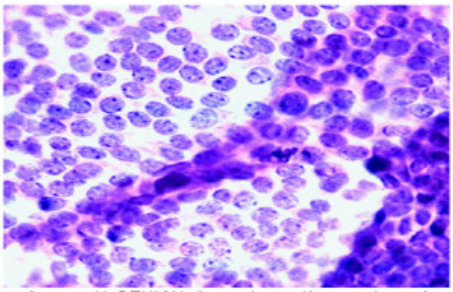
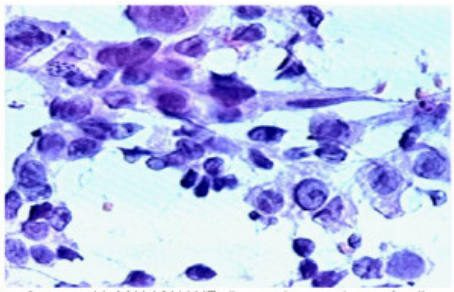
BREAST CANCER ANALYSIS

Breast cancer is the most common cause of cancer deaths in women. However, it 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. For this purpose we worked on the *Wisconsin Diagnostic Breast Cancer (WDBC)* dataset, implementing some classification algorithms: Logistic Regression, Support Vector Machine (SVM), Naive Bayes, Decision Tree, Random Forest, Extra Tree and k-Nearest Neighborhood.

The dataset is from the *University Hospital of California* and can be downloaded both from UCI Machine Learning Repository and Kaggle. It consists of 569 samples and 33 features, computed from a digitized image of a fine needle aspiration (FNA) of a breast mass and related to some characteristics of each cell nucleus (e.g. radius, texture, perimeter, area etc.). Some of these features are more selective and decisive than others and the determination of these features significantly increases the success of the models, reason why Feature Selection is applied to select them. Finally, there are 212 Malignant and 357 Benign out of the 569 breast cancer data in the dataset. The following figures show an example of them:



Smear with BENIGN diagnosis – uniform nucleus of cells, symmetrical, homogeneous, with areas within normal size



Smear with MALIGNANT diagnosis – nucleus of cells without uniformity, asymmetrical, not homogeneous (multiple sizes) and with areas above normal size

1 Loading and Preprocessing Data

[4]:

	id	diagnosis	...	fractal_dimension_worst	Unnamed: 32
0	842302	M	...	0.11890	NaN
1	842517	M	...	0.08902	NaN
2	84300903	M	...	0.08758	NaN
3	84348301	M	...	0.17300	NaN
4	84358402	M	...	0.07678	NaN
..	...	...	...	...	...
564	926424	M	...	0.07115	NaN
565	926682	M	...	0.06637	NaN
566	926954	M	...	0.07820	NaN
567	927241	M	...	0.12400	NaN
568	92751	B	...	0.07039	NaN

[569 rows x 33 columns]

This dataset is composed by 569 observations and 33 columns.

The features are computed from a digitized image of a fine needle aspirate (FNA) of a breast mass and describe characteristics of the cell nuclei present in the image.
We can see them in more details:

1) ID number

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

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)

The mean, standard error and “worst” (namely, ”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 recorded with four significant digits.

1.1 Preprocessing Data

Check for **Missing Values**:

```
[ ]: id                          0
     diagnosis                   0
     radius_mean                 0
     texture_mean                0
     perimeter_mean              0
     area_mean                   0
     smoothness_mean             0
     compactness_mean            0
     concavity_mean              0
     concave points_mean         0
     symmetry_mean               0
     fractal_dimension_mean      0
     radius_se                   0
     texture_se                  0
     perimeter_se                0
     area_se                     0
     smoothness_se               0
     compactness_se              0
     concavity_se                0
     concave points_se           0
     symmetry_se                 0
     fractal_dimension_se        0
     radius_worst                0
     texture_worst               0
     perimeter_worst             0
     area_worst                  0
     smoothness_worst            0
     compactness_worst           0
     concavity_worst             0
     concave points_worst        0
     symmetry_worst              0
     fractal_dimension_worst     0
     Unnamed: 32               569
     dtype: int64
```

As we can see, no variable has missing values, except the last one, *Unnamed: 32*, which, instead, contains only null values, reason why we can remove it from the dataset.

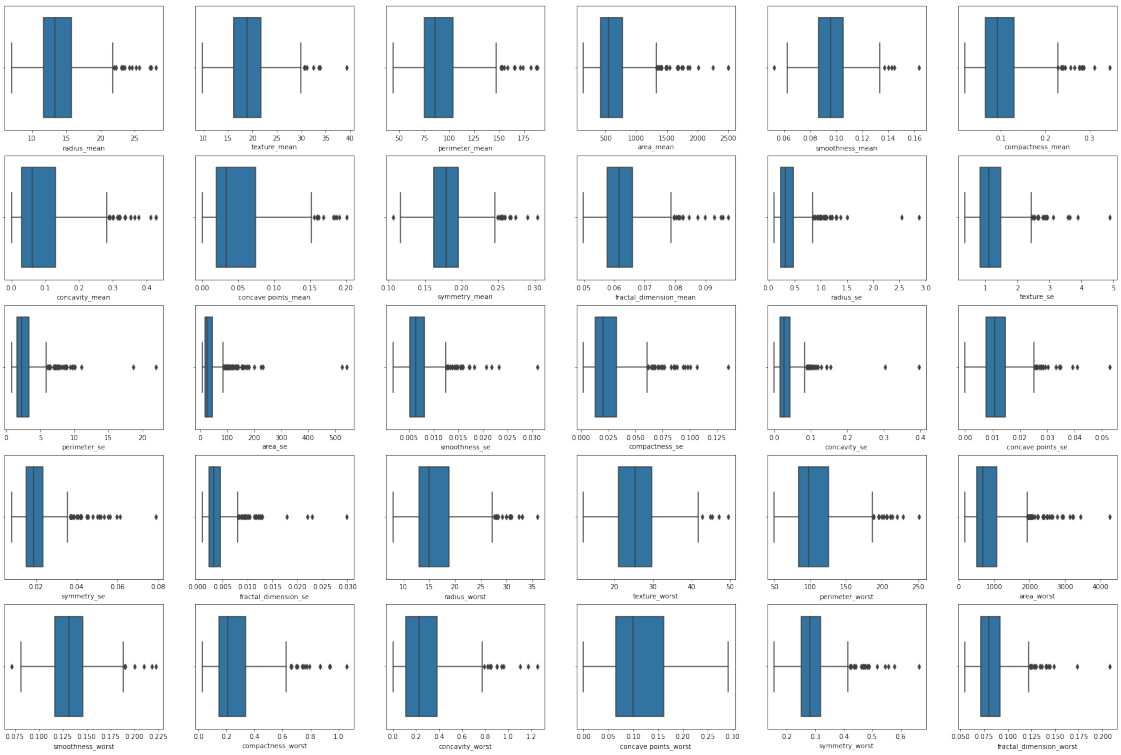
We remove also the *id* variable , because it is not useful for the purposes of our analysis and we end up with this dataset:

[6]:

	diagnosis	radius_mean	...	symmetry_worst	fractal_dimension_worst
0	M	17.99	...	0.4601	0.11890
1	M	20.57	...	0.2750	0.08902
2	M	19.69	...	0.3613	0.08758
3	M	11.42	...	0.6638	0.17300
4	M	20.29	...	0.2364	0.07678
..	...	...	...	...	...
564	M	21.56	...	0.2060	0.07115
565	M	20.13	...	0.2572	0.06637
566	M	16.60	...	0.2218	0.07820
567	M	20.60	...	0.4087	0.12400
568	B	7.76	...	0.2871	0.07039

[569 rows x 31 columns]

Check for **Outliers**:



As we can see from the boxplots, most of the variables have outliers. They may have significant effect on the outcome of tumor being Benign or Malignant and, in order to decide to remove or to keep them, we should have a strong knowledge about features. Therefore, we think it is important to keep them, as they could be significant for classification purposes.

Look at some **statistics**:

[ ]:

	mean	std	...	75%	max
radius_mean	14.127292	3.524049	...	15.780000	28.11000
texture_mean	19.289649	4.301036	...	21.800000	39.28000
perimeter_mean	91.969033	24.298981	...	104.100000	188.50000
area_mean	654.889104	351.914129	...	782.700000	2501.00000
smoothness_mean	0.096360	0.014064	...	0.105300	0.16340
compactness_mean	0.104341	0.052813	...	0.130400	0.34540
concavity_mean	0.088799	0.079720	...	0.130700	0.42680
concave points_mean	0.048919	0.038803	...	0.074000	0.20120
symmetry_mean	0.181162	0.027414	...	0.195700	0.30400
fractal_dimension_mean	0.062798	0.007060	...	0.066120	0.09744
radius_se	0.405172	0.277313	...	0.478900	2.87300
texture_se	1.216853	0.551648	...	1.474000	4.88500
perimeter_se	2.866059	2.021855	...	3.357000	21.98000
area_se	40.337079	45.491006	...	45.190000	542.20000
smoothness_se	0.007041	0.003003	...	0.008146	0.03113
compactness_se	0.025478	0.017908	...	0.032450	0.13540
concavity_se	0.031894	0.030186	...	0.042050	0.39600
concave points_se	0.011796	0.006170	...	0.014710	0.05279
symmetry_se	0.020542	0.008266	...	0.023480	0.07895
fractal_dimension_se	0.003795	0.002646	...	0.004558	0.02984
radius_worst	16.269190	4.833242	...	18.790000	36.04000
texture_worst	25.677223	6.146258	...	29.720000	49.54000
perimeter_worst	107.261213	33.602542	...	125.400000	251.20000
area_worst	880.583128	569.356993	...	1084.000000	4254.00000
smoothness_worst	0.132369	0.022832	...	0.146000	0.22260
compactness_worst	0.254265	0.157336	...	0.339100	1.05800
concavity_worst	0.272188	0.208624	...	0.382900	1.25200
concave points_worst	0.114606	0.065732	...	0.161400	0.29100
symmetry_worst	0.290076	0.061867	...	0.317900	0.66380
fractal_dimension_worst	0.083946	0.018061	...	0.092080	0.20750

We clearly see that **area_mean** and **smoothness_mean** are not on the same scale.
It is well known that machine learning algorithms do not converge well when facing unscaled features, reason why we'll standardize the data.

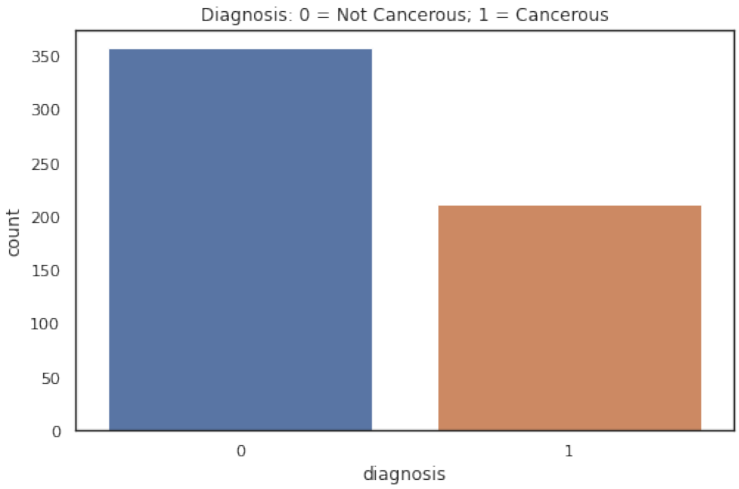
Look at the **target variable** and replace its values:

```
The number of malignant tumors is 212
The number of benign tumors is 357
```

```
[7]: 0      1
     1      1
     2      1
     3      1
     4      1
           ..
     564    1
     565    1
     566    1
     567    1
     568    0
```

Now our target variable assumes values 0 or 1:
- 0 if the tumor is Benign;
- 1 if the tumor is Malignant.

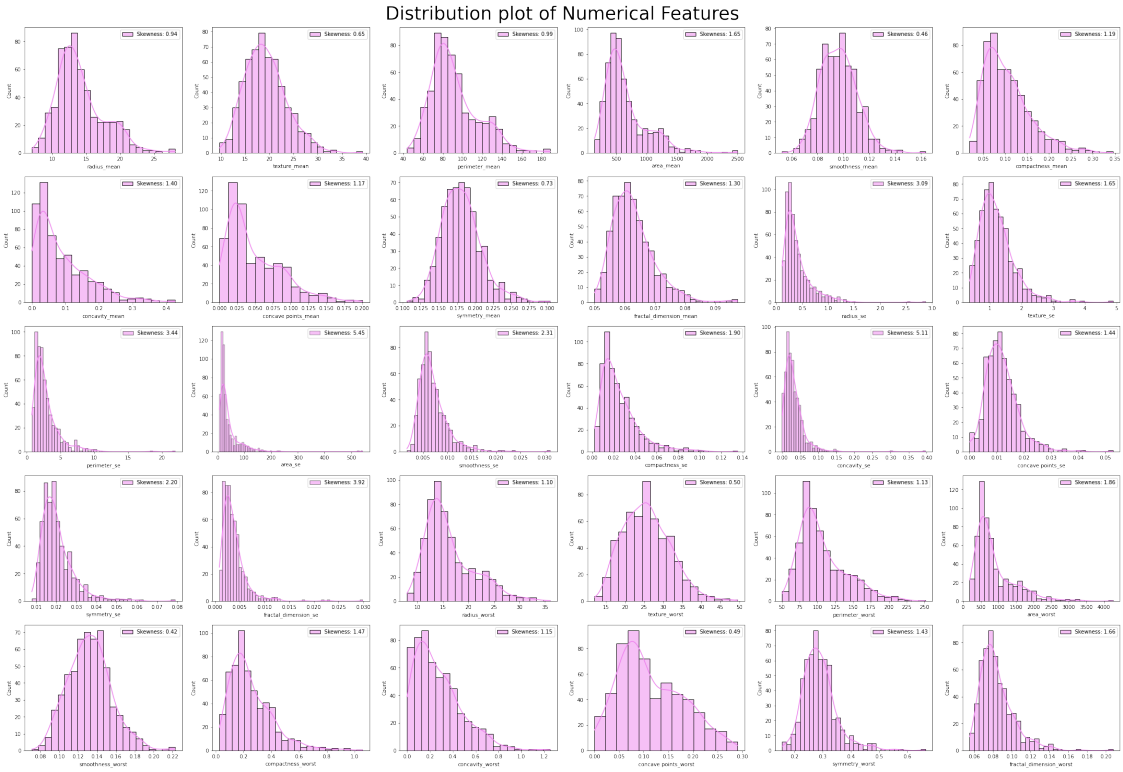
Check for **data imbalance**:



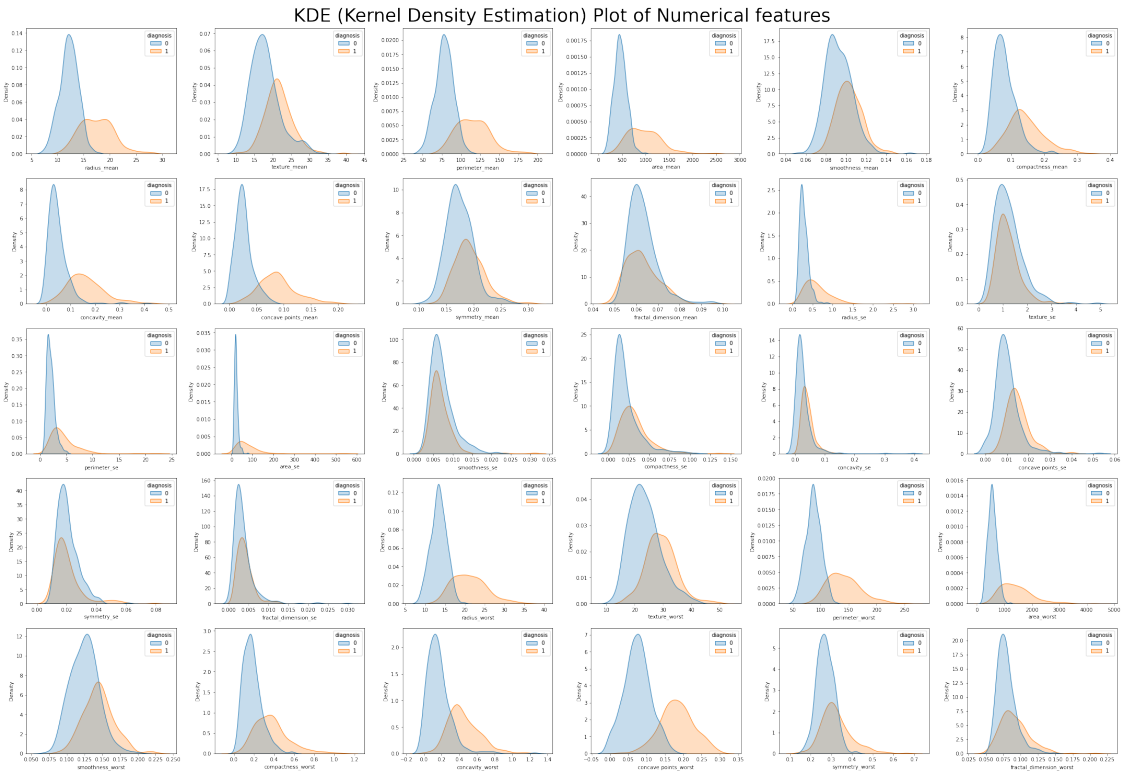
As we can see from the plot, the dataset is unbalanced. This could be a problem in the classification phase, since the probability of instances belonging to the majority class is significantly high, hence the algorithms are much more likely to classify new observations to the majority class. So, we have to handle this matter in the next phases of the analysis.

1.2 Exploratory Data Analysis

Let's have a look at the distribution of our data:



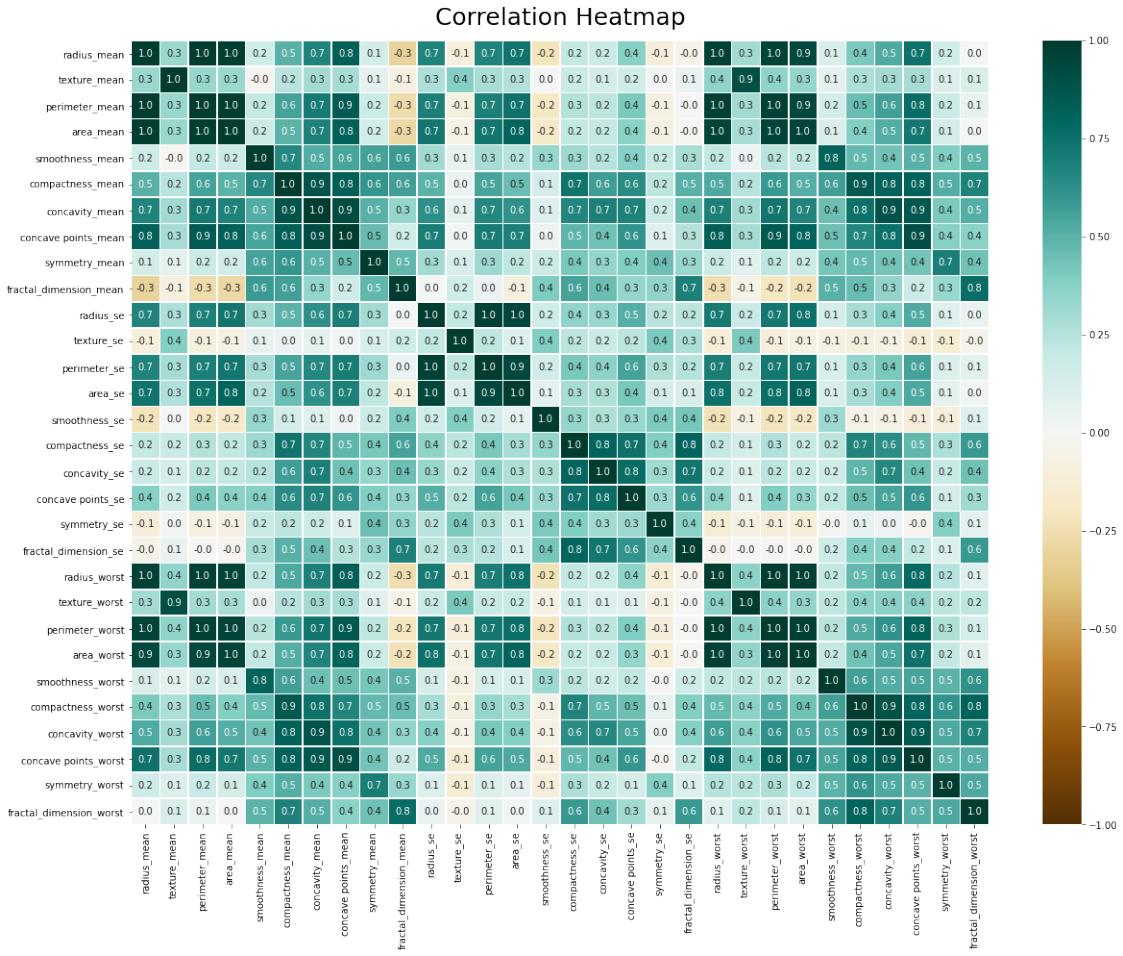
Here we can see the distribution of the numerical features of our dataset and the value of the skewness.
Skewness is a measure of the asymmetry of the probability distribution of a real-valued random variable about its mean. The skewness value can be positive, zero, negative, or undefined.
For a unimodal distribution, negative skew commonly indicates that the tail is on the left side of the distribution, and positive skew indicates that the tail is on the right. If, instead, the value is zero value means that the tails on both sides of the mean balance out overall; this is the case for a symmetric distribution.



This is a kernel density estimate (KDE) plot, which is a method for visualizing the distribution of observations in a dataset. It is analogous to a histogram, but, while the latter aims to approximate the underlying probability density function that generated the data by binning and counting observations, KDE presents a different solution to the same problem: rather than using discrete bins, a KDE plot smooths the observations with a Gaussian kernel, producing a continuous density estimate.

As we can see, many of the variables have distinct distributions from Benign to Malignant so we could think that these are going to be relevant variables. On the other hand, those variables with almost the same distribution in Benign and Malignant probably won't be as much important as others are. This is the reason why we are going to implement the feature selection.

1.3 Feature Selection



Basically, if two features are highly correlated with each other (linear correlation close to 1) it could be a good choice to drop one of them.

As we can see from the correlation matrix:

radius_mean, **perimeter_mean** and **area_mean** are correlated with each other -> we choose *area_mean*. Empirically, if we had to give a stronger guess, we would say that experimental measures on area might have lower uncertainties than measures of radius or perimeter. Another criterion to choose area_mean, backed up by data this time, is that the feature seems to express more differences between malignant and benign tumors on KDE plot. Based on these arguments, we also decide for the other variables.

Compactness_mean, **concavity_mean** and **concave points_mean** are correlated -> choose *concavity_mean*.

radius_se, **perimeter_se** and **area_se** are correlated -> choose *area_se*.

radius_worst, **perimeter_worst** and **area_worst** are correlated -> choose *area_worst*.

Compactness_worst, **concavity_worst** and **concave points_worst** are correlated -> choose *concavity_worst*.

Compactness_se, **concavity_se** and **concave points_se** are correlated -> choose *concavity_se*.

texture_mean and **texture_worst** are correlated -> choose *texture_mean*.

area_worst and **area_mean** are correlated -> choose *area_mean*.

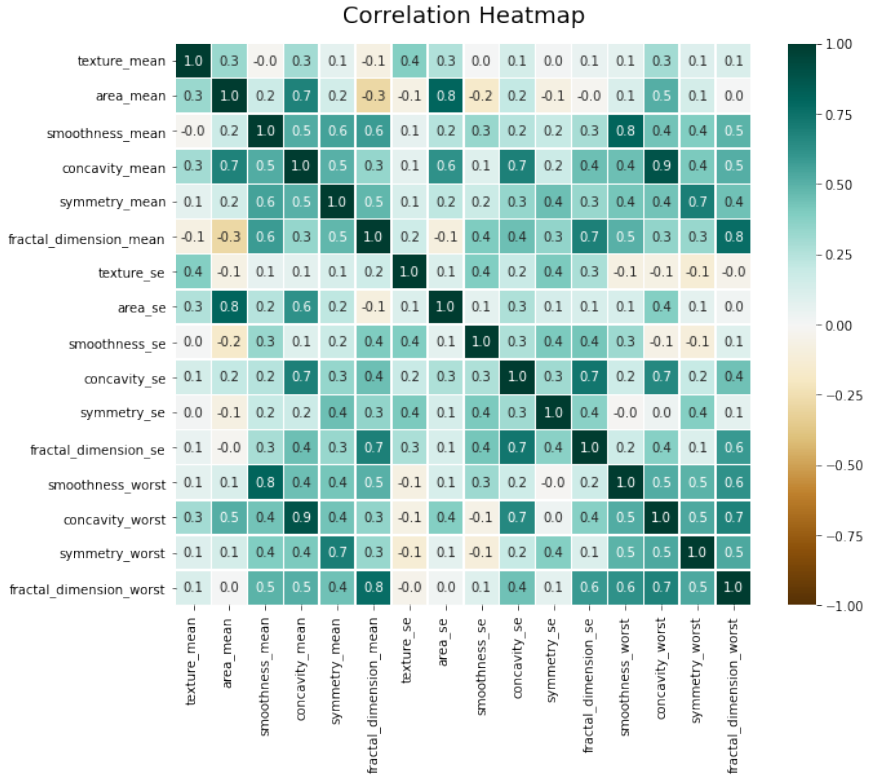
Let's have a look at our new dataset:

[9]:

	texture_mean	area_mean	...	symmetry_worst	fractal_dimension_worst
0	10.38	1001.0	...	0.4601	0.11890
1	17.77	1326.0	...	0.2750	0.08902
2	21.25	1203.0	...	0.3613	0.08758
3	20.38	386.1	...	0.6638	0.17300
4	14.34	1297.0	...	0.2364	0.07678
..	...	...	...	...	...
564	22.39	1479.0	...	0.2060	0.07115
565	28.25	1261.0	...	0.2572	0.06637
566	28.08	858.1	...	0.2218	0.07820
567	29.33	1265.0	...	0.4087	0.12400
568	24.54	181.0	...	0.2871	0.07039

[569 rows x 16 columns]

After dropping highly (linearly) correlated features, we end up with 16 features, almost uncorrelated between each other, as we can see from the following heatmap:



1.4 Standardizing Data

Standardize data in order to have mean=0 and sdv=1:

[ ]:

	texture_mean	area_mean	...	symmetry_worst	fractal_dimension_worst
0	-2.073335	0.984375	...	2.750622	1.937015
1	-0.353632	1.908708	...	-0.243890	0.281190
2	0.456187	1.558884	...	1.152255	0.201391
3	0.253732	-0.764464	...	6.046041	4.935010
4	-1.151816	1.826229	...	-0.868353	-0.397100
..	...	...	...	...	...
564	0.721473	2.343856	...	-1.360158	-0.709091
565	2.085134	1.723842	...	-0.531855	-0.973978
566	2.045574	0.577953	...	-1.104549	-0.318409
567	2.336457	1.735218	...	1.919083	2.219635
568	1.221792	-1.347789	...	-0.048138	-0.751207

[569 rows x 16 columns]

Statistics:

[ ]:

	mean	std	...	75%	max
texture_mean	1.049736e-16	1.00088	...	0.584176	4.651889
area_mean	-1.900452e-16	1.00088	...	0.363507	5.250529
smoothness_mean	1.490704e-16	1.00088	...	0.636199	4.770911
concavity_mean	-1.338511e-16	1.00088	...	0.526062	4.243589
symmetry_mean	2.081912e-16	1.00088	...	0.530779	4.484751
fractal_dimension_mean	5.408679e-16	1.00088	...	0.470983	4.910919
texture_se	-9.912009e-17	1.00088	...	0.466552	6.655279
area_se	-1.088760e-16	1.00088	...	0.106773	11.041842
smoothness_se	4.426014e-16	1.00088	...	0.368355	8.029999
concavity_se	1.678017e-16	1.00088	...	0.336752	12.072680
symmetry_se	1.523874e-16	1.00088	...	0.355692	7.071917
fractal_dimension_se	-5.658430e-17	1.00088	...	0.288642	9.851593
smoothness_worst	-2.189227e-16	1.00088	...	0.597545	3.955374
concavity_worst	1.143393e-16	1.00088	...	0.531141	4.700669
symmetry_worst	1.670212e-16	1.00088	...	0.450138	6.046041
fractal_dimension_worst	2.321908e-16	1.00088	...	0.450762	6.846856

[16 rows x 7 columns]

2 Supervised Learning

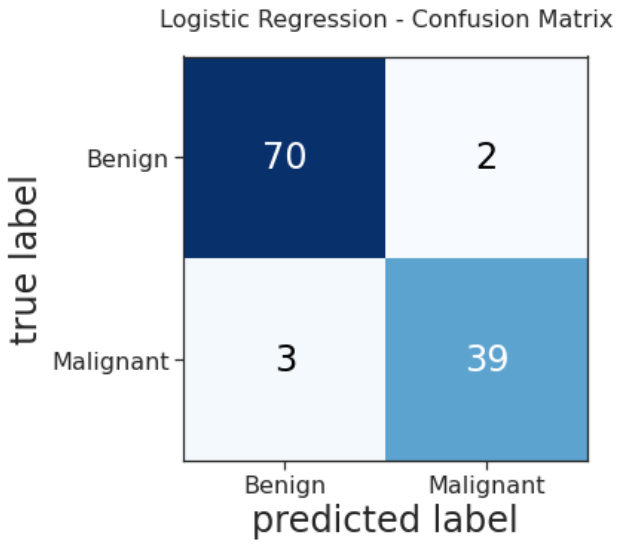
Supervised Learning is a machine learning technique in which the algorithms are designed to learn by example. In fact, the training data consist of inputs paired with the correct outputs. During training the algorithm will search for patterns in the data that correlate with the desired outputs. After training the algorithm will take in new unseen inputs and will determine which label the new inputs will be classified as, based on prior training data. The objective of a supervised learning model is to predict the correct label for newly presented input data.

We are going to implement the following supervised algorithms:

- Logistic Regression
- SVM
- Gaussian Naive Bayes
- Decision Tree
- Random Forest
- Extra Tree
- KNN

2.1 Logistic Regression

Logistic regression is a classification algorithm used to assign observations to a discrete set of classes. It transforms its output using the logistic (sigmoid) function (which can assume values between 0 and 1) to return a probability value.



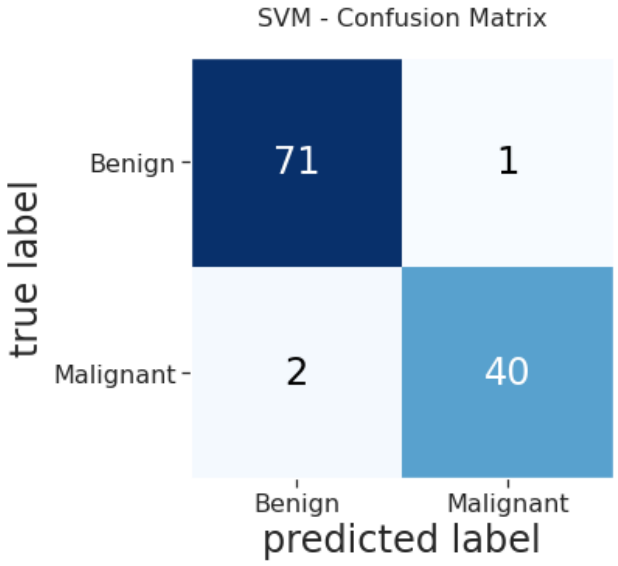
The trace of the matrix gives us the total of right classified units, therefore:

- 70 tumors are correctly classified as benign;
- 39 tumor are correctly classified as malignant.

Accuracy of Logistic Regression is: 95.61%

2.2 SVM

SVM algorithm is based on the idea of finding a hyperplane that best separates the features of one class from the others, i.e. the one which separates the classes being the farthest from the training observations, which is measured by the margin. This algorithm allows for some observations to be on the incorrect side of the margin or in the incorrect side of the hyperplane, defining a "soft" margin, in order to classify data even if they are not linearly separable.



Classification:

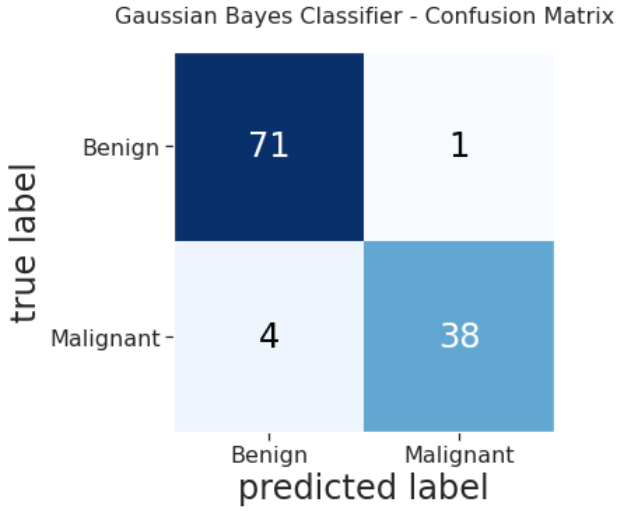
- 71 tumors are correctly classified as benign;
- 40 tumor are correctly classified as malignant.

Accuracy of SVM is: 97.37%

2.3 Gaussian Naive Bayes

Naive Bayes algorithms assume that features are independent of each other and there is no correlation between them (this is the reason why the algorithm is called *naive*), but this is not the case in real life, reason why this could lead to less accurate predictions.

Gaussian Naive Bayes algorithm is used with continuous data that follow a Gaussian normal distribution.



Classification:

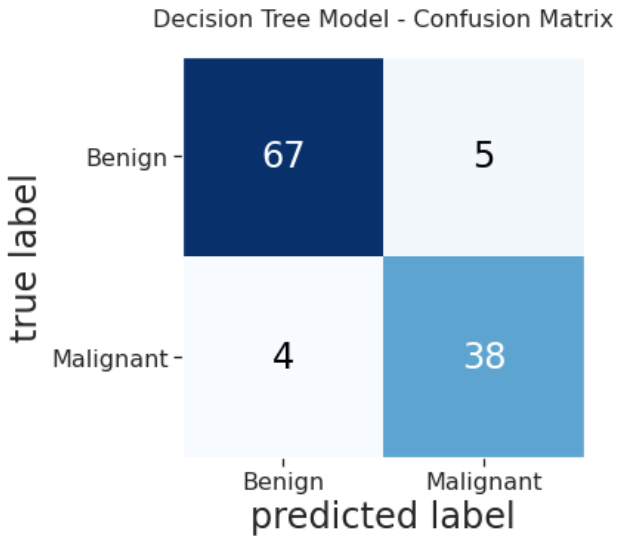
- 71 tumors are correctly classified as benign;
- 38 tumor are correctly classified as malignant.

Accuracy of Gaussian Classifier is: 97.37%

2.4 Decision Tree

DTs are ML algorithms that progressively divide datasets into smaller data groups based on a descriptive feature, until they reach sets that are small enough to be described by some label.

DTs are composed of nodes, branches and leafs. Each **node** represents a *feature*; each **branch** represents a *rule* (or decision); each **leaf** represents an *outcome*. The **depth** of a Tree is defined by the number of nodes, not including the root node.



Classification:

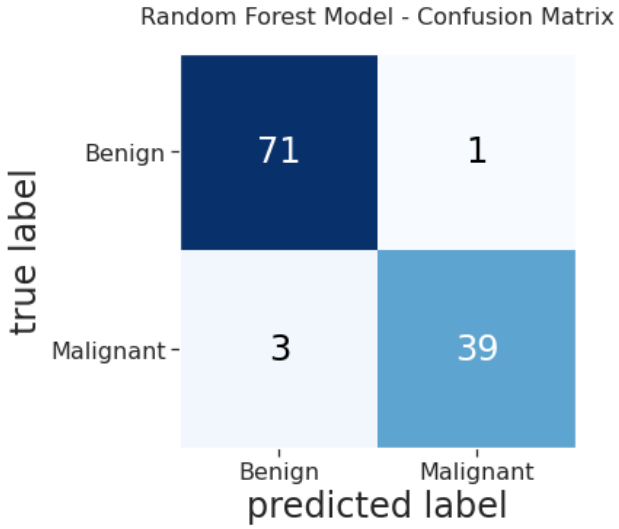
- 67 tumors are correctly classified as benign;
- 38 tumor are correctly classified as malignant.

Accuracy of Decision Tree Classifier is: 92.11%

2.5 Random Forest

Random Forest is an ensemble method. These kind of methods combine different DTs to improve the performance of a single DT, because it can suffer from high variance, which means that a small change in the data can result in a very different set of splits, making interpretation somewhat complex, and bias if some classes dominate over others (this is a problem in unbalanced datasets).

Random Forest is an extension over Bagging, which aims to reduce the variance of a DT creating in parallel random subsets of the training data (where any observation has the same probability to appear in a new subset data) and also taking a random selection of features rather than using all features to grow DTs. Next, each collection of subset data is used to train DTs, resulting in an ensemble of different DTs. Finally, an average of all predictions of those different DTs is used, which produces a more robust performance than single DTs.



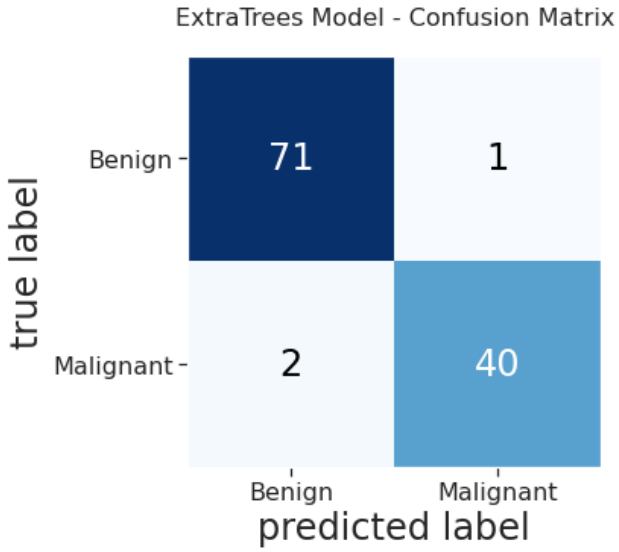
Classification:

- 71 tumors are correctly classified as benign;
- 39 tumor are correctly classified as malignant.

Accuracy of Random Forest Classifier is: 96.49%

2.6 Extra Tree

Extra Tree, also known as *Extremely Randomized Tree*, is an algorithm similar to the Random Forest one, from which differs, however, in the fact that it uses the whole original sample as opposed to subsampling the data as Random Forest does. Another difference is in how the nodes are split. While Random Forest is built to always choose the best possible split, Extra Tree chooses random splits, adding one further step of randomization.



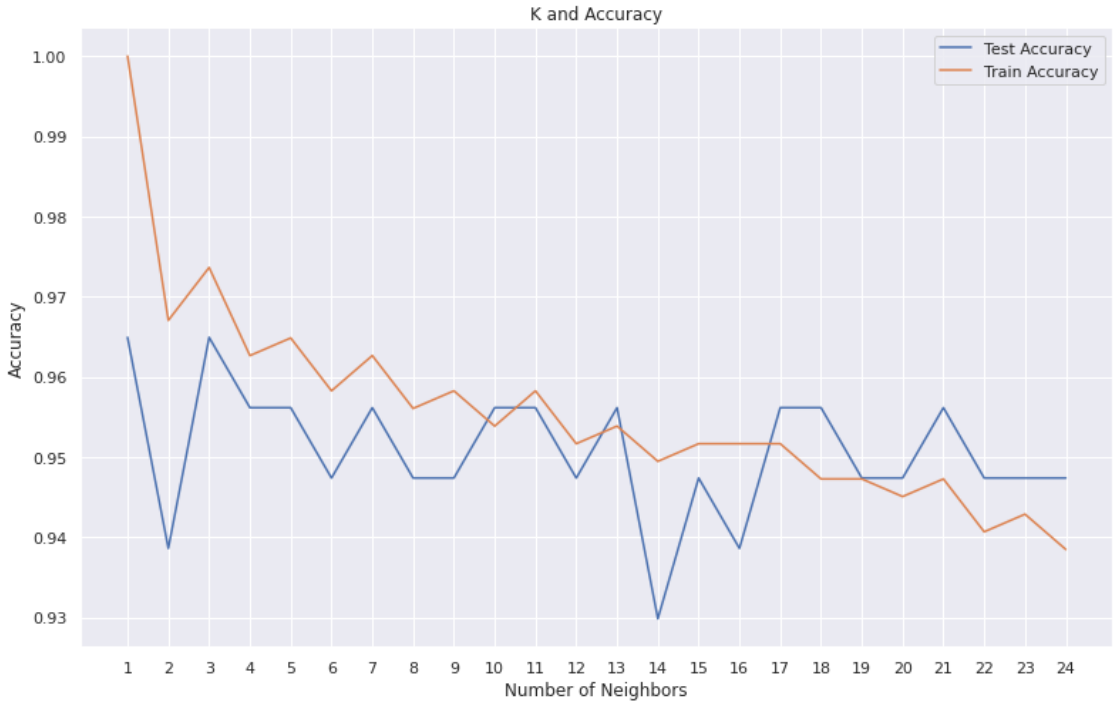
Classification:

- 71 tumors are correctly classified as benign;
- 40 tumor are correctly classified as malignant.

Accuracy of Extra Trees Classifier is: 97.37%

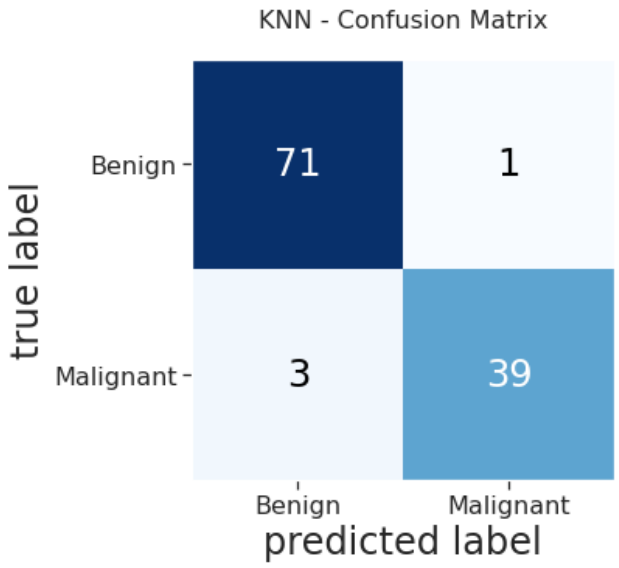
2.7 K-Nearest Neighbors

The KNN algorithm assumes that similar units are close to each other. Determine the class of a data point based on the majority voting principle: if k is set to 5, the classes of the 5 closest points w.r.t. the one taken into account are checked and the prediction is made according to the majority class.



Maximum accuracy: 96.49% at K = 1

Although the highest test accuracy results in correspondence of k=1, we note that it corresponds also to a training accuracy equal to 1, which implies the model to overfit. Therefore, we prefer to implement the algorithm with k=3, in corrispondence of which the test accuracy is always 96.49%.

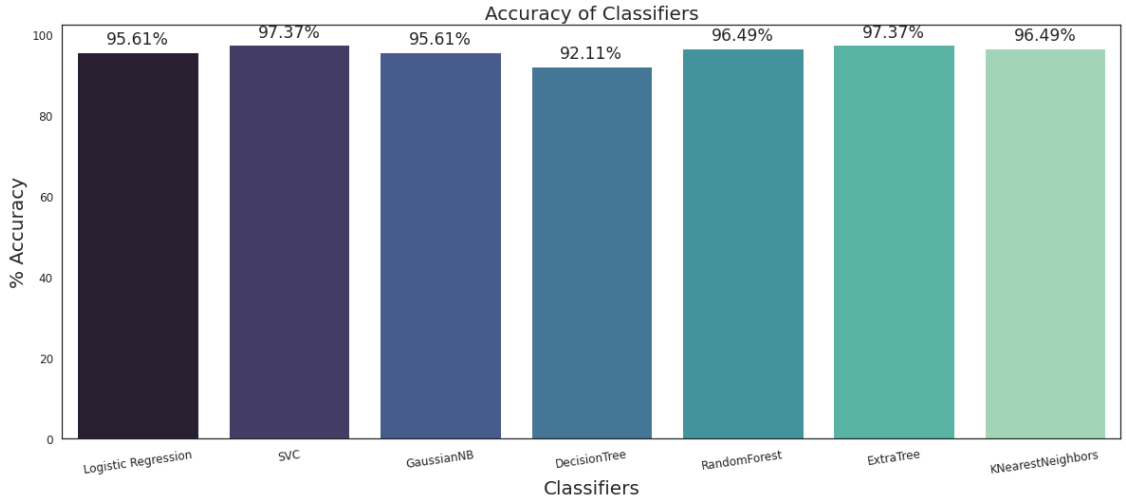


Classification:

- 71 tumors are correctly classified as benign;
- 39 tumor are correctly classified as malignant.

Accuracy of KNN Classifier is: 96.49%

2.8 Models Comparison



As we can see SVC and Extra Tree have the same accuracy: 97.37%.

2.9 SMOTE

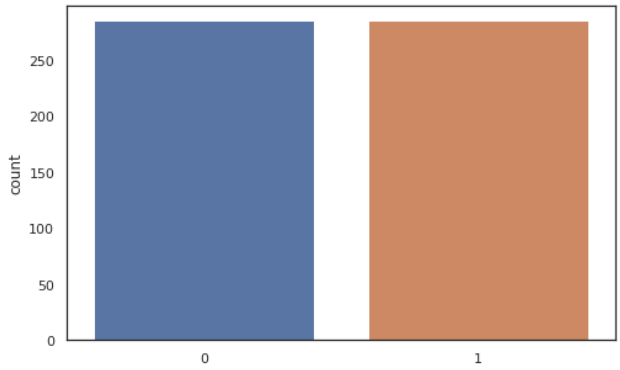
When the dataset is imbalanced - as in this case- models could become biased towards selecting the majority group. This is the reason why we are going to implement the **Synthetic Minority Over-sampling** technique which creates new data instances of the minority groups by copying existing minority instances and making small changes to them.

Check how many malignant and benign tumors are there BEFORE oversampling:

```
The number of malignant tumors is 170
The number of benign tumors is 285
```

Check how many malignant and benign tumors are there NOW:

```
The number of malignant tumors is 285
The number of benign tumors is 285
```



Now the data is balanced and we are going to compare the classifiers after fitting them to the balanced data.

2.9.1 Models comparison

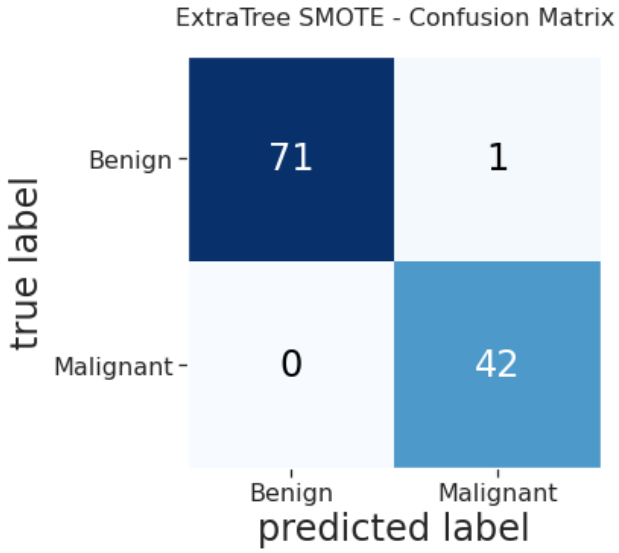
As we can see from the plot Extra Tree has the highest accuracy: 99.12%.

2.10 Baseline vs SMOTE models

	Classifiers	Accuracy_basic	Accuracy_smote
0	Logistic Regression	0.956140	0.964912
1	SVC	0.973684	0.982456
2	GaussianNB	0.956140	0.964912
3	DecisionTree	0.921053	0.929825
4	RandomForest	0.964912	0.973684
5	ExtraTrees	0.973684	0.991228
6	KNearestNeighbors	0.964912	0.964912

Now we can state that the best model to classify tumors into benign or malignant is the **Extra Tree** with the 99% of accuracy.

Let's have a look at it:



	precision	recall	f1-score	support
0	1.00	0.99	0.99	72
1	0.98	1.00	0.99	42
accuracy			0.99	114
macro avg	0.99	0.99	0.99	114
weighted avg	0.99	0.99	0.99	114

TP: correctly predicted samples having cancerous cells ($1 \rightarrow 1$);

TN: correctly predicted samples not having cancerous cells ($0 \rightarrow 0$);

FP: incorrectly predicted samples not having cancerous cells ($0 \rightarrow 1$);

FN: incorrectly predicted samples having cancerous cells ($1 \rightarrow 0$).

Precision: ability of a classification model to identify only relevant instances (in other words, it expresses the proportion of the data points our model says was relevant actually were relevant).

Recall: ability of a classification model to identify all the data points of interest in a dataset ((e.g. malignant tumors).

F1 score: single metric combining recall and precision using the harmonic mean (harmonic mean is used instead of a simple average because it punishes extreme values).

$$\text{Precision} = \frac{TP}{TP + FP}$$

$$\text{Recall} = \frac{TP}{TP + FN}$$

$$\text{F1} = 2 * \frac{precision * recall}{\text{precision + recall}}$$

3 Unsupervised Learning

Unsupervised learning is a machine learning technique in which no labels are given to the learning algorithm, leaving it on its own to find patterns in its input. It can be a goal in itself (discovering hidden patterns in data) or a means towards an end (feature learning - learn the features and use them to perform a specific task).

Unsupervised learning is mostly used for clustering. Clustering is the act of creating groups with differing characteristics; it attempts to find various subgroups within a dataset. As this is unsupervised learning, we are not restricted to any set of labels and are free to choose how many clusters to create. This is both a blessing and a curse: picking a model that has the correct number of clusters (complexity) has to be conducted via an *empirical* model selection process.

As regard to the clustering algorithms, we are going to implement the following ones:

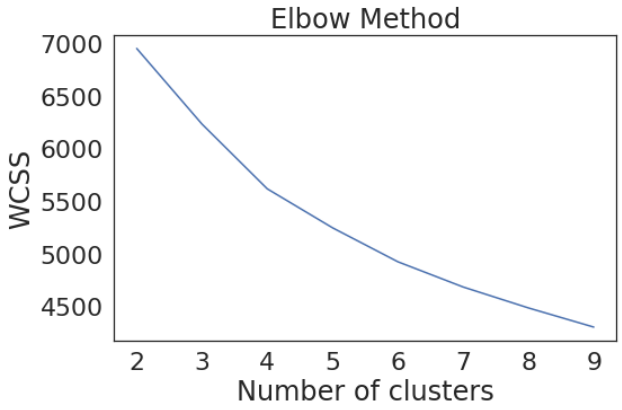
- K-means
- DBSCAN
- Hierarchical Clustering

Let's start to implement our algorithms and, just to be clear, we are going to use our **data_std** dataset, the standardized one, containing the most relevant features found in the Feature Selection step, except, of course, the target variable, which will be used as external information to compare the clustering results with the actual situation.

3.1 K-means

K-means is the most used partitioning algorithm. The procedure follows a simple and easy way to classify *n* units of the dataset into a certain number of K clusters, fixed a priori. The main idea is that the units within the same cluster are as similar as possible, whereas units from different clusters are as dissimilar as possible. It does this through an iterative process which aims to minimize an objective function: the within sum of squares, **WSS**: sum of squared Euclidean distances between the points assigned to the cluster and the cluster mean - it is a measure of within cluster dissimilarity, reason why it must be minimized by the algorithm.
This iterative process starts by randomly select k centroids, one for each cluster, which should be placed in a smart way because of different location causes different result (the better choice is to place them as much as possible far away from each other).
The next step is to take each point belonging to the dataset and associate it to the nearest centroid.
At this point new centroids are calculated, computing the new mean value of all the data points in the cluster.
This procedure is iterated until the centroids do not change their location anymore or the maximum number of iterations is reached.

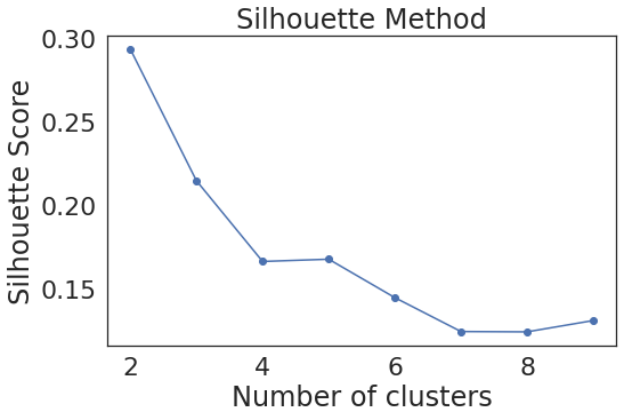
Let's have a look at which could be the right number of clusters.



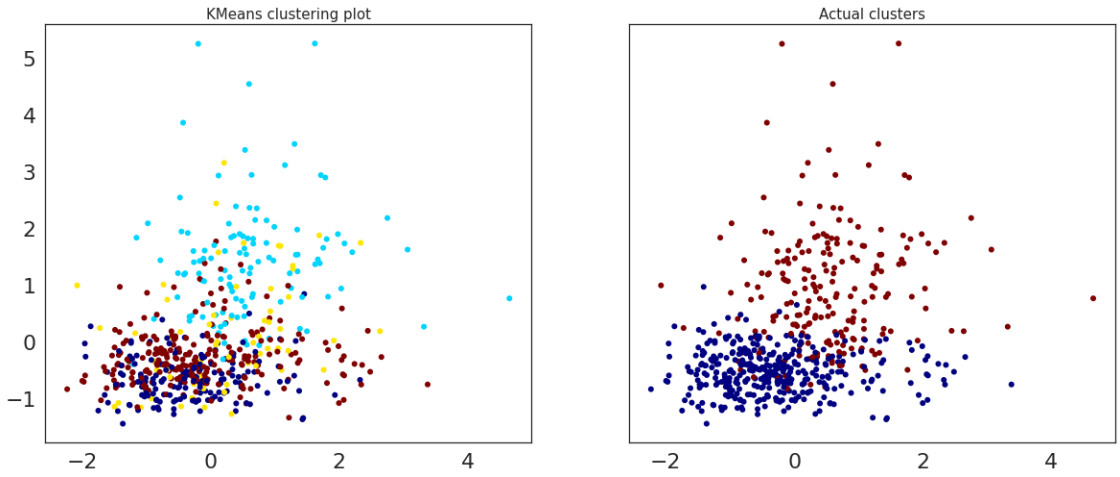
The elbow in the plot is not very clear. We could guess it is in corrispondence of 4, but it's better to try also with the Silhouette Method.

Silhouette is a measure of how close each point in one cluster is to points in the neighboring clusters, ranging between [-1, 1].

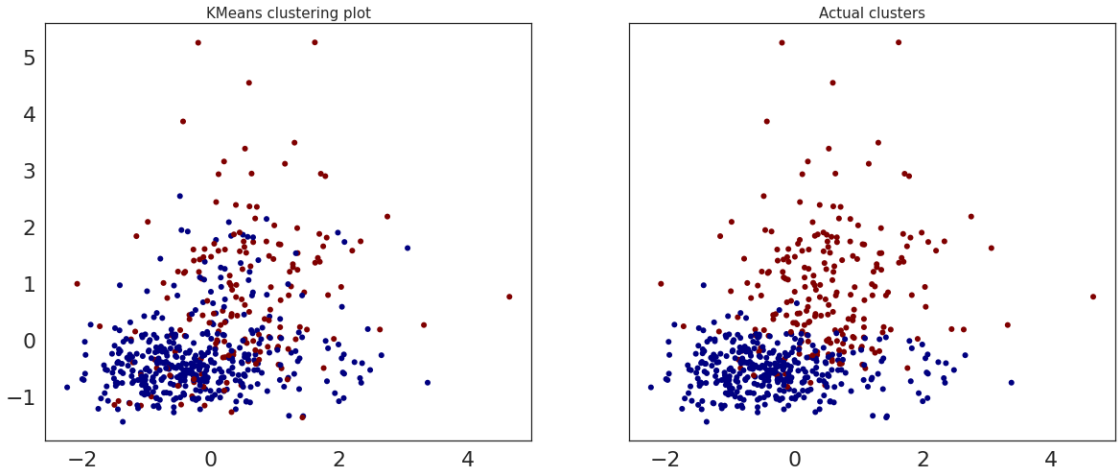
Silhouette coefficients near **+1** indicate that the sample is far away from the neighboring clusters; value of **0** indicates that the sample is on or very close to the decision boundary between two neighboring clusters; **-1** indicate that those sample might have been assigned to the wrong cluster.



Now the right number of clusters seems to be 2, but let's implement the algorithm both for k=4 and k=2.



As we can see form the "KMeans clustering plot", there is a strong overlapping between the 4 clusters, which means that there are not 4 groups in the data! This is even more clear comparing the plot with the one representing the target variable, *diagnosis*: we can see that it is distributed into 2 clusters (in fact we know that it can assume only 2 values), which are well defined, not overlapped between each other. Therefore, we can state that the data should be clustered into 2 clusters... let's try!



Now, with K=2, there is an improvement in the plot, as the 2 clusters are much less overlapped w.r.t. to the previous 4, and the situation is closer to the actual one. Therefore, we can state that the right number of clusters for K-means is 2.

3.2 Hierarchical Clustering

Hierarchical clustering means creating a tree of clusters by iteratively grouping (**agglomerative clustering**) or dividing (**divisive clustering**) data points. These algorithms do not require the number of clusters as input and the cluster hierarchy built by them is displayed by means of **dendrograms**.

Divisive clustering is a *top-down* approach and it is not commonly used in real life; *Agglomerative clustering* is the most used and it is a *bottom-up* approach, in which each data point is assumed to be a separate cluster at first, then the similar clusters are iteratively combined.

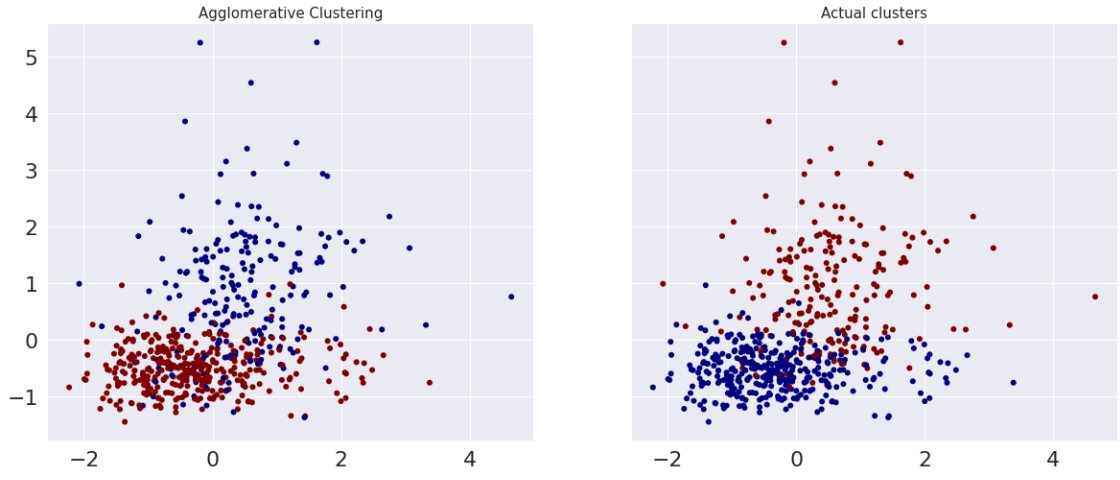
In order to decide which clusters should be combined or where a cluster should be split, the **linkage criterion** is used: it specifies the dissimilarity of units as a function of the pairwise distances of observations in the sets. We can distinguish between 4 different linkage criterions:

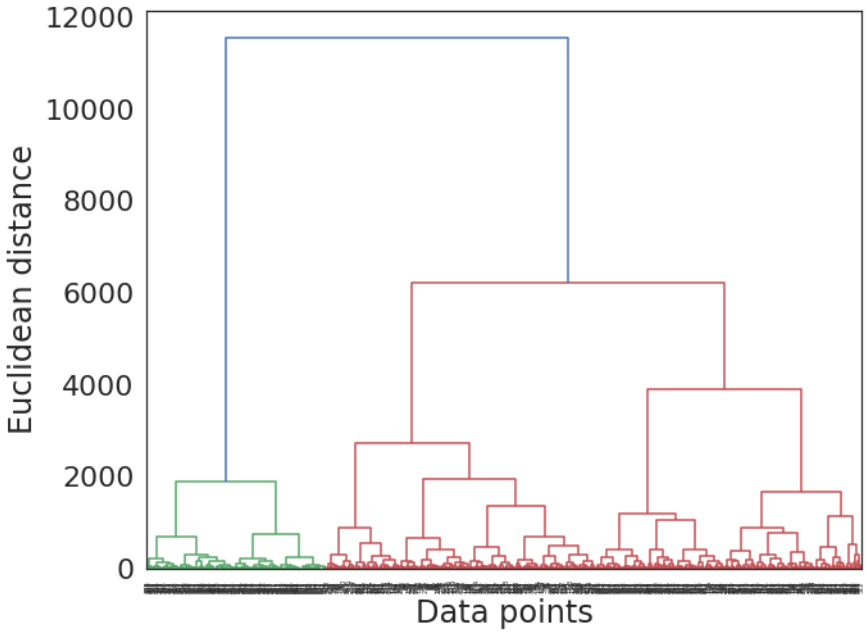
1. **Ward's linkage:** it minimizes the variance of the clusters being merged - total within-cluster variance/ total sum of squares, TSS. Minimum increase in total within-cluster variance is aimed (this increase is a weighted squared distance between cluster centers - centroids).
2. **Average linkage:** the distance between two clusters is defined as the average of the distances between all observations in the two clusters.
3. **Complete linkage:** the distance between two clusters is defined as the maximum distance between all data points in the two clusters.
4. **Single linkage:** the distance between two clusters is defined as the smallest distance between all data points in the two clusters.

Searching for the best parameters we find this:

[132]:

	Number of clusters	Linkage	Distance	ARI
0	2	ward	euclidean	0.689242





In the dendrogram displayed above, each leaf corresponds to one object. As we move up the tree, objects that are similar to each other are combined into branches, which are themselves fused at a higher height.
The height of the fusion, provided on the vertical axis, indicates the (dis)similarity/distance between two objects/clusters. The higher the height of the fusion, the less similar the objects are. This height is known as the *cophenetic distance* between the two objects.

3.3 Cluster Validation

Until now, we have examined the distribution of our data in clusters in a 2D space, comparing it with the distribution of our target variable, but we do not know whether these two clusters are actually able to separate the tumor into *benign* and *malignant*.

To understand this we are going to implement the **External Cluster Validation**, which consists in comparing the results of CA with an externally known results, the class labels in our case.
Specifically, we are going to look at the **Adjusted Rand Index** and at the **Confusion Matrix**.

3.3.1 Adjusted Rand Index

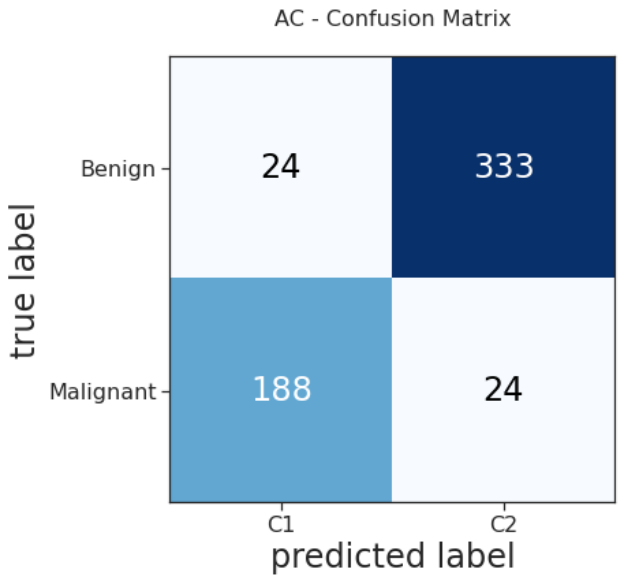
The adjusted Rand index is the corrected-for-chance version of the Rand index. More specifically, it is a similarity measure between two clusterings by considering all pairs of samples and counting pairs that are assigned in the same or different clusters in the predicted and true clusterings. Therefore, it attempts to express what proportion of the cluster assignments is 'correct'; it is bounded between -1 and 1:
-closer to 1 is good;
- closer to -1 is bad.

Let's have a look at the ARI value for both the algorithms:

- ARI for Kmeans: 0.44
- ARI for AC: 0.69

The Agglomerative Clustering has an ARI higher than the one for K-Means Clustering, reason why we can state that the best clustering algorithm for our data is the former.

3.3.2 Confusion Matrix



- Cluster 1 contains:
  - 24 benign tumors (out of 357);
  - 188 malignant tumors (out of 212).
- Cluster 2 contains:
  - 333 benign tumors (out of 357);
  - 24 malignant tumors (out of 212).

Therefore, according to the Agglomerative Clustering the majority of malignant tumors belong to Cluster 1, while the majority of benign tumors belong to Cluster 2.