

Introduction to Pattern Recognition

**Wisconsin Diagnostic Breast Cancer (WDBC)**

**Yasin Samet Karaca**

January 2023

**Wisconsin Diagnostic Breast Cancer (WDBC)**

Abstract: Diagnostic Wisconsin Breast Cancer Database

|  |  |  |  |
| --- | --- | --- | --- |
| **Data Set Characteristics:** | Multivariate | **Number of Instances:** | 569 |
| **Attribute Characteristics:** | Real | **Number of Attributes:** | 32 |
| **Associated Tasks:** | Classification | **Missing Values?** | No |

**Predict whether the cancer is benign or malignant.**

**Problem**: Cancer is one of the leading causes of human death in the world and has caused the death of approximately 9.6 million people in 2018. Breast cancer is the most common cause of cancer deaths in women. However, breast cancer is a type of cancer that can be treated when diagnosed early. The aim of this study is to identify cancer early in life by using machine learning methods.

Predicting field 2, diagnosis: B = benign, M = malignant. Sets are linearly separable using all 30 input features. Best predictive accuracy obtained using one separating plane in the 3-D space of Worst Area, Worst Smoothness and Mean Texture.

Features are computed from a digitized image of a fine needle aspirate (FNA) of a breast mass. They describe characteristics of the cell nuclei present in the image. A few of the images can be found at http://www.cs.wisc.edu/~street/images/

Separating plane described above was obtained using Multisurface Method-Tree (MSM-T) [K. P. Bennett, "Decision Tree Construction Via Linear Programming." Proceedings of the 4th Midwest Artificial Intelligence and Cognitive Science Society, pp. 97-101, 1992], a classification method which uses linear programming to construct a decision tree. Relevant features were selected using an exhaustive search in the space of 1-4 features and 1-3 separating planes. The actual linear program used to obtain the separating plane in the 3-dimensional space is that described in: [K. P. Bennett and O. L. Mangasarian: "Robust Linear Programming Discrimination of Two Linearly Inseparable Sets", Optimization Methods and Software 1, 1992, 23-34]. This database is also available through the UW CS ftp server:

-ftp ftp.cs.wisc.edu

-cd math-prog/cpo-dataset/machine-learn/WDBC/

**Number of instances:** 569

**Number of attributes:** 32 (ID, diagnosis, 30 real-valued input features)

**Attribute information**

1) ID number

2) Diagnosis (M = malignant, B = benign)

3-32

Ten real-valued features are computed for each cell nucleus:

a) radius (mean of distances from center to points on the perimeter)

b) texture (standard deviation of gray-scale values)

c) perimeter

d) area

e) smoothness (local variation in radius lengths)

f) compactness (perimeter^2 / area - 1.0)

g) concavity (severity of concave portions of the contour)

h) concave points (number of concave portions of the contour)

i) symmetry

j) fractal dimension ("coastline approximation" - 1)

Several of the papers listed above contain detailed descriptions ofhow these features are computed.

The mean, standard error, and "worst" or largest (mean of the three largest values) of these features were computed for each image, resulting in 30 features. For instance, field 3 is Mean Radius, field 13 is Radius SE, field 23 is Worst Radius.

All feature values are recoded with four significant digits.

Missing attribute values: none

**Class distribution**: 357 benign, 212 malignant

RangeIndex: 569 entries, 0 to 568

dtypes: float64(30), int32(1)

I wanted to show the analysis of the best percentage result using the given data set using the Python program in the project. In order to achieve the best success result, it is aimed to reach the best percentage result by trying different algorithms and different parameters within these algorithms. Although some algorithms work fast, they give low percentage results, while some slower algorithms give high percentage results. With this project it is expected to have the highest possible correct classification percentage. To achieve this, cross validation was performed on the feature selection. I made classification with selected attributes.

The algorithms I use are as follows.

- AdaBoost Classifier

- Logistic Regression

- Gradient Booster Algorithm

- K-Nearest Neighbours

- Random Forest Classifier

- Naive Bayes

- Decision Tree

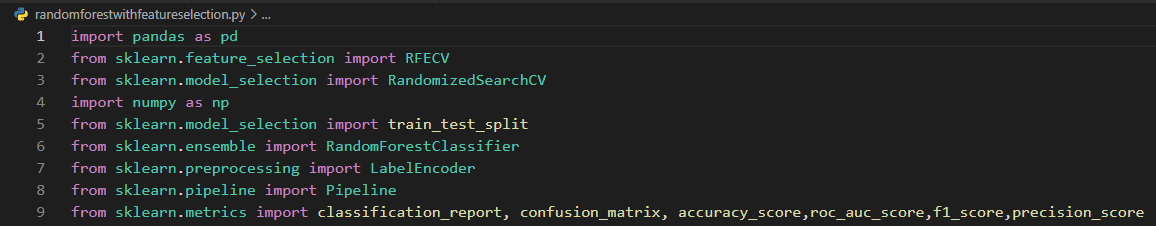
- Artificial neural network

- Support Vector Machine

**Random Forest Classifier with Feature Selection**

Firstly, I use Random Forest Classifier Algorithm.

**PluginS that we need to uSe**

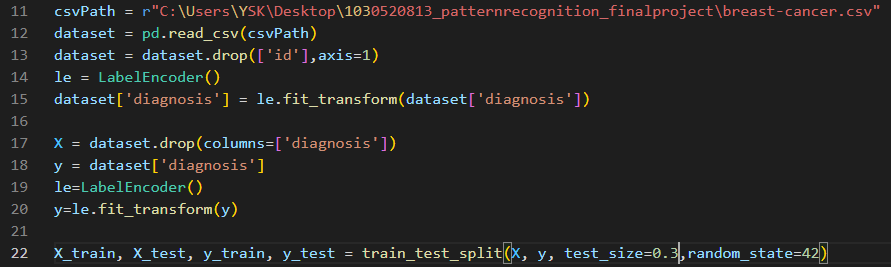
****

Here, need to set a path to read csv file.

Random\_state: Controls the shuffling applied to the data before applying the split.

Test size = 0.3

Benign and Malignant results are converted to numbers 0 and 1 with the Label encoder.



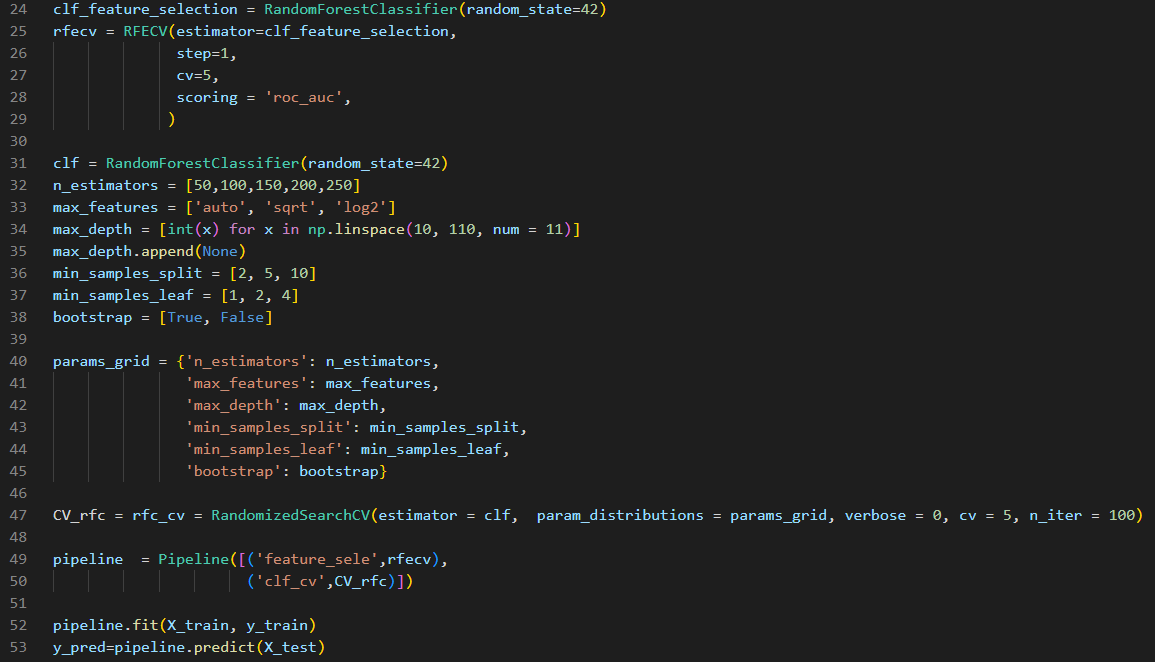
RFECV (Recursive Feature Elimination with Cross-Validation) performs recursive feature elimination with cross-validation loop to extract the optimal features. Scikit-learn provides RFECV class to implement RFECV method to find the most important features in a given dataset.

RandomizedSearchCV implements a “fit” and a “score” method. It also implements “score\_samples”, “predict”, “predict\_proba”, “decision\_function”, “transform” and “inverse\_transform” if they are implemented in the estimator used. The parameters of the estimator used to apply these methods are optimized by cross-validated search over parameter settings. In contrast to GridSearchCV, not all parameter values are tried out, but rather a fixed number of parameter settings is sampled from the specified distributions. The number of parameter settings that are tried is given by n\_iter. If all parameters are presented as a list, sampling without replacement is performed. If at least one parameter is given as a distribution, sampling with replacement is used. It is highly recommended to use continuous distributions for continuous parameters.

Basically, I want to fine tune the hyper parameter of our classifier (with Cross validation) after feature selection using recursive feature elimination (with Cross validation).

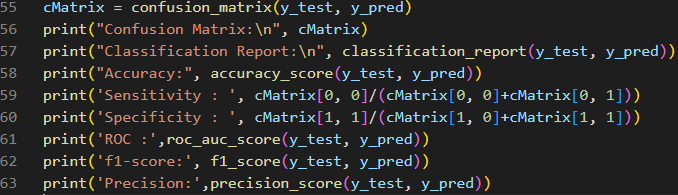
Pipeline object is exactly meant for this purpose of assembling the data transformation and applying estimator. It would be possible with the following approach

The purpose of the pipeline is to assemble several steps that can be cross-validated together while setting different parameters.



Cross validation = 5

So right now I have test and predicdion data. I use them to create Confusion Matrix for classificafion report and accuracy.



I have everything we need to find Sensitivity, auc and Specificity. Now take a look at the formula.

**Sensitivity = (True Positive)/(True Positive + False Negative)**

**Specificity = (True Negative)/(True Negative + False Positive)**

To find out how can we find true positive, true negative, false positive and false negative we can use this table.



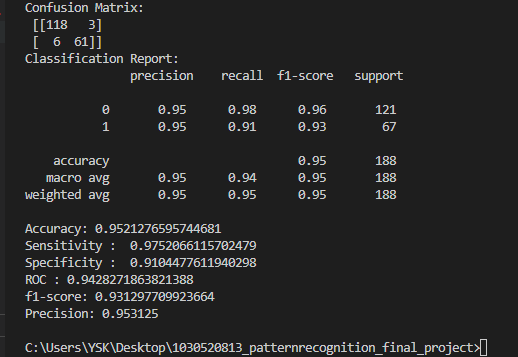
**Sensitivity = cMatrix[0, 0]/(cMatrix[0, 0]+cMatrix[0, 1])**

**Specificity = cMatrix[1, 1]/(cMatrix[1, 0]+cMatrix[1, 1])**

**Precision = TP/(TP+FP)**

**F1 Score = 2\*(Recall \* Precision) / (Recall + Precision)**

**Converted to numbers 0 and 1 to malignant and benign results with Label encoder**

**ClaSSification Report: **

**Random Forest Classifier with My Cross Validation in Feature Selection**

I used Stratified K-Fold cross-validation to evaluate the attribute selection

**Steps**

Importing Libraries and reading dataset.

Data preprocessing. Dividing the Data and Encoding the Labels.

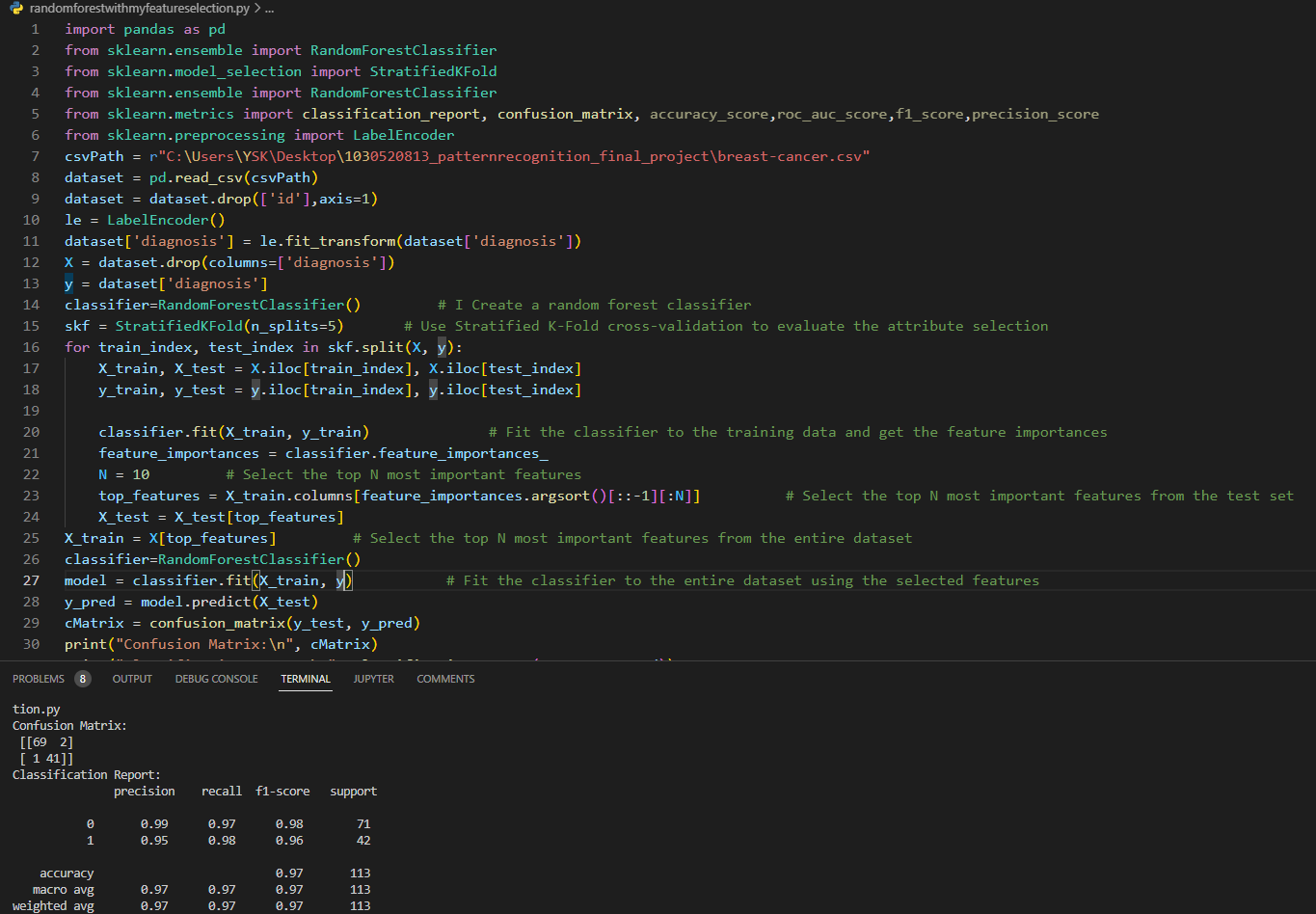
Create a random forest classifier

Use Stratified K-Fold cross-validation to evaluate the attribute selection

Fit the classifier to the training data and get the feature importances

Select the top N most important features from the entire dataset

Fit the classifier to the entire dataset using the selected features



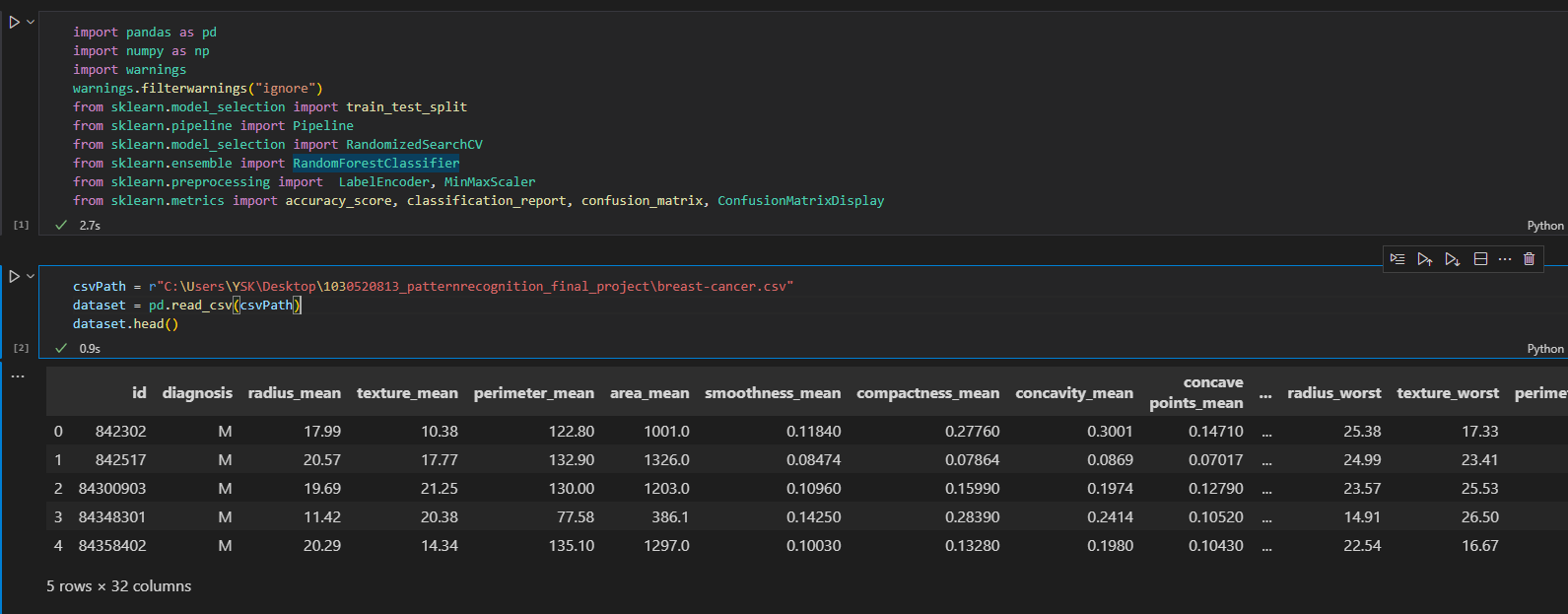
Cross validation= 5

**Random Forest Classifier without Feature Selection**

Random Forest is a supervised machine learning algorithm which is based on ensemble learning.

**Steps**

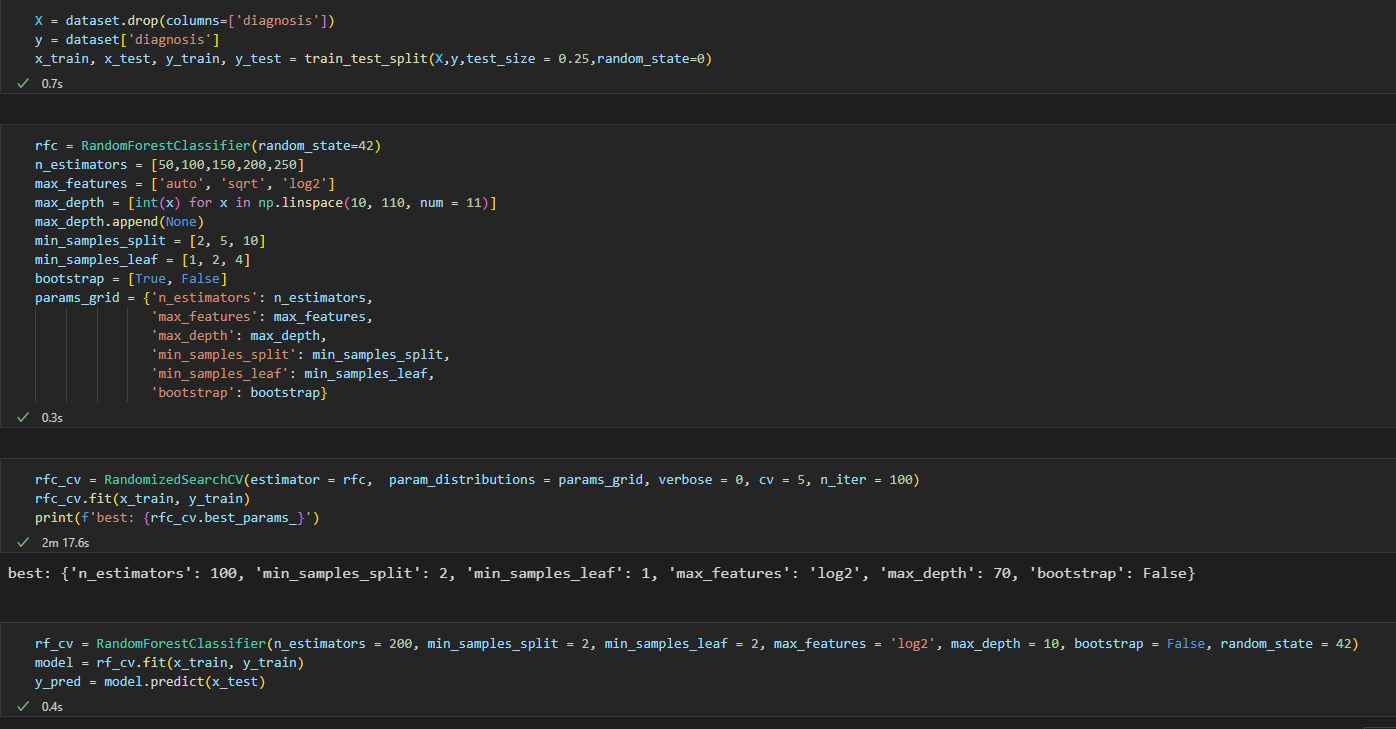
Importing Libraries and reading dataset.

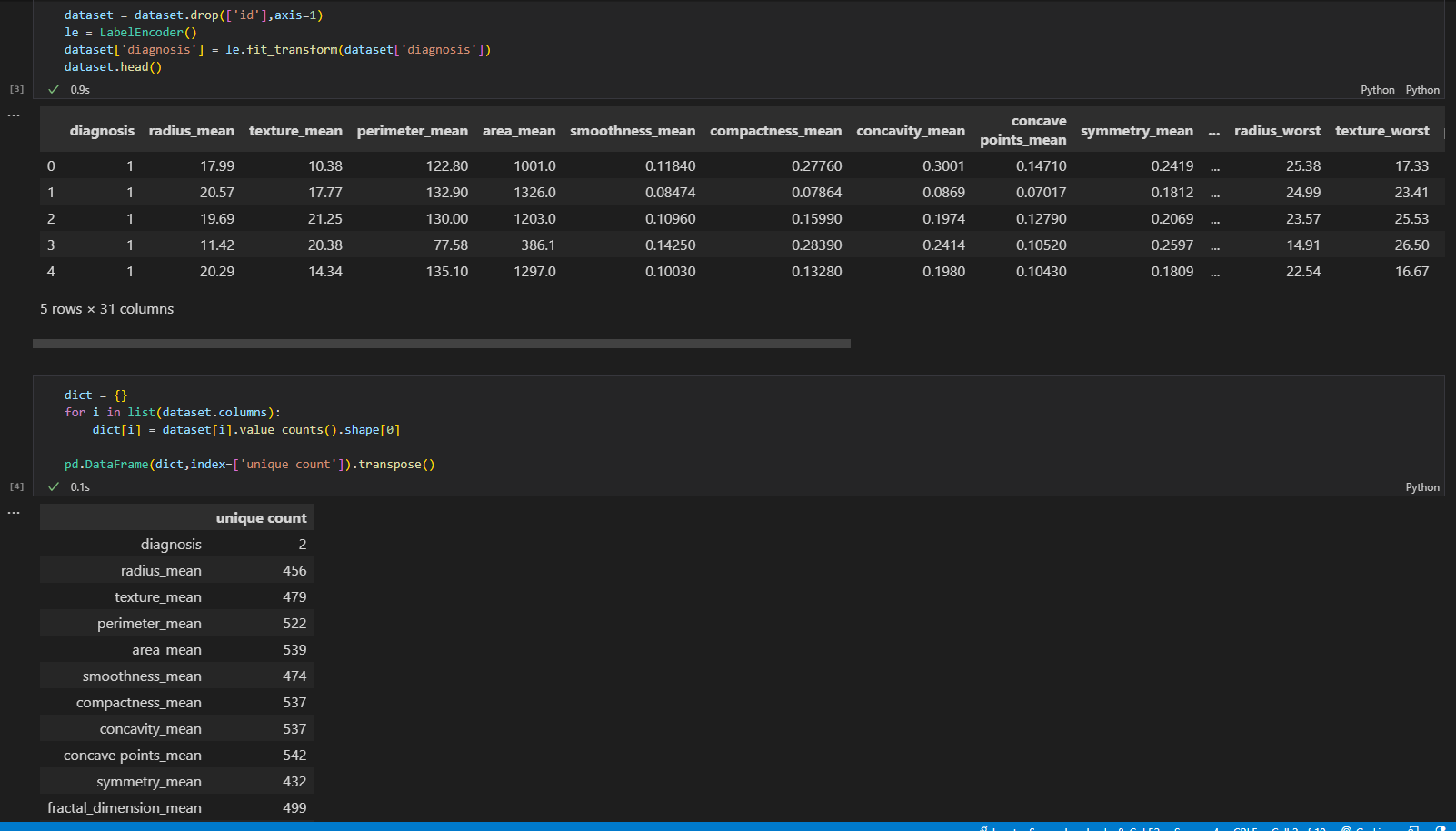
dataset.head() 

Data preprocessing. Dividing the Data and Encoding the Labels.

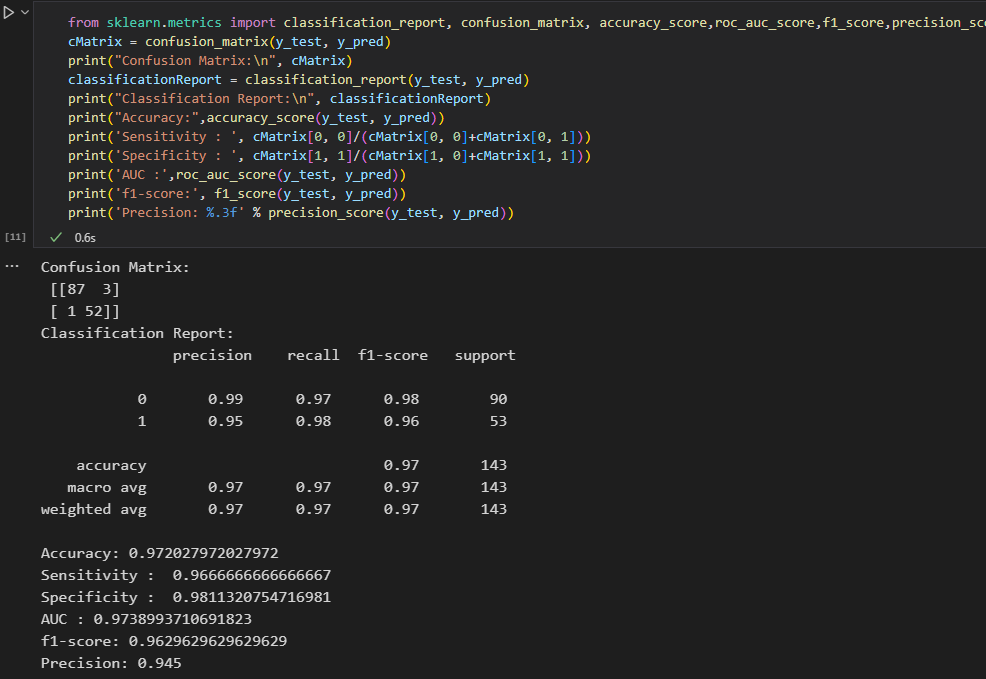
Splitting dataset into Training and Testing Set.

Using RandomizedSearchCV and Printing the results.



****

**Classification report:**

****

**K Nearest Neighbours without Feature Selection**

KNN classifier is one of the most sophisticated and widely used classification algorithm. Some of the features that make this algorithm so popular are as mentioned below:

KNN does not make any underlying assumption about the data.

It gives the user the flexibility to choose the distance measure metric.

It has a relatively higher accuracy than many classification algorithms.

With the addition of more data points, the classifier constantly evolves and is capable of quickly adapting to the changes in input dataset.

**Standart Scaler** : Standardize features by removing the mean and scaling to unit variance.

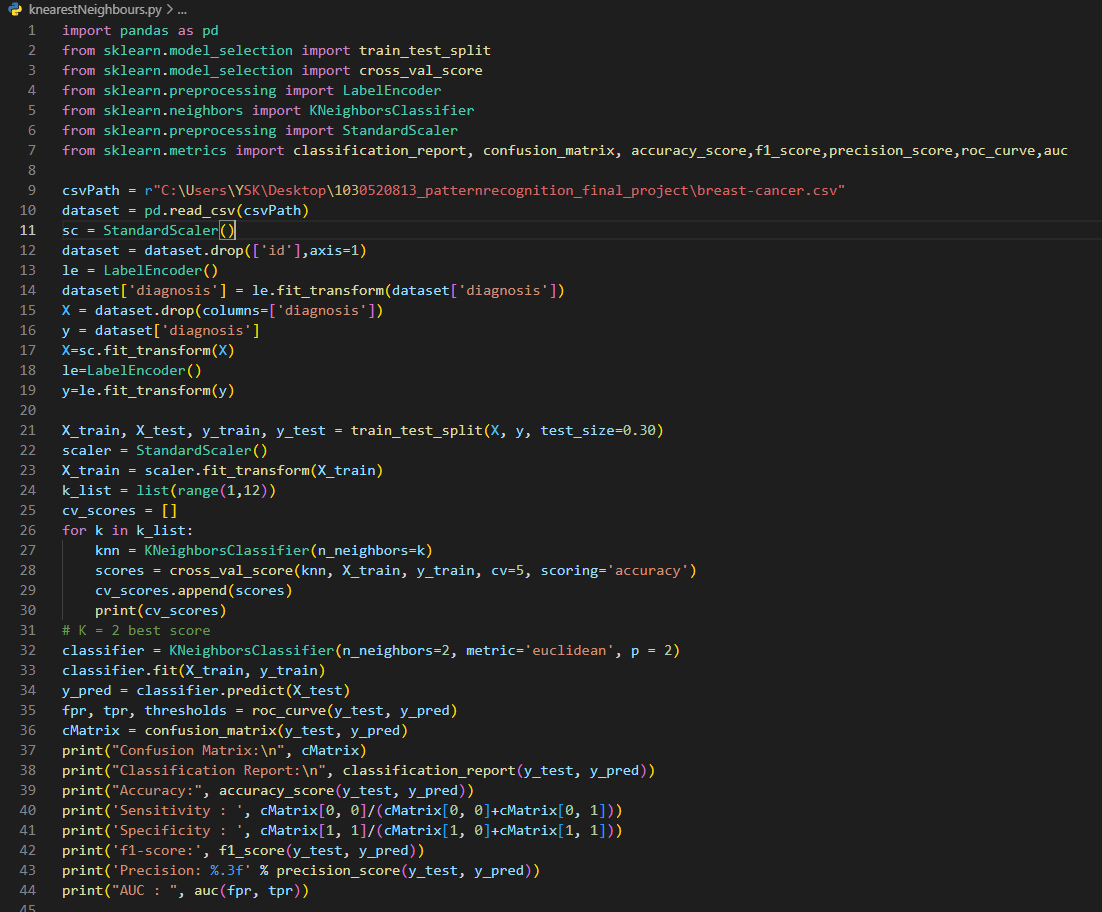
**Steps**

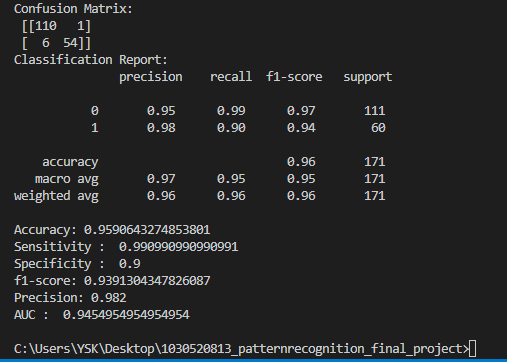
Importing the Required Libraries and Loading the Dataset

Dividing the Data and Encoding the Labels (Label Encoding)

Splitting the Dataset, Feature Scaling and Fitting the Model

Evaluating the Predictions and Cross Validating

****



**K Nearest Neighbours With Feature Selection**

Sequential Feature Selector adds (forward selection) or removes (backward selection) features to form a feature subset in a greedy fashion. At each stage, this estimator chooses the best feature to add or remove based on the cross-validation score of an estimator.

We have added our features to another csv file so that our algorithm can make feature selection.

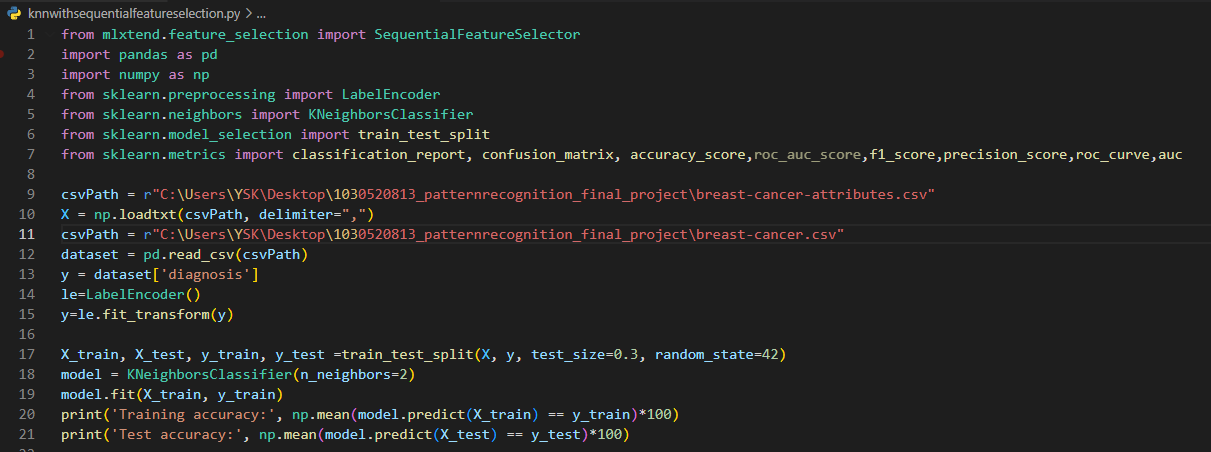
**Steps**

Import dataset and libraries

Drop Duplicate columns.

Split data into separate training and test set

Feature Scaling and engineering



**Training accuracy: 94.28571428571428**

**Test accuracy: 93.8596491228070**

Perform feature selection using the KNN algorithm

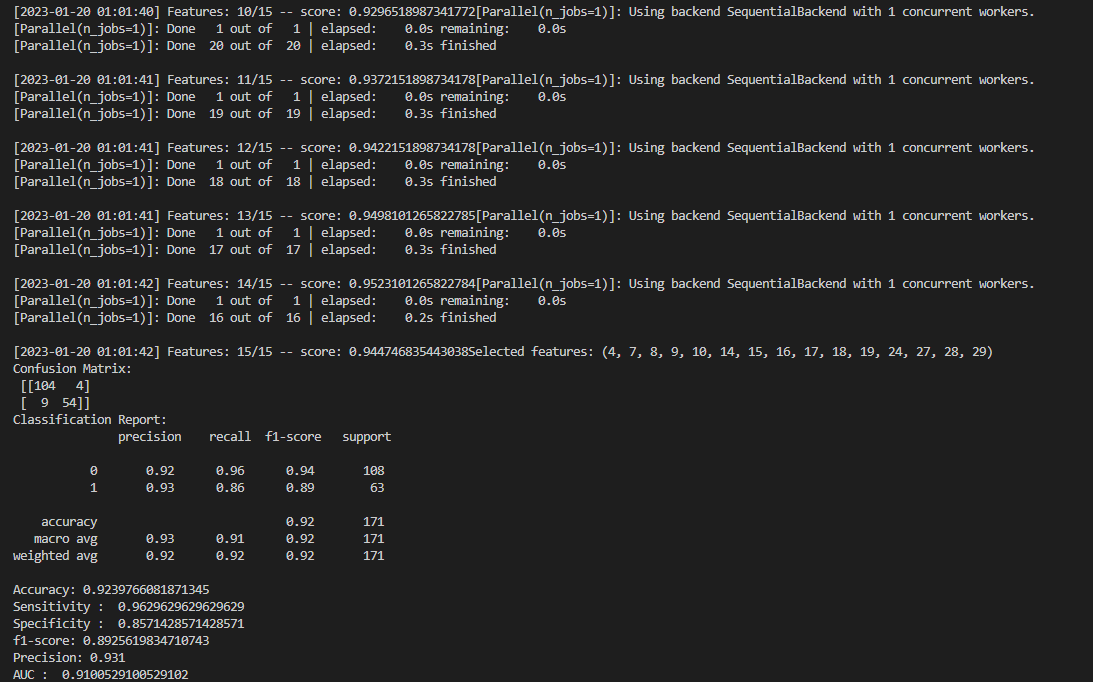
Print the selected features

Train a KNN classifier on the selected features

Make predictions on the test set

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu



**Ada Boost Classifier With Feature Selection**

It combines multiple classifiers to increase the accuracy of classifiers. AdaBoost is an iterative ensemble method. AdaBoost classifier builds a strong classifier by combining multiple poorly performing classifiers so that you will get high accuracy strong classifier.

GridSearchCV is the process of performing hyperparameter tuning in order to determine the optimal values for a given model. As mentioned above, the performance of a model significantly depends on the value of hyperparameters. Note that there is no way to know in advance the best values for hyperparameters so ideally, we need to try all possible values to know the optimal values. Doing this manually could take a considerable amount of time and resources and thus we use GridSearchCV to automate the tuning of hyperparameters.

It works in the following steps:

Initially, Adaboost selects a training subset randomly. It iteratively trains the AdaBoost machine learning model by selecting the training set based on the accurate prediction of the last training. It assigns the higher weight to wrong classified observations so that in the next iteration these observations will get the high probability for classification. Also, It assigns the weight to the trained classifier in each iteration according to the accuracy of the classifier. The more accurate classifier will get high weight. This process iterate until the complete training data fits without any error or until reached to the specified maximum number of estimators. To classify, perform a "vote" across all of the learning algorithms you built.

Recursive feature elimination with cross-validation to select features.

Gridsearchcv algorithm is used for parameter selection with cross validation and rfecv algorithm is used for feature selection.

**Steps**

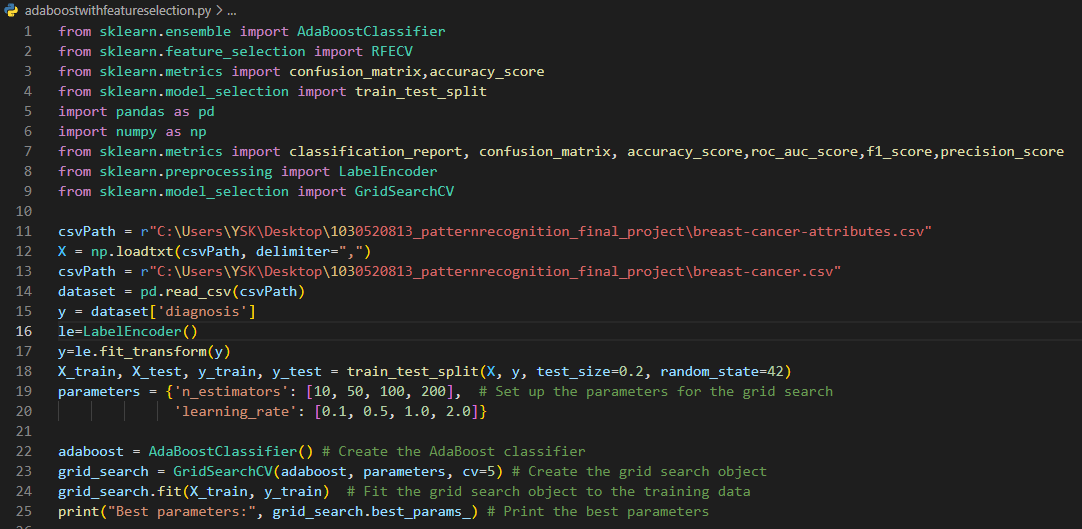
Import dataset and libraries.

Drop Duplicate columns and use label encoder.

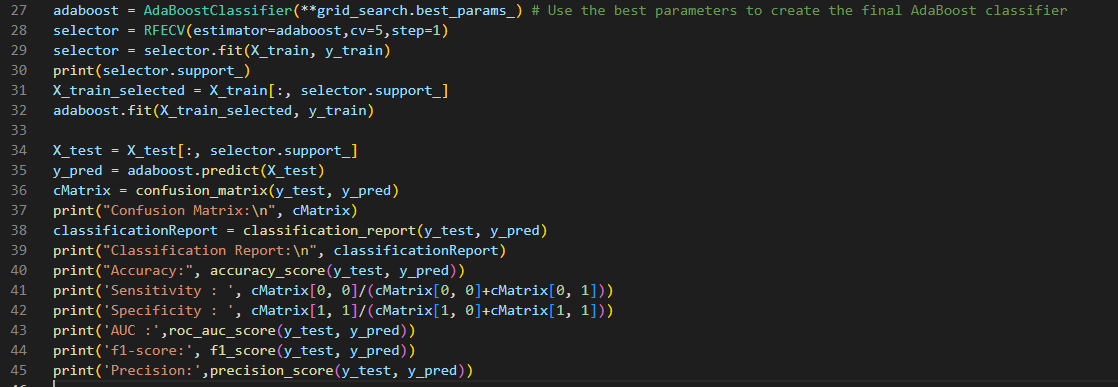
Split the data into a training set and a test set.

Create the grid search object.

Fit the grid search object to the training data and use the best parameters to create the final AdaBoost classifier.



Best parameters: {'learning\_rate': 0.1, 'n\_estimators': 200}



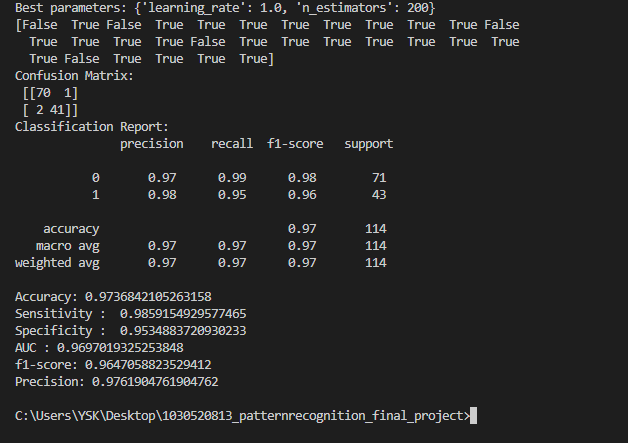
Use RFECV to select the best attributes. Print the selected attributes

Get the selected features. Train the AdaBoost classifier on the selected features

Make predictions on the test set

Calculate the accuracy of the predictions

Report:



**Logistic Regression With My Cross Validation in Feature Selection**

Logistic Regression is a classification technique used in machine learning. It uses a logistic function to model the dependent variable.

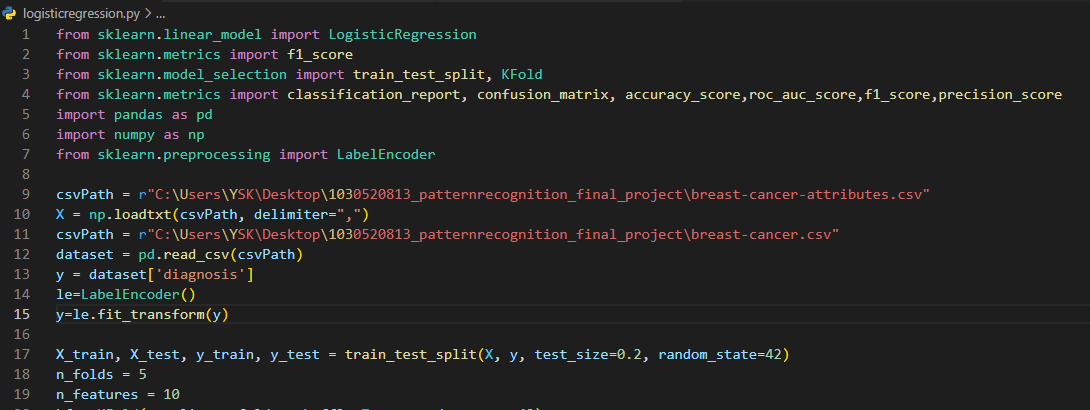
**Steps**

Load the dataset

Split the data into training and test sets

Define the number of folds for cross validation

Define the number of features to select



Initialize a KFold instance

Initialize lists to store the cross validation scores

Iterate over the folds

Split the data into training and validation sets

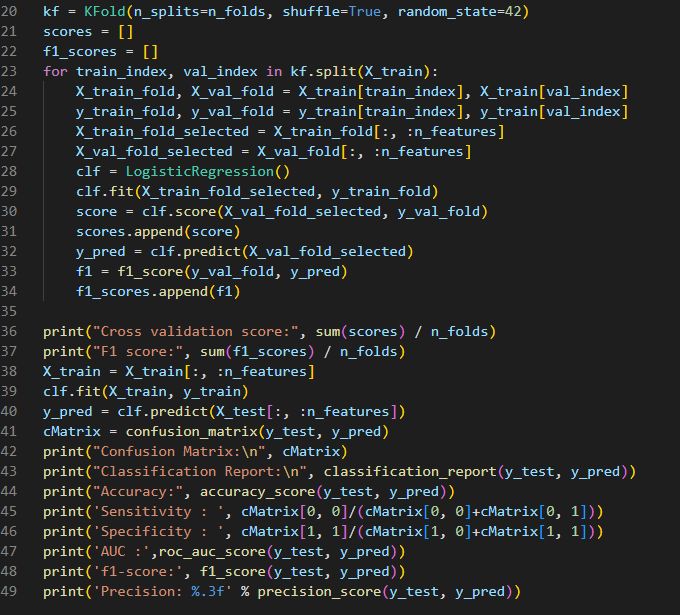
Select the top n\_features by manually selecting the columns from X\_train\_fold

Train a logistic regression model on the selected features

Evaluate the model on the validation set

Select the top n\_features by manually selecting the columns from X\_train

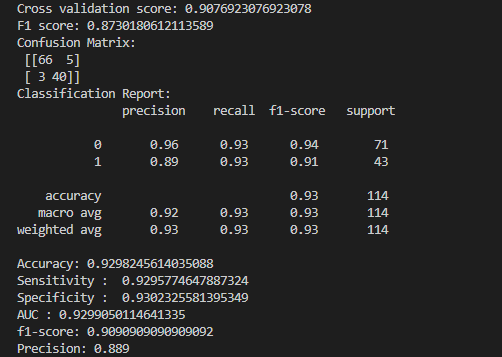
Make predictions on the test set



Cross validation score: 0.9076923076923078

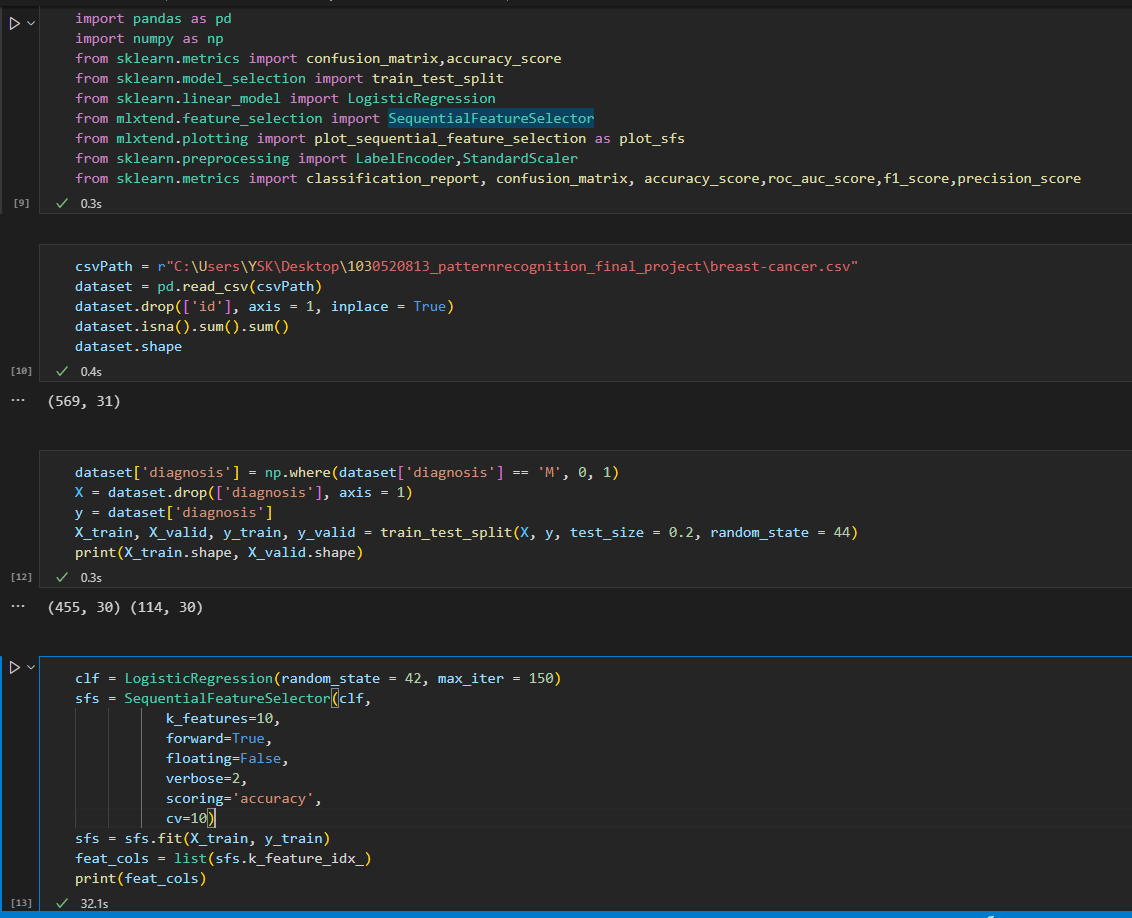
F1 score: 0.8730180612113589

Classification Report:



**Logistic Regression With Wrapped Methods**

Sequential forward selection (SFS), in which features are sequentially added to an empty candidate set until the addition of further features does not decrease the criterion.



Build step forward feature selection

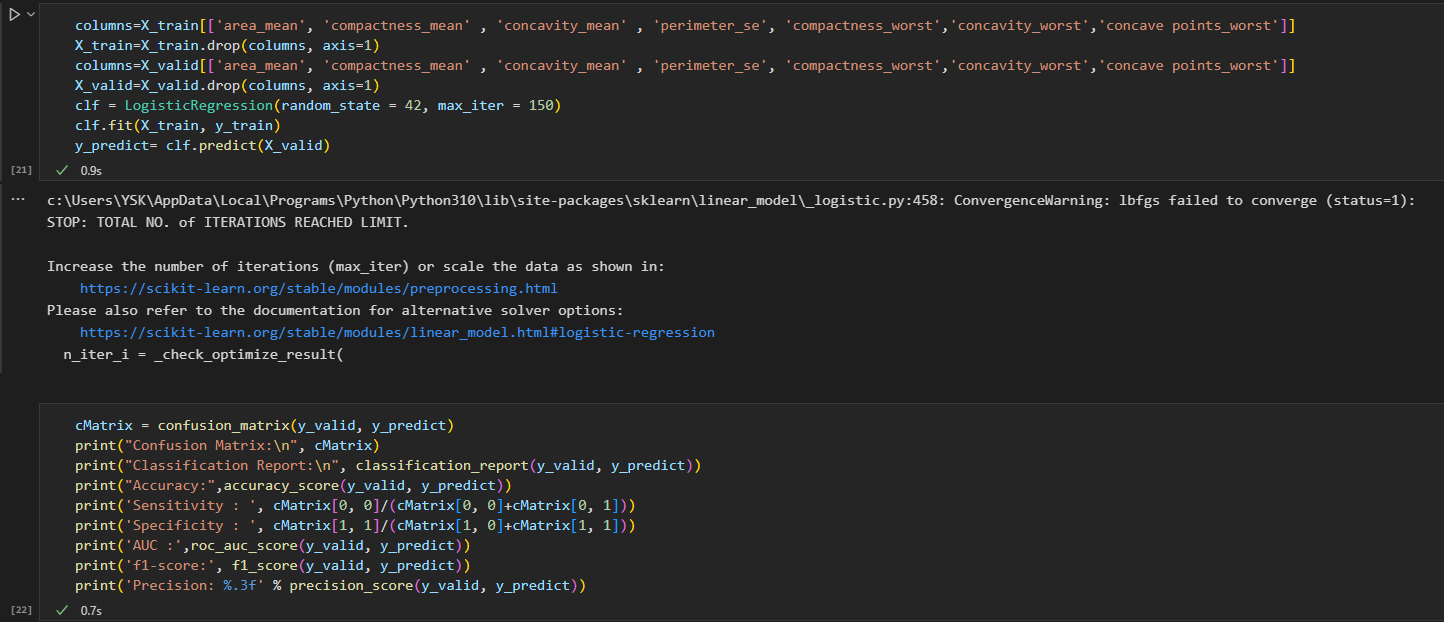
Perform SFS

Calculating Prediction for test data

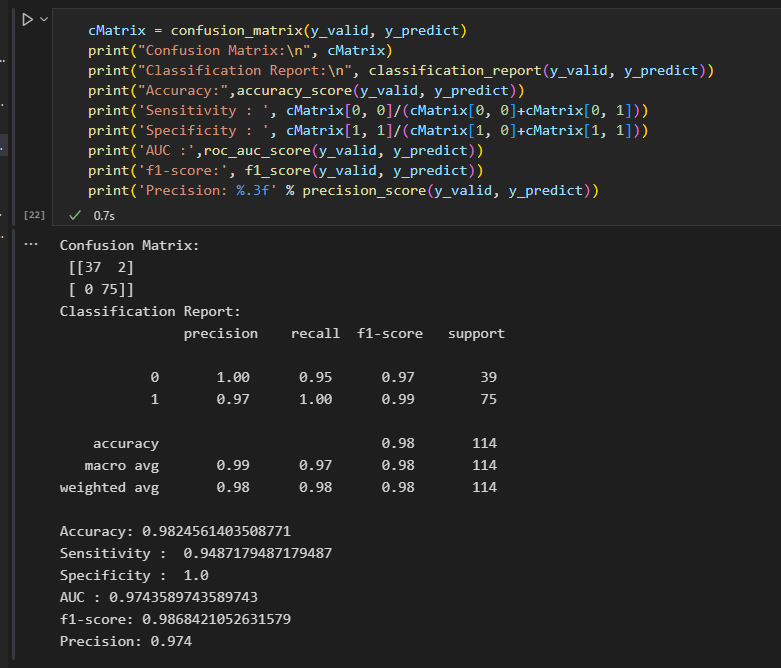
[0, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 14, 19, 21, 22, 24, 26, 27, 29] columns selected.

[Parallel(n\_jobs=1)]: Done 11 out of 11 | elapsed: 2.6s finished

[2023-01-20 12:30:34] Features: 20/20 -- score: 0.9582608695652175



Classification Report:



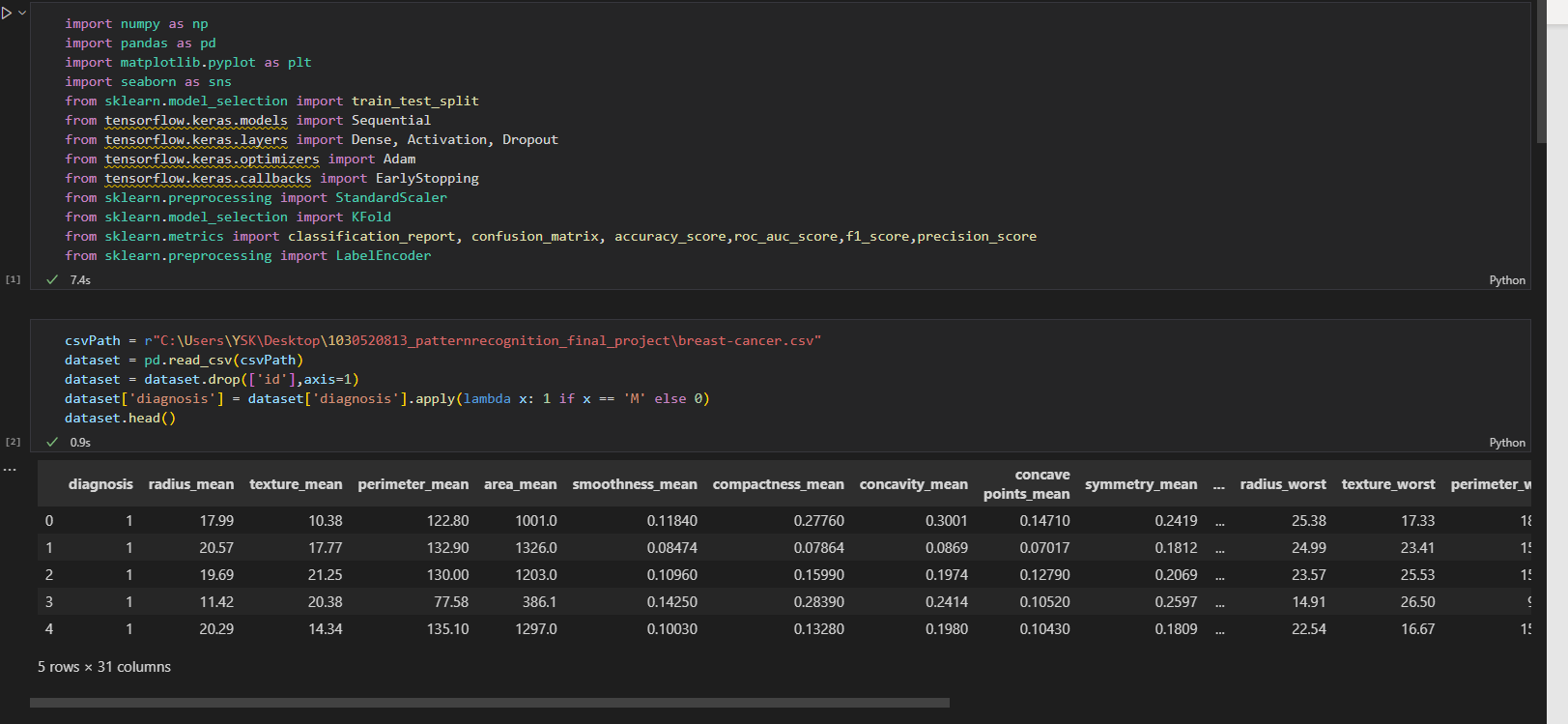
**Artificial Neural Network with Cross Validation**

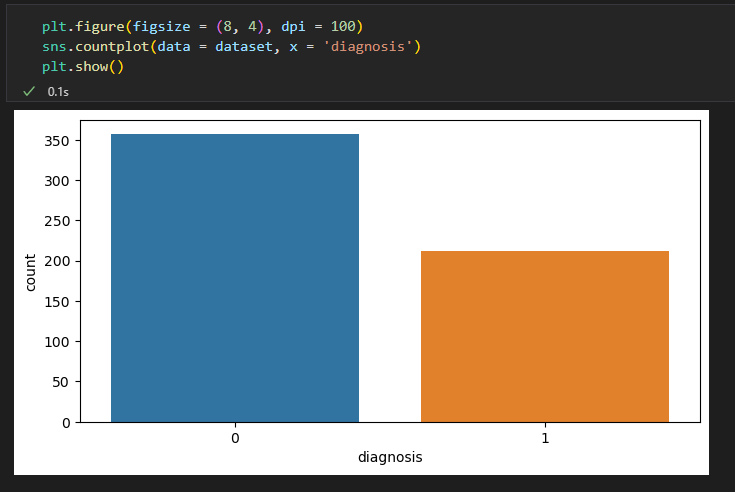
As the name suggests artificial neural network, is the network of artificial neurons. It refers to a biologically inspired modeled after the brain. We can say that it is usually a computational network based on biological neural networks that construct the structure of the human brain.

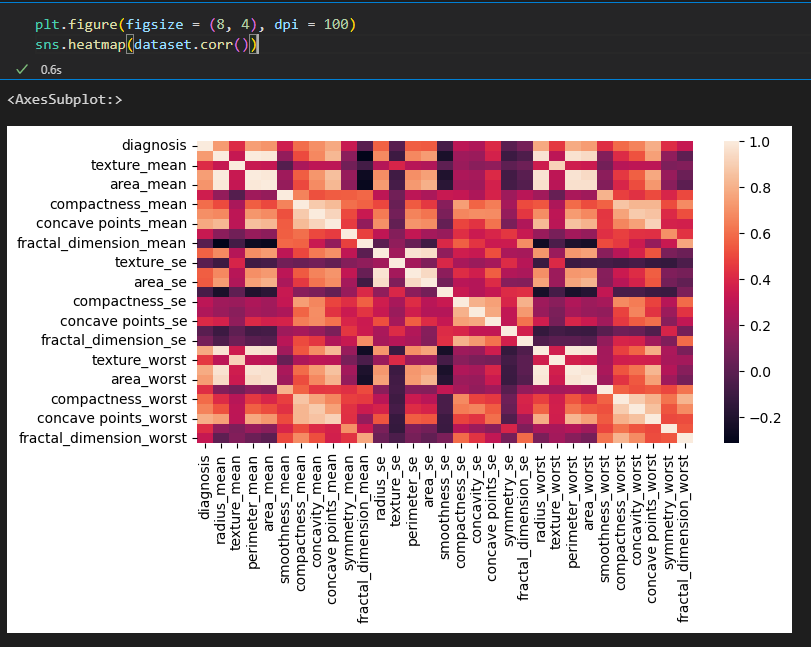
**Steps**

Import Libraries

Visualize Value Counts



****



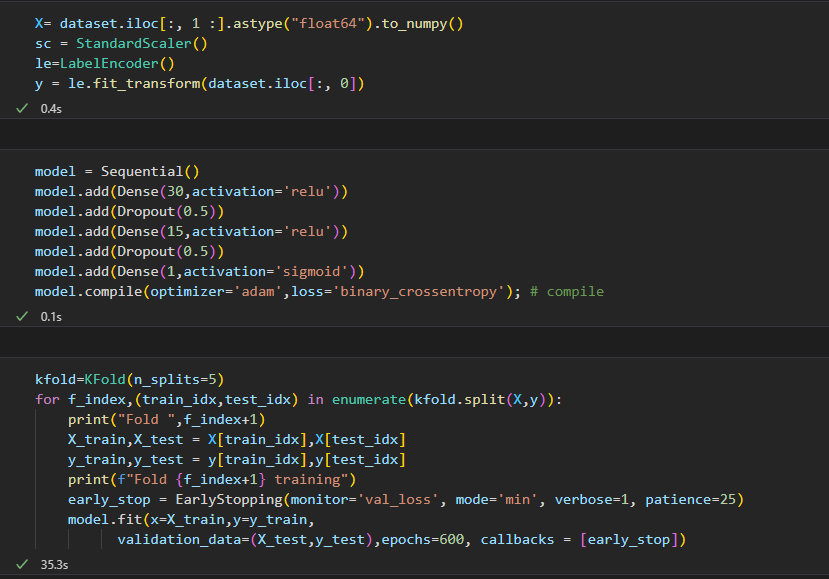
Handling Categorical Value

Splitting Data

Scaling the Data

Creating Layers and Compiling ANN,

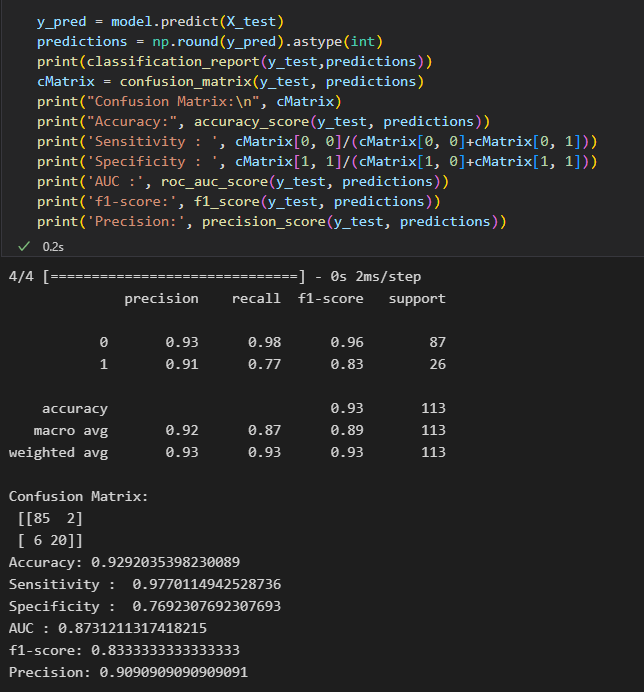
Fitting the ANN in kfold cross validation



Early stopping is designed to monitor the generalization error of one model and stop training when generalization error begins to degrade.

Epoch 60: early stopping

Classification Result:



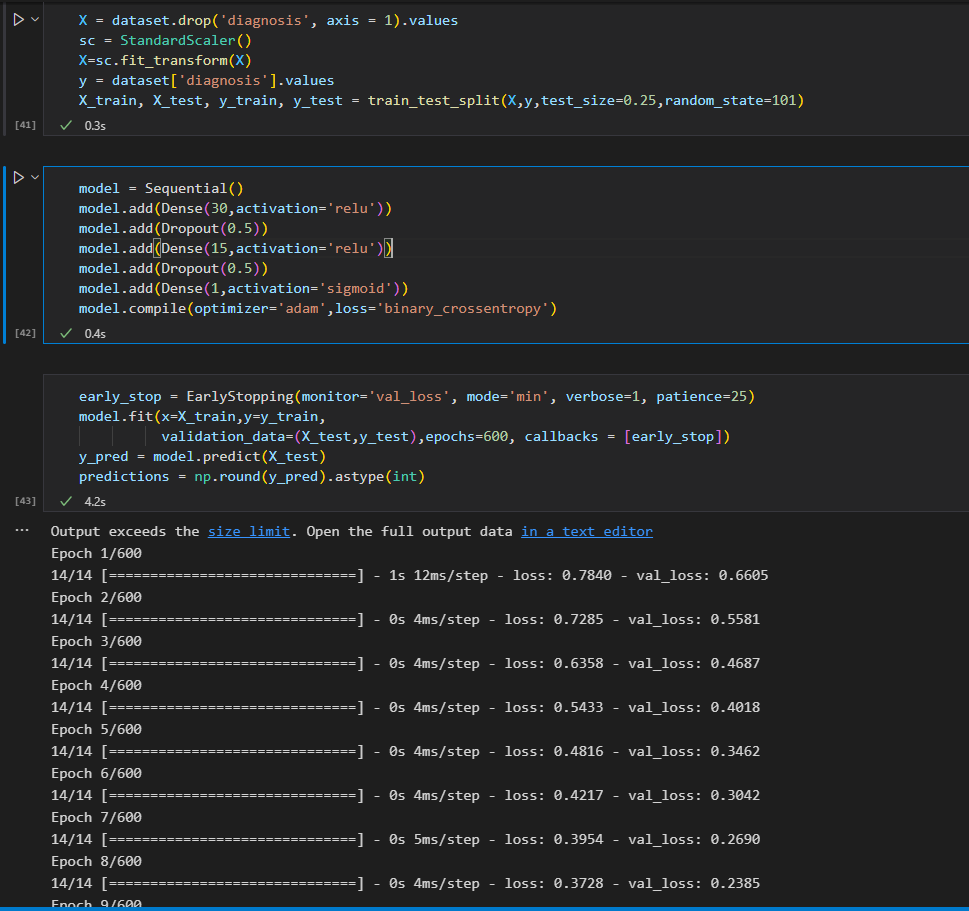
**Artificial Neural Network without Cross Validation**

Made to Stop Early to Avoid Overtraining Neural Networks.Build model architecture. Early stopping requires that you configure your network to be under constrained, meaning that it has more capacity than is required for the problem.When training the network, a larger number of training epochs is used than may normally be required, to give the network plenty of opportunity to fit, then begin to overfit the training dataset.

Compile the model

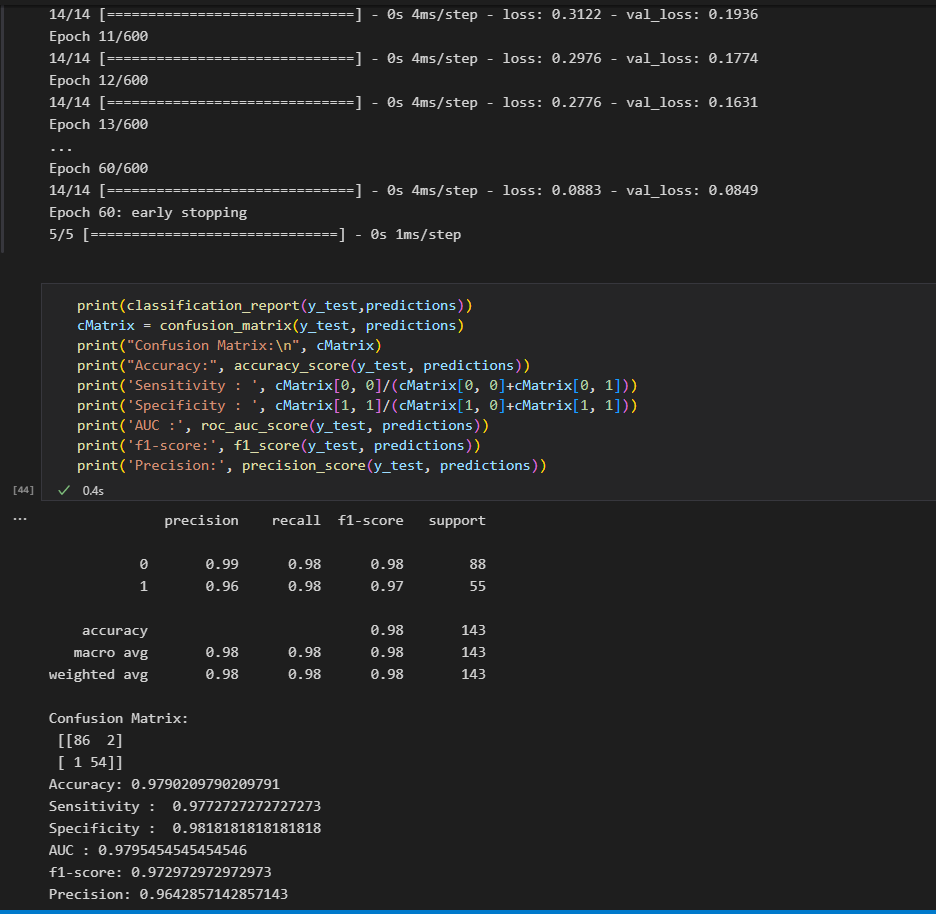
Define Early Stopping

Fit the model

****

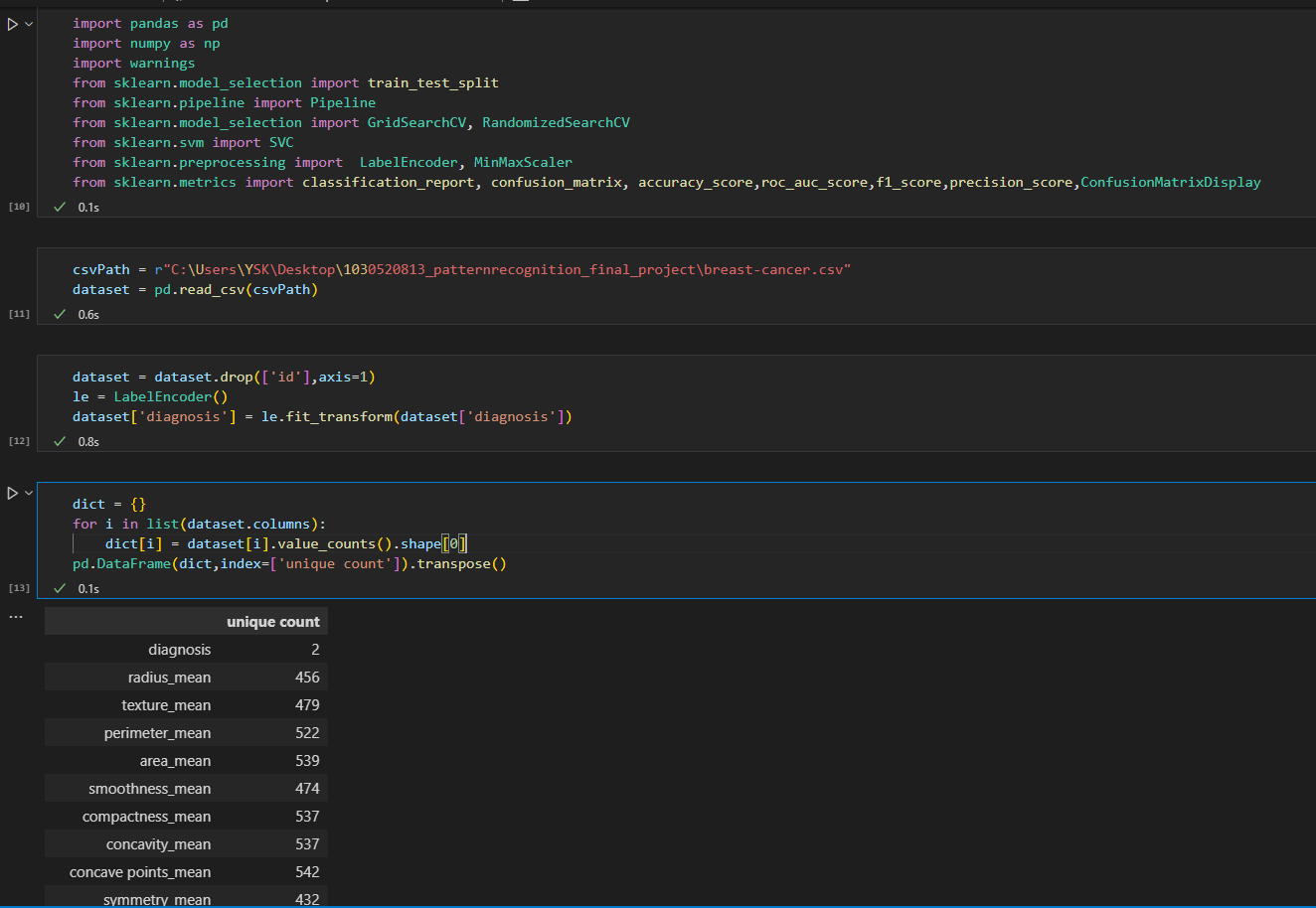
Epoch 60: early stopping

Classification Result:

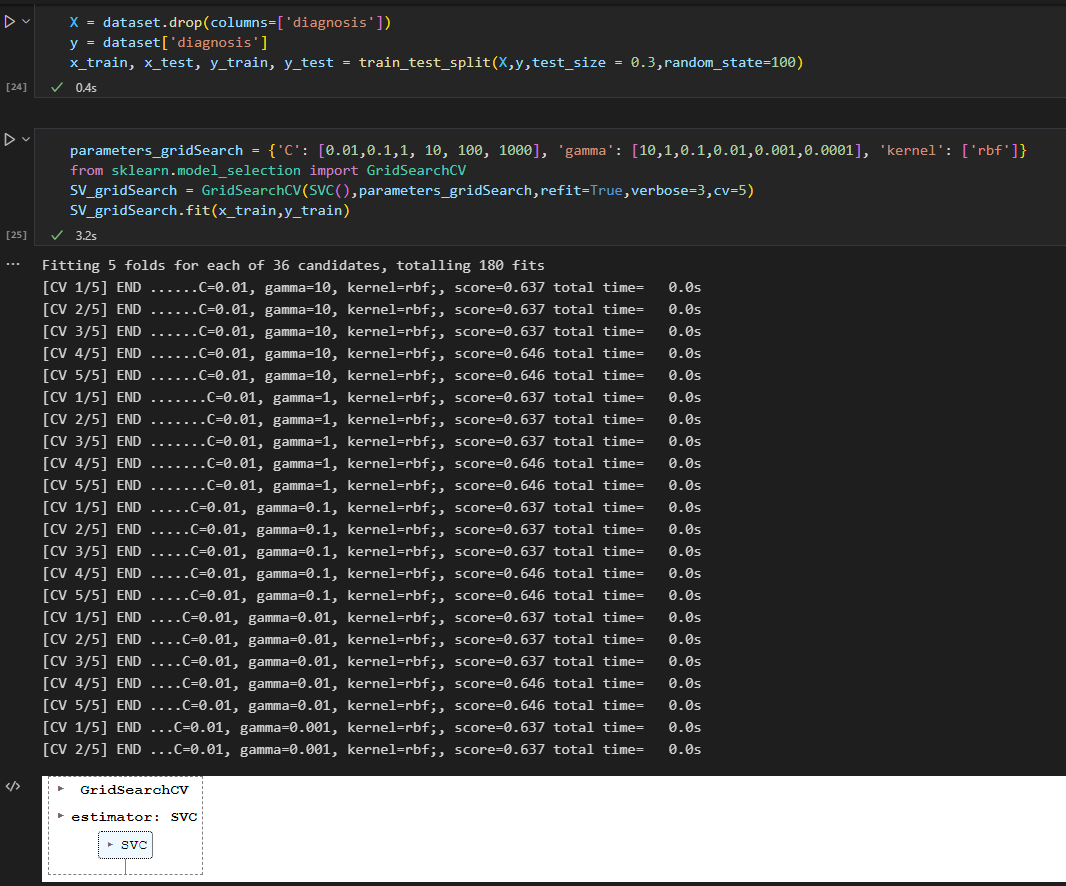
****

**Support Vector Machine**

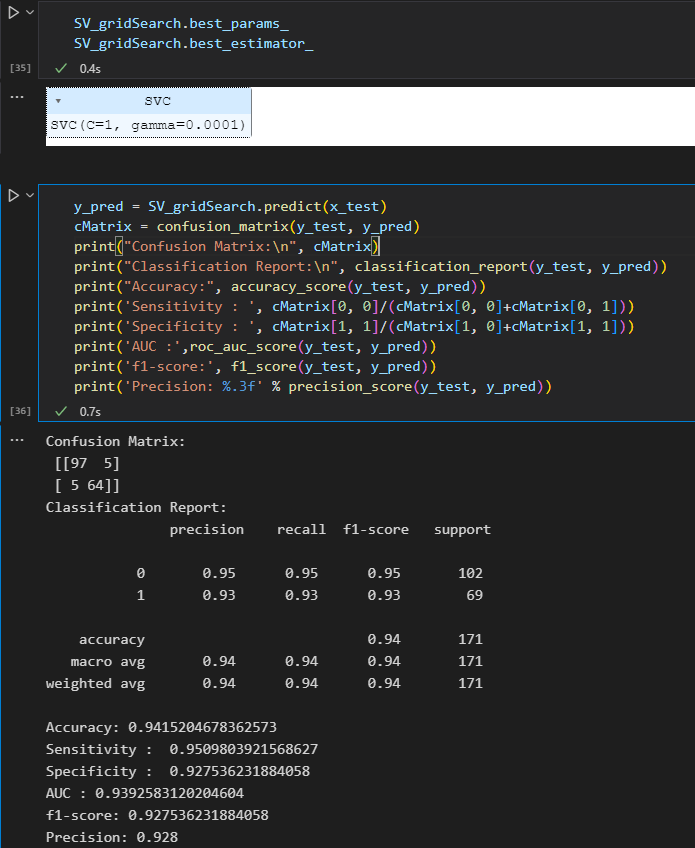
A Support Vector Machine (SVM) is a binary linear classification whose decision boundary is explicitly constructed to minimize generalization error. It is a very powerful and versatile Machine Learning model, capable of performing linear or nonlinear classification, regression and even outlier detection. SVM is well suited for classification of complex but small or medium sized datasets.

****

The ‘GridSearchCV’ estimator taken in the model along with the grid parameters. Then choose verbose. The higher the number, the more verbose. verbose is text output describing the process. Next this grid is fit on the training data. First it finds the best parameter combination by running the same loop with cross-validation. After getting the best combination it runs fit again on all data passed to fit wit cross-validation, to build a single new model using the best parameter setting. The best parameter setting can be obtained by using the ‘best\_params\_’ function and the best estimator can be obtained by using the ‘best\_estimator\_’ setting.



Classification Result (C=1,Gamma=0,0001)

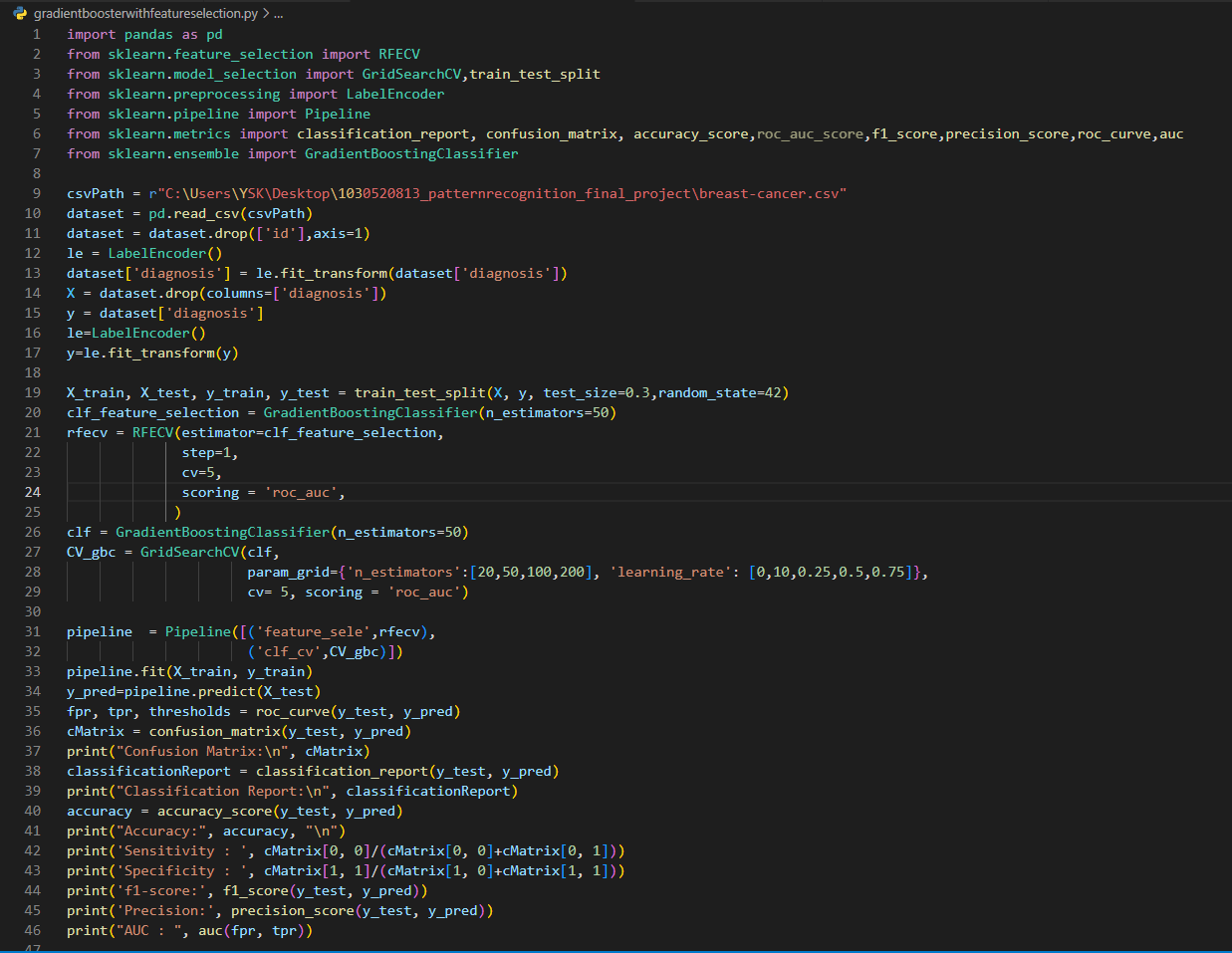
****

**Gradient Booster Algorithm With Feature Selection**

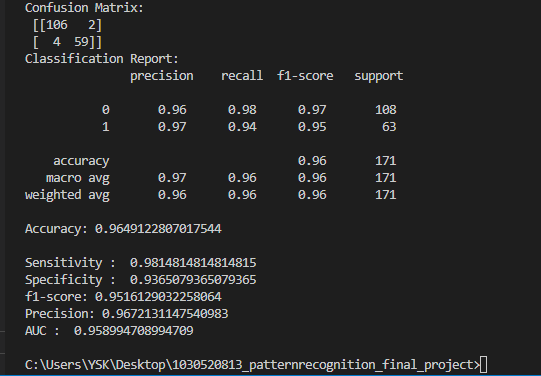
Gradient boosting classifiers are a group of machine learning algorithms that combine many weak learning models together to create a strong predictive model. Gradient boosting models are becoming popular because of their effectiveness at classifying complex datasets.

The purpose of the pipeline is to assemble several steps that can be cross-validated and selected features together while setting different parameters.

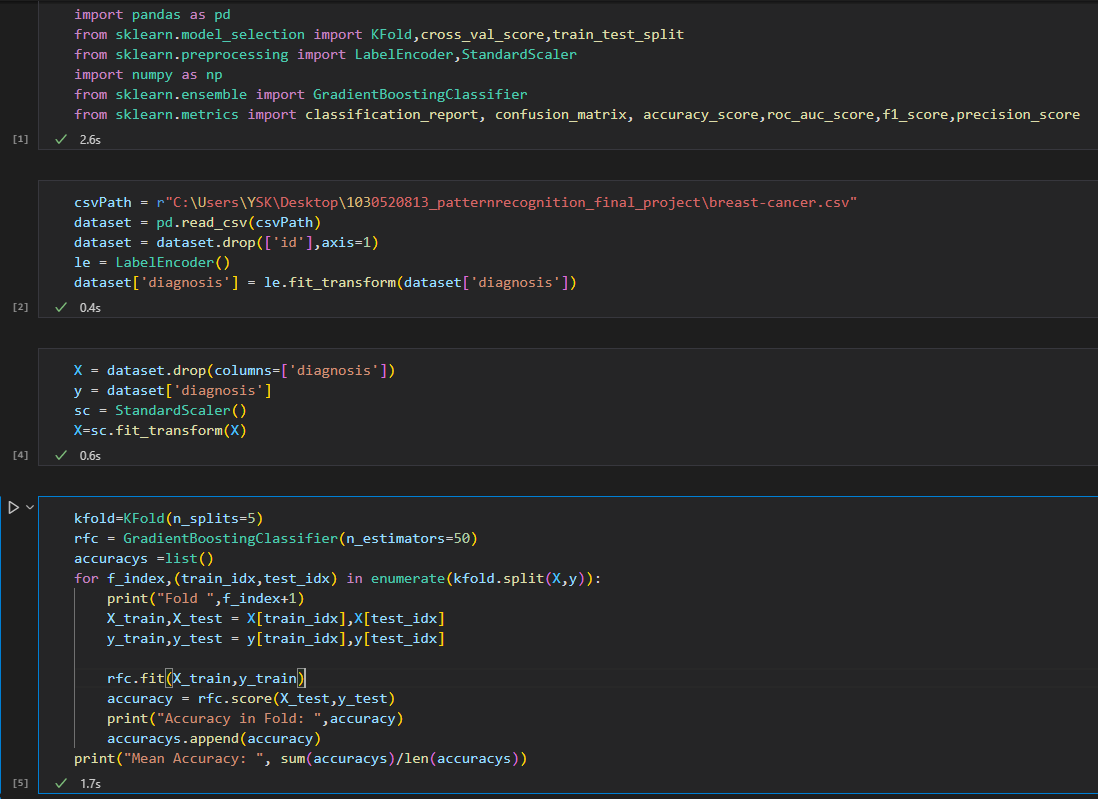
I want to fine tune the hyper parameter of our classifier (with Cross validation) after feature selection using recursive feature elimination (with Cross validation).



Hyperparemetes are key parts of learning algorithms which effect the performance and accuracy of a model. Learning rate and n\_estimators are two critical hyperparameters for gradient boosting. Learning rate, denoted as α, simply means how fast the model learns. Each tree added modifies the overall model. The magnitude of the modification is controlled by learning rate. The lower the learning rate, the slower the model learns.



**Gradient Booster Algorithm With My Feature Selection**

****

Fold 1 Accuracy in Fold: 0.9122807017543859

Fold 2 Accuracy in Fold: 0.9298245614035088

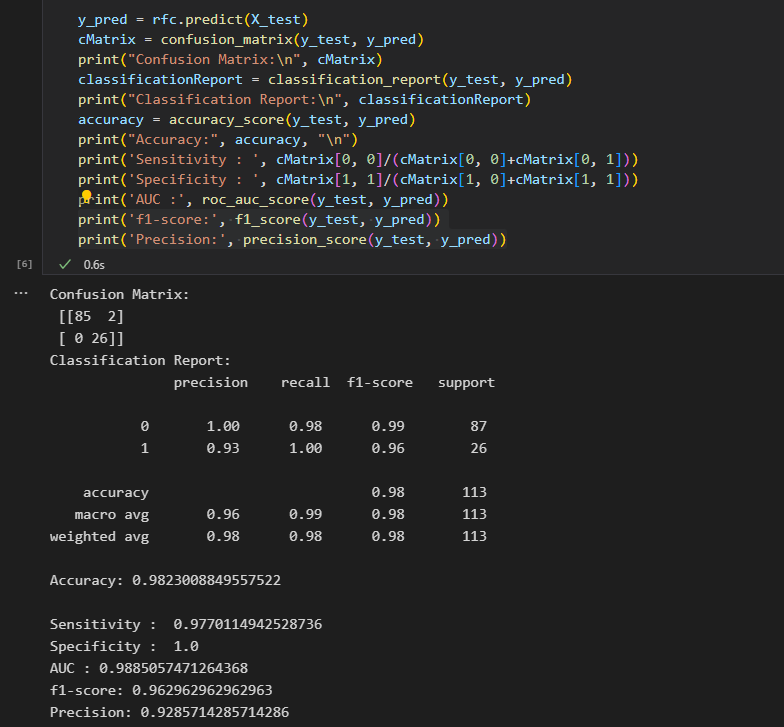
Fold 3 Accuracy in Fold: 0.9912280701754386

Fold 4 Accuracy in Fold: 0.956140350877193

Fold 5 Accuracy in Fold: 0.9823008849557522

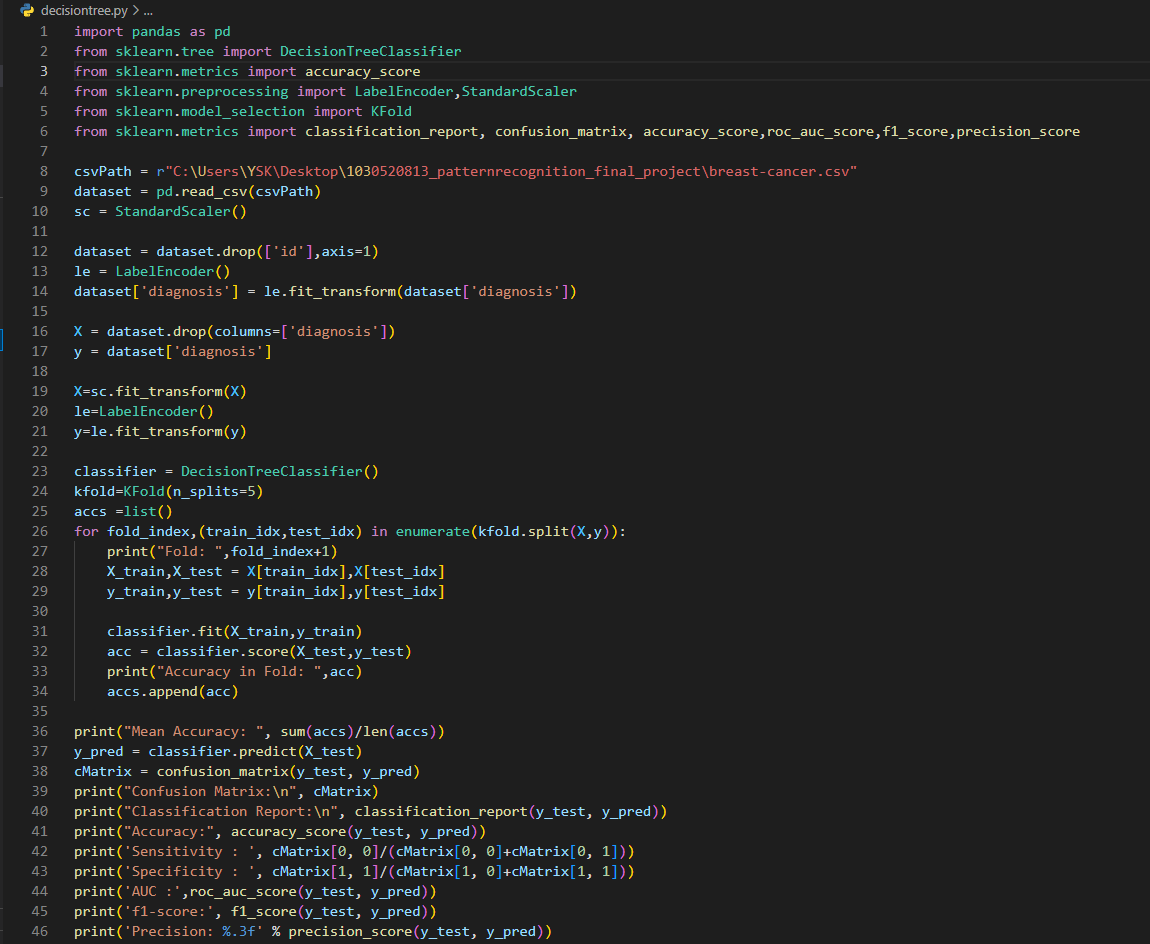
Mean Accuracy: 0.9543549138332557

Classification Result:



**Decision Tree**

Decision Tree is a Supervised learning technique that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. It is a tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome.



Fold Result:

Fold: 1 Accuracy in Fold: 0.8771929824561403

Fold: 2 Accuracy in Fold: 0.9122807017543859

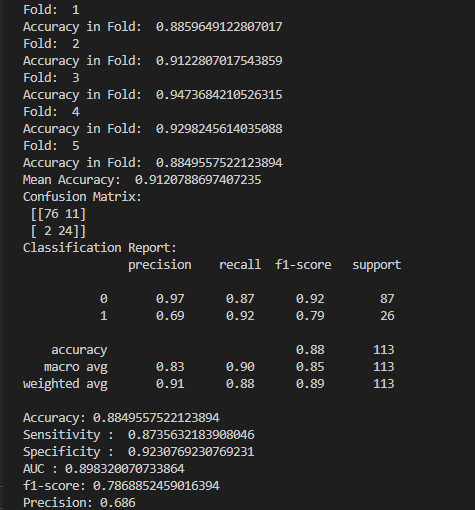
Fold: 3 Accuracy in Fold: 0.9473684210526315

Fold: 4 Accuracy in Fold: 0.9122807017543859

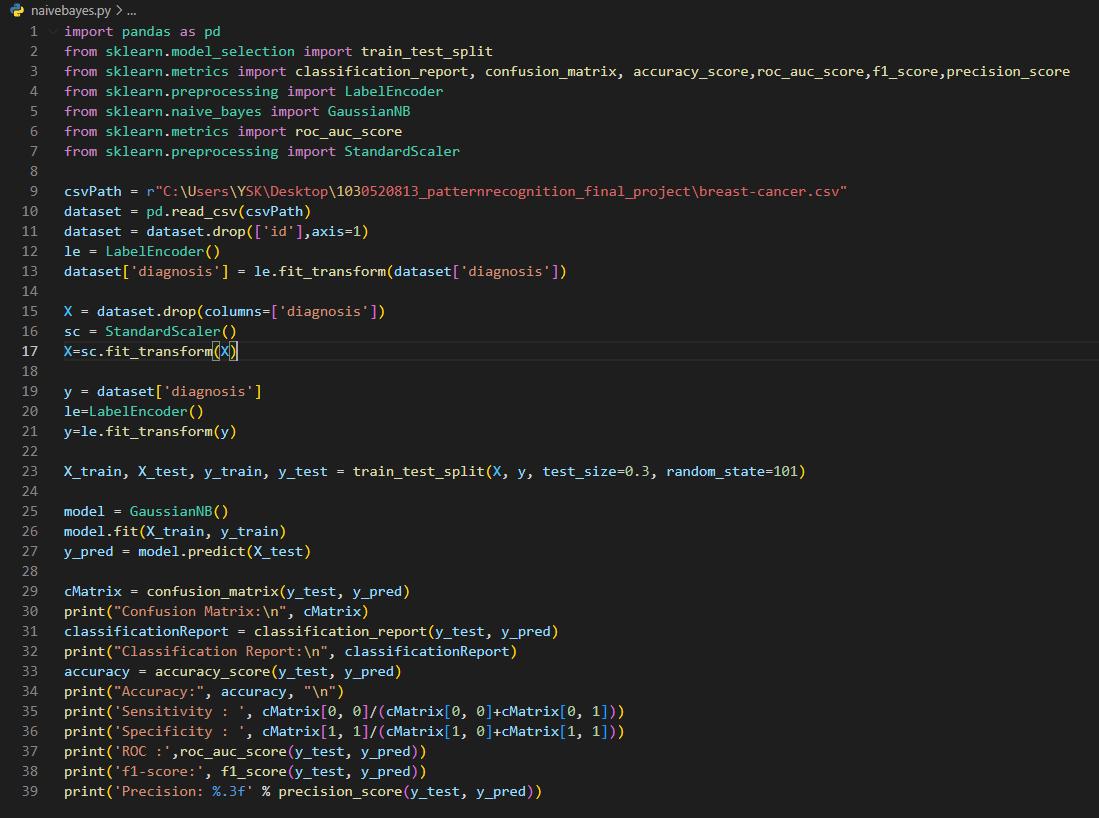
Fold: 5 Accuracy in Fold: 0.8053097345132744

Mean Accuracy: 0.8908865083061637

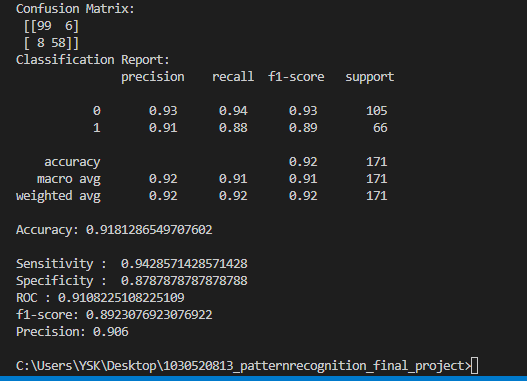
Classification Result:



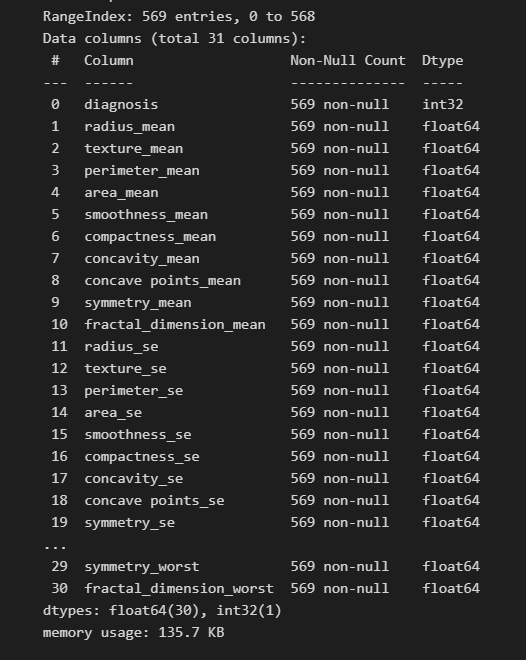
**Naive Bayes Classifier**

****

Classification Result:

****

**dataset.info()**

****

**With All the Results**

The random forest algorithm and artificial neural networks gave the best result.