Dataset: Breast Cancer Wisconsin (Diagnostic) Data Set
Software Tool: Google Colaboratory
Programming Language: Python

BREAST CANCER PREDICTION

Machine Learning & Data Analysis
Final Project 2021/2022

Patrone Paolo (S4643377) **Raffo Matteo (S4620552)**

Dataset Introduction

We have chosen it because in the medical world prevention is never too much and through research it is possible to prevent and treat many diseases that could otherwise be fatal

We decided to use this dataset to predict whether a patient has benign or malignant breast cancer based on properties of the cancer

Attribute Information:

1. ID number
2. Diagnosis (M = malignant, B = benign)

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

3. Radius (mean of distances from center to points on the perimeter)
4. Texture (standard deviation of gray-scale values)
5. Perimeter
6. Area
7. Smoothness (local variation in radius lengths)
8. Compactness
9. Concavity (severity of concave portions of the contour)
10. Concave points (number of concave portions of the contour)
11. Symmetry
12. Fractal dimension

Algorithms

01

Decision Tree

02

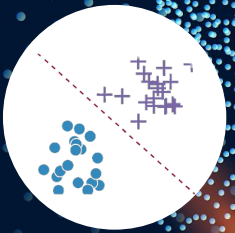
Random Forest

Main purpose: To predict whether a person has benign cancer or malignant cancer we use and compare these 2 models / algorithms

Steps and goals

- Dataset Reading and Correction
- Observations
- Data Preparation
- Prediction without tuned parameters
- Prediction with tuned parameters
- Conclusions

About Classification



- The classification algorithm is a supervised learning technique that is used to identify the category of new observations based on the training data.
- The goal of the classification algorithm is to identify the category of a given data set

There are two types of classifications:

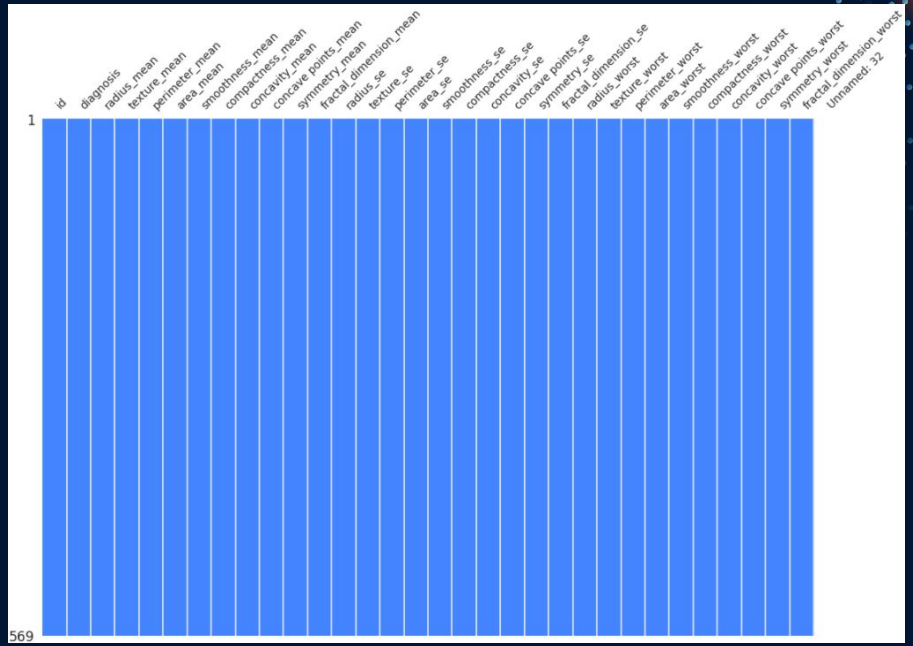
- **Binary Classifier**: If the classification problem has only two possible outcomes
- **Multiclass Classifier**: If a classification problem has more than two results

- In our project we have a binary classification, because we have 2 classes: Benign cancer and Malignant cancer
- A classifier utilizes some training data to understand how given input variables relate to the class

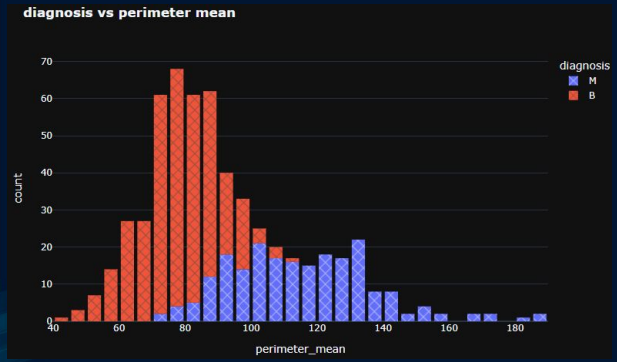
Dataset Reading and Correction

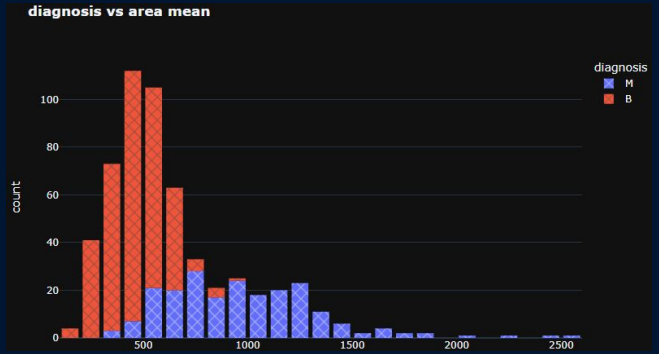
In this phase:

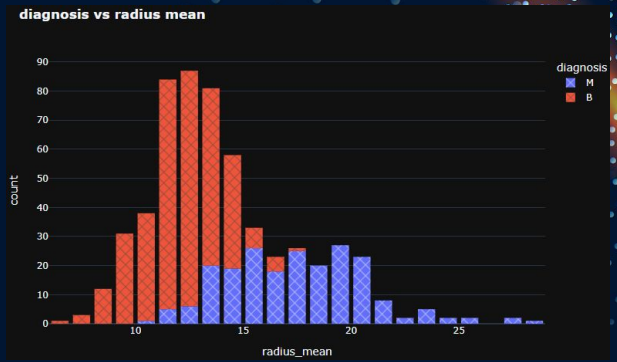
- We acquire the dataset
- We realize that all the values of the "Unnamed: 32" column are null
- Since the entire column contains null values, we have decided to delete it from the dataset
- Finally we check and see that there are no more null values



Observations







The higher the perimeter mean of the tumor, the more likely it is to be a malignant tumor

Up to 700 area mean value cancer tumors of type B are more, instead from 700 area mean value cancer tumors of type M are more

The higher the radius mean of the tumor, the more likely it is to be a malignant tumor

N.B.: For other observations using other graphs, look at the .ipynb file created with Google Colaboratory

Smote

Before moving on to creating the ML models we must first improve the dataset so that our models run at their best. We do this by converting all values to numbers and applying the SMOTE algorithm

When use SMOTE?
Our dataset is slightly unbalanced, the number of elements with Diagnosis cancer at 1 has fewer elements than Diagnosis at 0 and this could affect accuracy to ours ML models

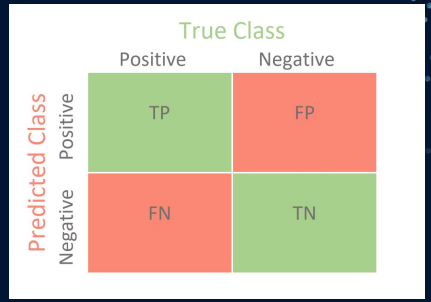
It is necessary to use it?
Since the random forest model is built on decision trees and decision trees are sensitive to class imbalance, it is essential to use a balanced dataset

ROC and Confusion Matrix

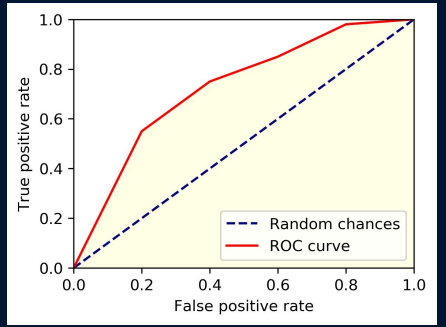
First we will train the model on the train data and then we will run it on the test data

Precision: The ratio between correct elements and total elements will be expressed as a percentage, this will tell us the accuracy of the prediction

Confusion Matrix: This matrix is composed of 2 rows and 2 columns and indicates the predicted values corresponding to the real values. In this way we could evaluate both which targets were predicted correctly and which ones were wrong



ROC: This concept is plotted with the ROC curve (Receiver operating characteristic). True Positive Rate (TPR, fraction of true positives) on Y axes and False Positive Rate (FPR, fraction of false positives) on X axes. The yellow area is called AUC and tells us if we have many correct values over the wrong ones



Prediction without tuned parameters

In this phase:

- We train our models on our dataset
- We use confusion matrix to evaluate the accuracy of the classification to estimate the skill of machine learning models
- Finally we are able to compare the models at the end of the training

Prediction with tuned parameters

In this phase:

- With GridSearchCV(...) we find the best parameters and the best accuracy for each model
- We use them to train and test all models
- Also in this case to evaluate and estimate the skill of machine learning models we use metrics
- Finally, thanks to these metrics, we are able to compare the models at the end of testing

Decision Tree Tuned Parameters

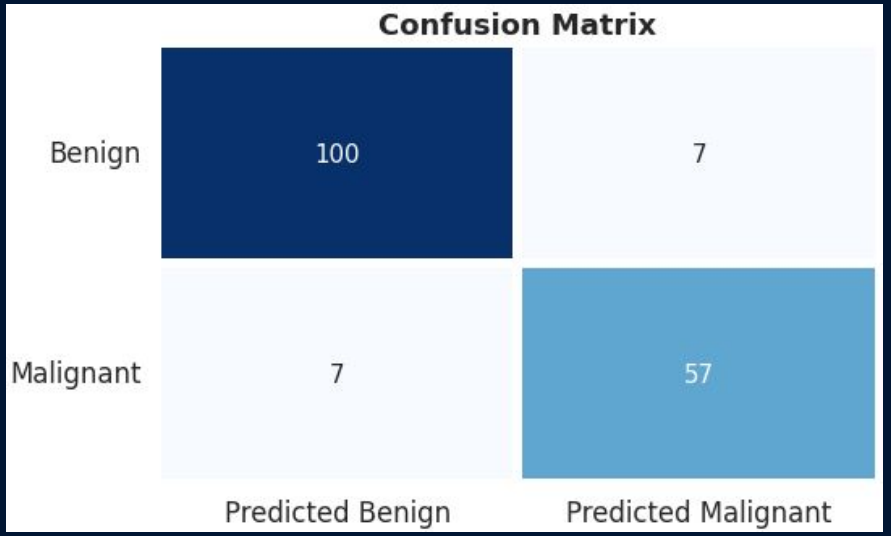
The parameter tuning needs in order to find the best parameters.

The parameters we are going to use are:

- **Max Depth**: The max depth of the tree
- **Max_features**: The number of features to consider when looking for the best split
- **Min_samples_leaf**: The min number of samples required to be at a leaf node
- **Min_sample_split**: The min number of samples required to split an internal node

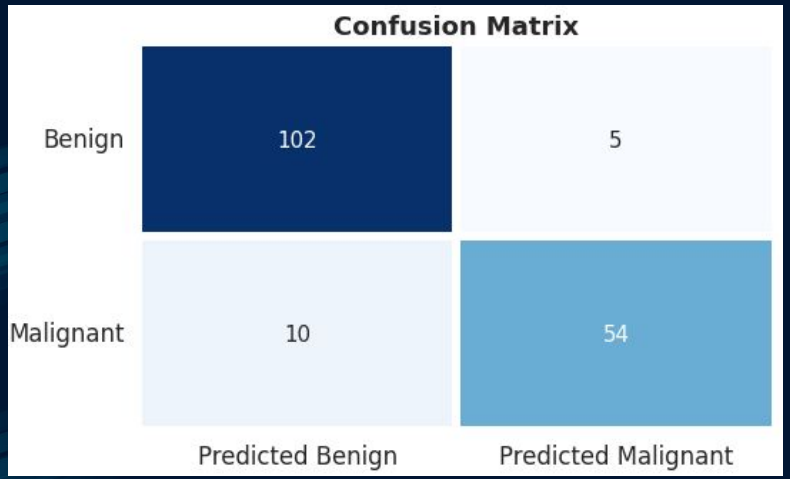
Decision Tree Tuned Parameters

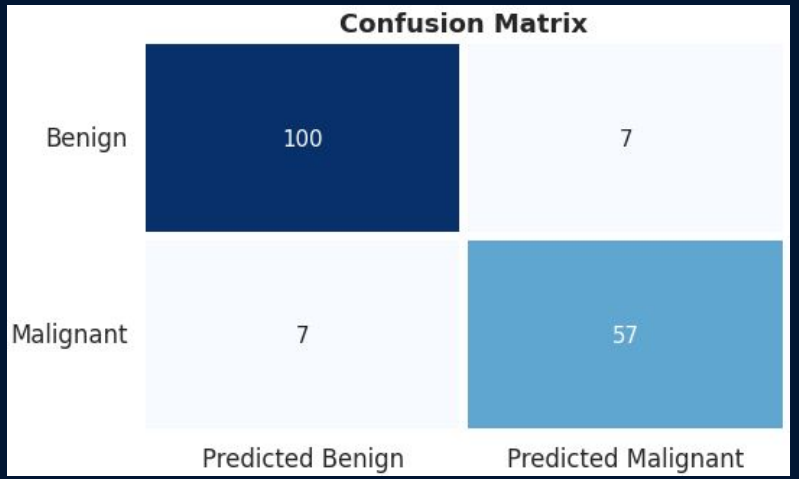
```
DecisionTreeClassifier():
Best Parameters: {'max_depth': 4, 'max_features': 8, 'min_samples_leaf': 1, 'min_samples_split': 4}
```



Decision Tree with vs without Tuning

	Before tuning	After tuning
Accuracy :	~91%	~92%





Random Forest Tuned Parameters

The parameter tuning needs in order to find the best parameters.

Hyperparameters

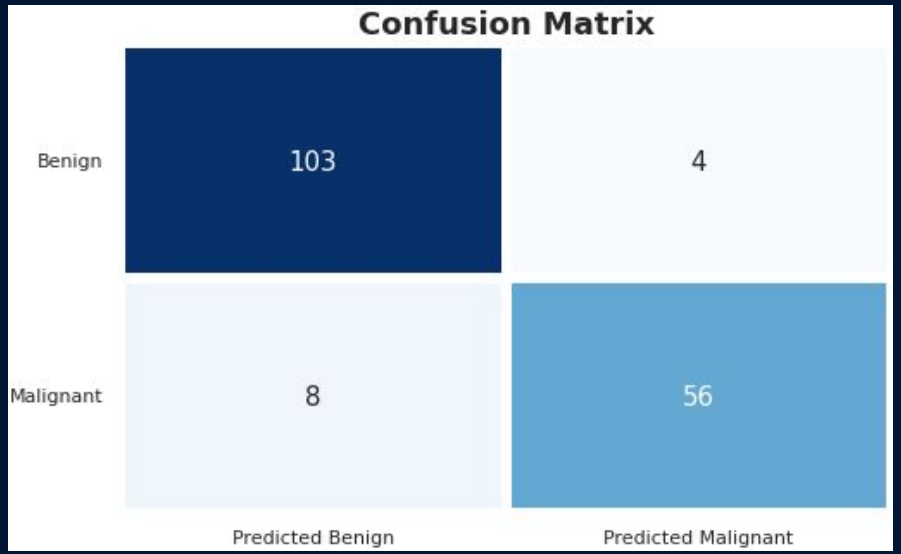
- **Bootstrap values:** Number of values extracted from the bootstrap → default: 2/3
- **Max features:** The max number of features I look at each node → default: sqrt(n_features)
- **Max depth:** depth of trees / number of leaves → default: None

Other important parameters

- **N_estimators:** Number of trees → default: 100
- **Algorithm for voting** → default: Majority Voting
- **Min sample leaf:** The min number of samples required to be at a leaf node → default: 1
- **Min sample split:** The min number of samples required to split an internal node → default: 2

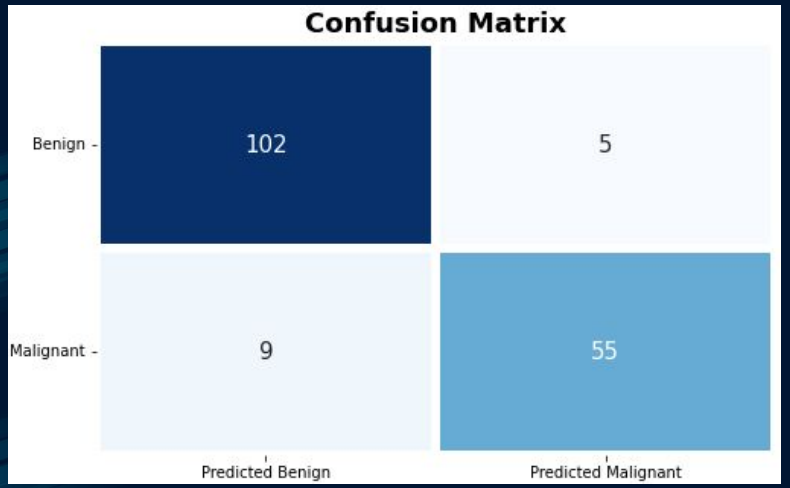
Random Forest Tuned Parameters

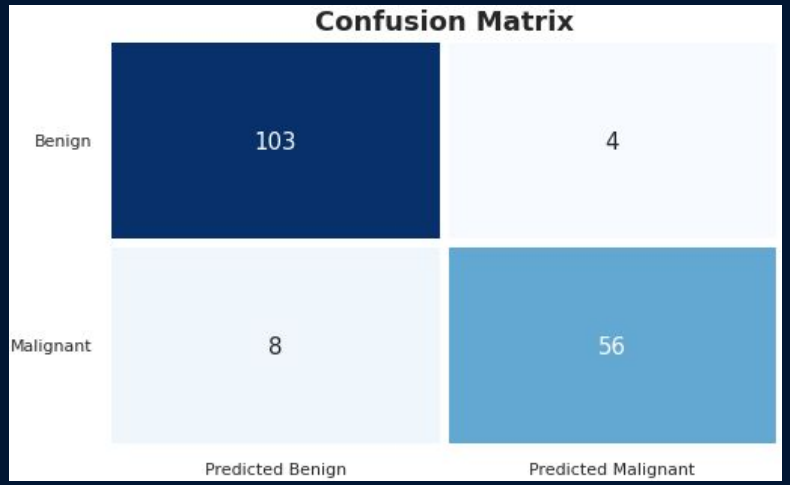
```
 RandomForestClassifier():
Best Parameters: {'max_depth': 7, 'max_features': 1, 'min_samples_leaf': 1, 'min_samples_split': 4, 'n_estimators': 100 }
```



Random Forest with vs without Tuning

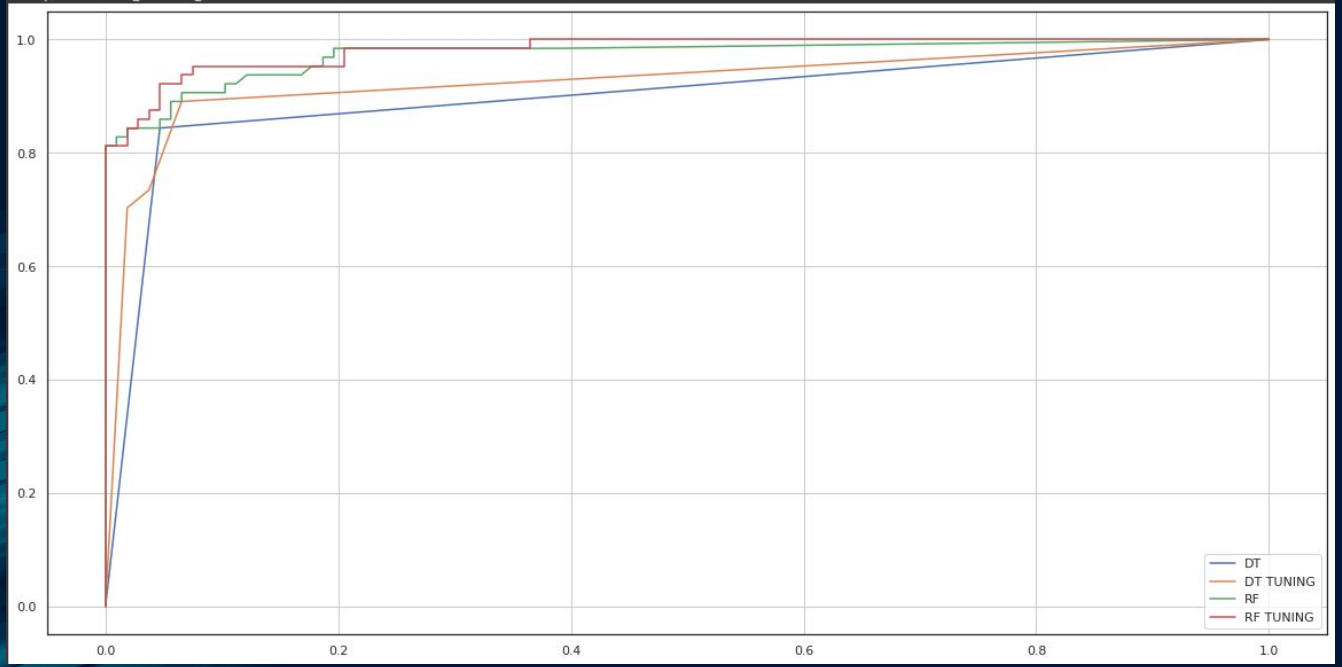
	Before tuning	After tuning
Accuracy :	~92%	~93%





Conclusions

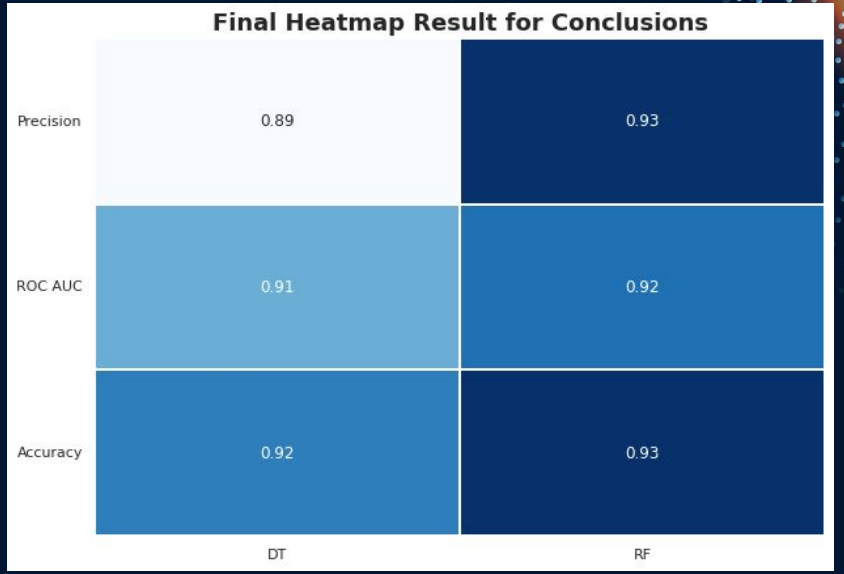
```
DT 0.8985105140186915
DT TUNING 0.9126022196261683
RF 0.9063230140186916
RF TUNING 0.9188084112149533
<matplotlib.legend.Legend at 0x7f4050374650>
```



	Accuracy
DT	~91%
DT Tuning	~92%
RF	~92%
RF Tuning	~93%

Conclusions

- These two algorithms can predict who has had benign cancer or malignant cancer quite well
- The best classifier for not labeling a benign sample that is not benign is Random Forest (highest accuracy)
- The best classifier not to label as positive a sample that is negative (as we can see from the precision value) is Random Forest



THANKS FOR THE ATTENTION!

References:

- Our .ipynb document created with Google Colaboratory
- Machine Learning Classifiers - towardsdatascience.com
- Introduction to Random Forest in Machine Learning - section.io
- Decision Trees for Classification: A Machine Learning Algorithm - xoriant.com
- SMOTE for Imbalanced Classification with Python

Patrone Paolo (S4643377) - Raffo Matteo (S4620552)