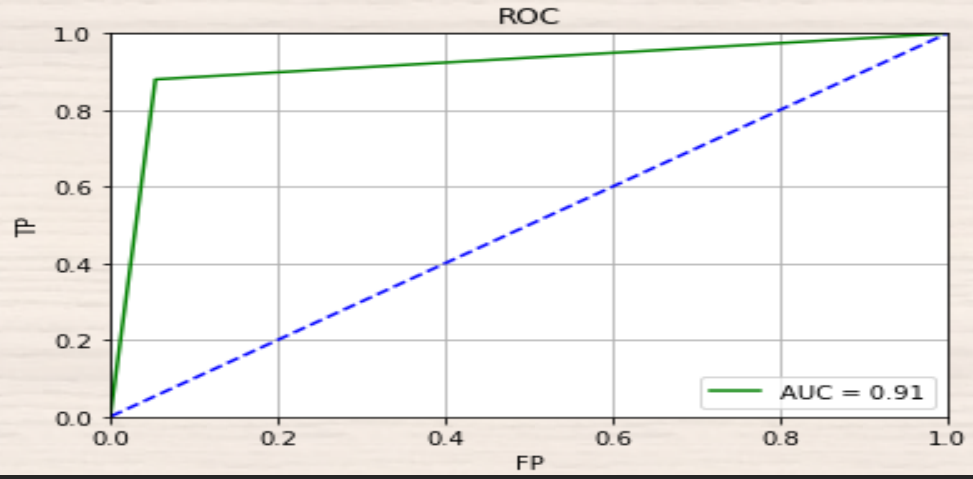




2. Decision Trees From Scratch

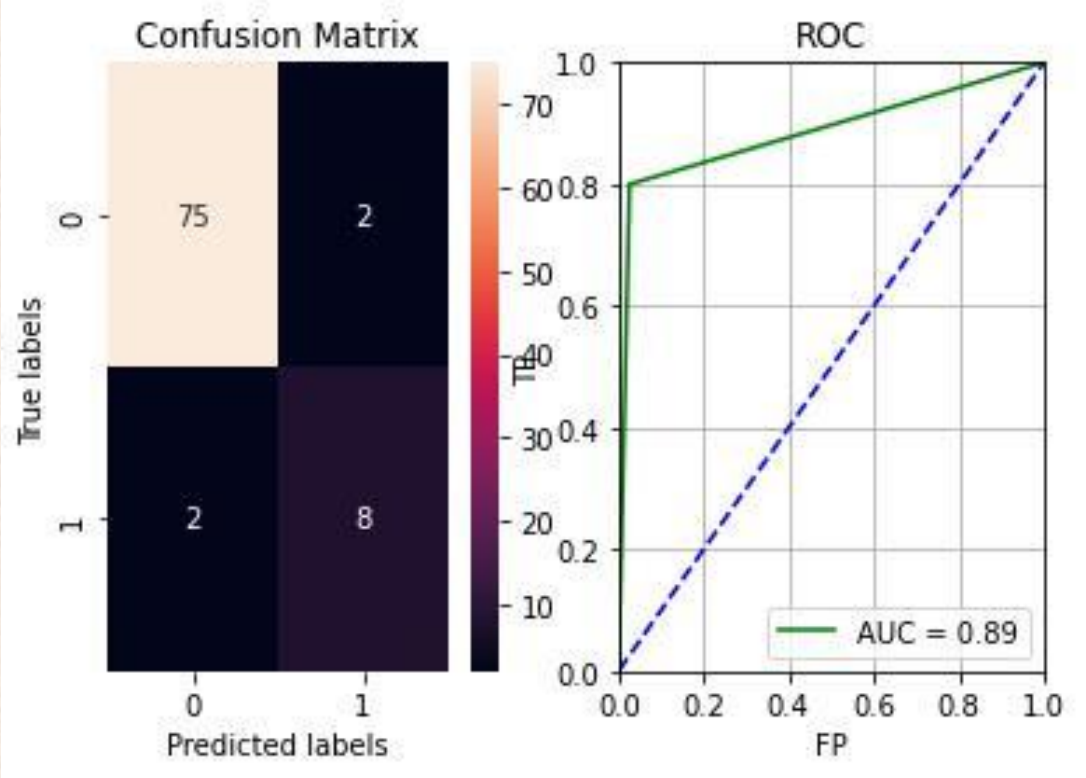
Performance	DT without One Hot Encoding
Accuracy for testing	94 %
Precision	0 => 98 % / 1 => 73 %
Recall	0 = > 95 % / 1 => 88 %
F1-score	0 => 96 % / 1 => 80 %
ROC/AUC	91 %





3. Logistic Regression From Scratch

Performance	LR without One Hot Encoding
Accuracy for testing	95 %
Precision	0 => 97 % / 1 => 80 %
Recall	0 = > 97 % / 1 => 80 %
F1-score	0 => 97 % / 1 => 80 %
ROC/AUC	89 %



4. Naïve Bayes

The Accuracy for this model was 85%

5. SVM

Unfortunately The Accuracy for this model was very bad.

Resources

Sklearn Documentation

The Github of mlfromscratch

Stackoverflow

scikit-learn package