ECS648U/ECS784P - Data Analytics - 2018/19

A Machine Learning Classification and Exploratory Data Analysis of Breast Cancer

Word Count: 4190 (without table of contents and appendices)

Group size: 1

Daniel Cristian Dumitrescu 150178521

Table of Contents

Introduction

Breast cancer, being ranked first and second out of all diseases reported in women, is an important threat to women's life and physical and psychological health [1]. As it represents around 20% out of all the cancer cases that are reported in the United States [2]. this topic represents a great value for research. As seen in the last 10 years, the trend of breast cancer reports is increasing, with more and more cases every year [3]. Of course, this unveils a big amount of data which, after a rigorous analysis, can provide very important information on the type of cancerous cells. This is of significant use for both academics and practitioners looking into research, and also for the use of data science techniques and machine learning algorithms in this specific domain.

Different papers have analysed the implementation of machine learning algorithms used in the classification of different cancerous cells [4],[5] and presented significant results. As almost every woman nowadays is affected by breast cancer, whether by constant worrying about the disease or by diagnosis, this paper proposes another study on the said topic, which focuses on the exploratory analysis of the Wisconsin Diagnostic Breast Cancer (WDBC) dataset [6] for a better understanding of the relationship between the variables, and on the identification of the machine learning algorithm that is best-suited for the problem at hand.

Background Information and methodology

Breast Cancer and the Differences Between Benign and Malignant Tumours

Breast Cancer is defined as an uncontrolled growth of epithelial cells with origins in the breast lobules [7], and a tumour, is a swelling which is also caused by an abnormal growth of cells, which is classified in two: benign or malignant [7] If a tumour is found, usually further tests are proposed in order to find if it is cancerous or not, as not every tumour is of cancerous type. Benign cells are non-cancerous and won't spread to another part of the body, but malignant cells are cancerous and can spread to surrounding tissue.

External Libraries

For this paper, the Python programming language is being used for creating all the visualisation plots and also the implementation of the Machine Learning algorithms. In order to do this, several external libraries are being used as following:

Matplotlib

Matplotlib is a powerful library used for 2D plotting and visualisation of data. A nice function of this library is that the graphs created with Matplotlib can be changed in size, shape and also can be scaled to view only a specific area of interest, which can be made through the written code or through the interactive interface.

NumPy

Another important Python library is NumPy, which adds support for large, multi-dimensional arrays and matrices, along with a number of high-level mathematical functions which can operate on these arrays. The "ndarray" is the core functionality of this library, those arrays being stridden views on memory [8], which is, of course, in contrast with the built-in list data structure which the Python programming language is based on. The main difference between those two is that the built-in Python lists are homogeneously typed, meaning that all the elements must have the same type, but, the NumPy arrays can be created with a mixture of types (string, float, etc).

Pandas

This is a free Python library used for data manipulation and analysis, which offers specific data structures and operations for manipulating numerical tables [9]. In this paper, the main Pandas data structure used is the DataFrame object, which is a 2D labelled data matrix that supports a range of functions and operations on it.

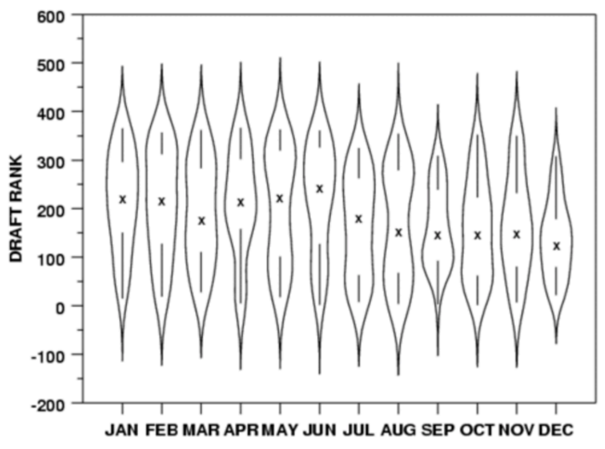
Scikit-Learn

Scikit-Learn is a general purpose library which is used for the implementation of Machine Learning algorithms for Python [10]. Of course, it provides a number of classification, clustering and regression algorithms, being designed to interoperate with some other Python libraries like NumPy and SciPy[11]. There are a number of ways in which the Machine Learning algorithms can be implemented, but this paper is making use of the Scikit-Learn library as its interface is easy to work with, and also it is easy to implement it.

Exploratory Analysis

Violin Plots

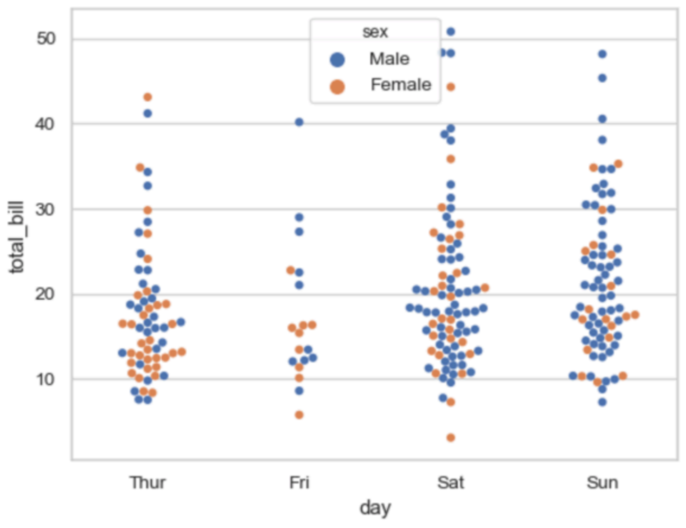
A violin plot is a very useful method of plotting data, which is formed of a rotated kernel density plot on each side [12]. Typically, this visualisation tool includes a marker which shows the median, a marker which presents the interquartile range, and all the sample points, of course, if the number of data is not too high.



Example of a violin plot.
Source: https://en.wikipedia.org/wiki/Violin_plot#/media/File:Violin_plot.gif

Swarm Plots

The swarm plot presents the points adjusted in a way that they do not overlap giving a better representation of the distribution of values, although it does not scale as well to large numbers of observation (both in terms of ability to show all the points but also in computational resources needed for such a task). A swarm plot can be very useful when it complements a violin plot in cased where one wants to show all observations along with some representation of the underlying distribution. An example of the swarm plot can be seen below.



Swarm Plots
Accessed at https://seaborn.pydata.org/generated/seaborn.swarmplot.html

Correlation Matrix

A correlation matrix is a table that explains the correlation coefficient between two or more variables. The correlation between two specific variables is shown in a cell table. The correlation matrix is usually used to summarize data, and also to choose the input into a more advanced analysis technique.

It can be represented with a matrix, with the same variables shown in the rows and columns, and on the main diagonal are only values of 1, as it represents the correlation between a specific feature with itself. Also, the matrix is symmetrical, with the same correlation shown above the main diagonal being a mirror image of those below the main diagonal. When there is a positive/negative correlation, both variables change in the same/different direction, and when the correlation is neutral, no relationship exists in the change of variables. As the positive correlation can be between 0 and 1, the relationship between the variables is stronger when the number is closer to 1.

Supervised Learning

As a subset of Machine Learning, supervised learning is the task of learning different functions that can map an input to an output based on input-output pairs [13], and it requires labelled data, contrary to the unsupervised learning techniques which do not need labels for the given data. This learning method helps in determining how to differentiate between different classes (in this paper between benign or malignant cells) using different models and represents substitution to human judgement and manual rules. Usually, there are several steps which one must follow when approaching a supervised learning method: gathering the data, preparing the data, choosing one or more models, training, evaluation, parameter tuning and prediction [14].

All the algorithms implemented in this paper are being measured using accuracy on the training set, testing set, and precision, recall, f-score and support.

i) The precision is the ratio

$$\frac{t_p}{t_p + f_p}$$

where t_p is the number of true positives and f_p the number of false positives. So intuitively, the precision is the ability of the classifier not to label as positive a sample that is negative.

ii) The recall is the ratio

$$\frac{t_p}{t_p + f_n}$$

where t_p is the number of true positives and f_n the number of false negatives. So, the recall shows the ability of the classifier to find all the positive samples.

iii) The F-beta score can be interpreted as a weighted harmonic mean of the precision and recall, where an F-beta score reaches its best value at 1 and the worst score at 0 [17].

iv) The number of occurrences of each class in y_true is called support.

Naïve Bayes

Naïve Bayes is a commonly used machine learning classifier, and it is a probabilistic classifier that makes classifications using the Maximum A Posteriori decision rule in a Bayesian setting [15]. The goal of a probabilistic classifier is when having features from x_0 to x_n and c_0 to c_n classes to determine the probability of the features occurring in each class, and to return the most likely class. So, for each individual class, it is necessary to calculate P(c_i | x_0, …, x_n). For doing this, the Bayes rule is being used, which can be seen below.

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

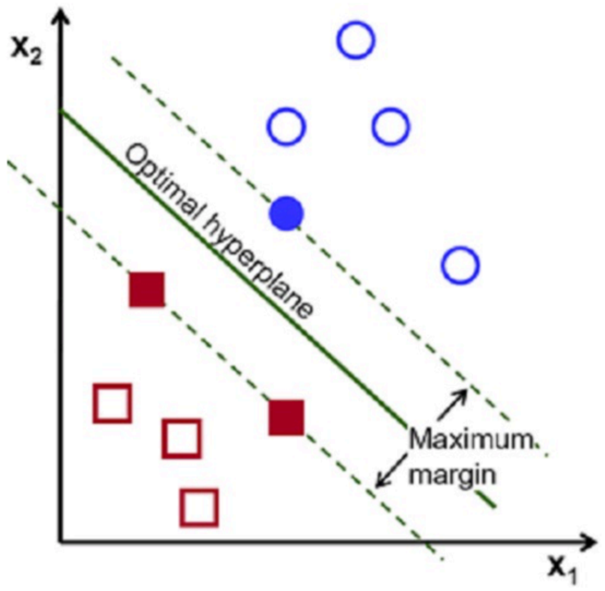
KNN Algorithm

One of the simplest and easiest to understand machine learning algorithms is the K-nearest neighbour. A new data point is classified by a majority of its neighbours, with it being assigned to the class most common amongst its K neighbours measured by a distance function. If the K is set to 1, then the new entry will be assigned to the class of its neighbour. Usually, the most used distance function is the Euclidean distance shown below.

$$\sqrt{\sum_{i=1}^{k}(x_i - y_i)^2}$$

Support Vector Machines

Another supervised learning algorithm is the Support Vector Machine (SVM), which uses a given dataset in order to solve different problems by converting them into linearly separable problems [16], working as a binary classifier. It classifies a given input in two classes, trained with a learning algorithm for optimization theory that implements a learning bias. At the first approximation, the SVM finds a separating line (called also a hyperplane) between data points of two classes. So, it can be said that it takes data as an input, and as an output creates a line that separates the classes if that is possible. A hyperplane in an n-dimensional Euclidean space is a flat, n-1 dimensional subset of that space that divides the space into two disconnected parts [17]. The hyperplane is a means to divide the classes, so when a new data point comes along, the algorithm decides whether it is in one class or another.



Support Vector Machines by Dr. Lance Eliot, the AI Trends Insider

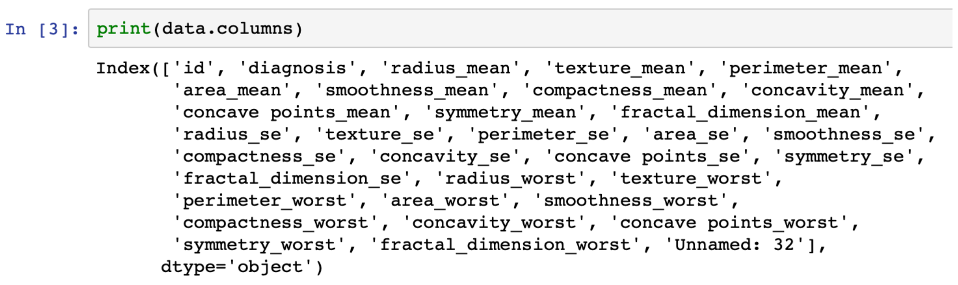
Aims

The aim of this paper is to perform different classification algorithms and compare their efficiency in determining whether a specific cell is benign or malignant. Before doing this, a visual approach of the dataset is proposed. The aim of this visual inspection is to better understand the data, the correlation between the features, and which of them is more likely to determine the type of the cell.

Data Used

Dataset

For this paper, the Breast Cancer Wisconsin Data Set is used [6]. According to [6], the dataset formed of 570 instances and 32 features was computed from a digitized image of a fine needle aspirate (FNA) of a breast mass, and the features describe the different characteristics of the cell nucleus. The features analysed in this dataset are the following: (1) area, (2) compactness, (3) concavity, (4) concave points, (5) fractal dimension, (6) symmetry, (7) smoothness, (8) radius, (9) texture, and (10) perimeter, with each of those ten features having three information [6]: mean, standard error and worst (which is described as the mean of the three largest values), plus ID and Diagnosis (B for Benign and M for Malignant). Each entry of the dataset is a cell from a different person which presents a tumour.

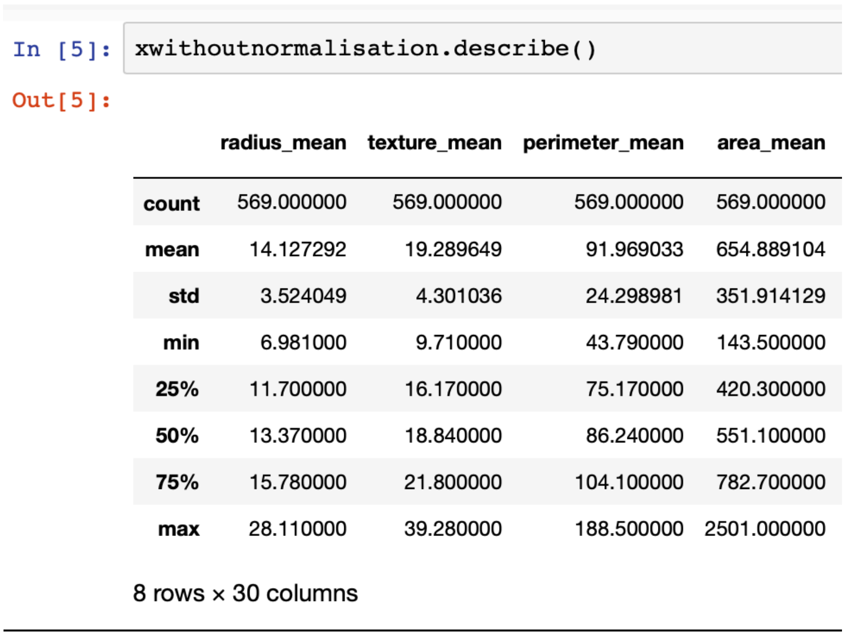


```
In [3]: print(data.columns)

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

Data pre-processing and normalisation

In this dataset, there are two features that are not needed for the analysis, and so, for a reduction to the dataset and an easier manipulation, those are removed. The features not needed are the 'ID' and 'Unnamed: 32'. Also, the dataset is split into two as follows: the labels ('diagnosis') are assigned to the variable y and the rest of the dataset to the variable x. Also, after checking for missing data, it was not found any missing values.

The table below presents a print screen of only the first four features' description of the dataset.



```
In [5]: xwithoutnormalisation.describe()
Out[5]:
```

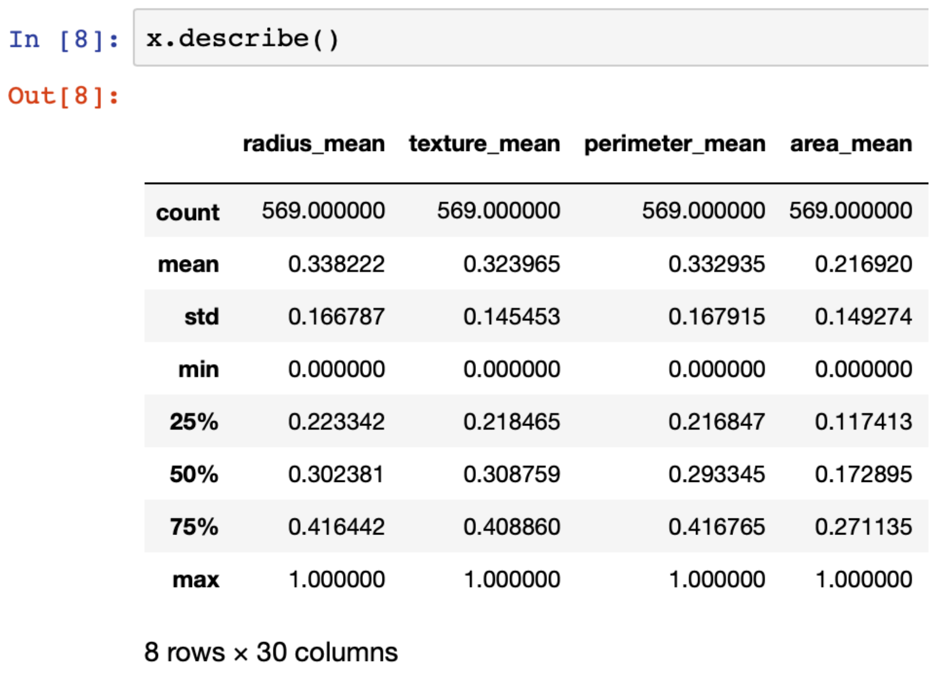
	radius_mean	texture_mean	perimeter_mean	area_mean
count	569.000000	569.000000	569.000000	569.000000
mean	14.127292	19.289649	91.969033	654.889104
std	3.524049	4.301036	24.298981	351.914129
min	6.981000	9.710000	43.790000	143.500000
25%	11.700000	16.170000	75.170000	420.300000
50%	13.370000	18.840000	86.240000	551.100000
75%	15.780000	21.800000	104.100000	782.700000
max	28.110000	39.280000	188.500000	2501.000000

8 rows × 30 columns

It can be clearly seen that there is a very large difference between the perimeter_mean feature's max value and the radius_mean feature's max value. Therefore, because there are some large differences between the scale in which each feature is represented, the dataset is normalised before creating any visualisation or classification on it, and the new features will be rescaled in order to have values from zero to one. The formula this paper uses to achieve this is the following:

$$x_{new} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

After the normalisation is made, the first four features from the datasets are described in the table below.



```
In [8]: x.describe()
```

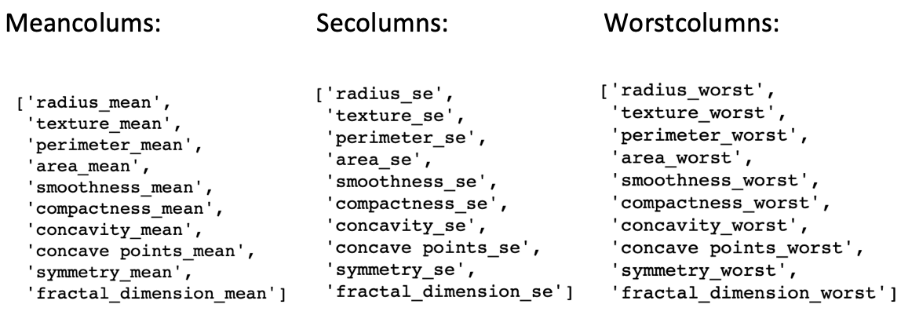
Out[8]:

	radius_mean	texture_mean	perimeter_mean	area_mean
count	569.000000	569.000000	569.000000	569.000000
mean	0.338222	0.323965	0.332935	0.216920
std	0.166787	0.145453	0.167915	0.149274
min	0.000000	0.000000	0.000000	0.000000
25%	0.223342	0.218465	0.216847	0.117413
50%	0.302381	0.308759	0.293345	0.172895
75%	0.416442	0.408860	0.416765	0.271135
max	1.000000	1.000000	1.000000	1.000000

8 rows × 30 columns

As it can be seen, all the new features range now from zero to one, which is better when trying to visualise and apply different classification algorithms to the dataset.

Furthermore, as each feature has three information (mean, standard error, and worst), three new variables are created for easier manipulation: "meancolumns", "secolumns", and "worstcolumns". The picture below shows what each new variable consists of.



Meancolums:

```
['radius_mean',
 'texture_mean',
 'perimeter_mean',
 'area_mean',
 'smoothness_mean',
 'compactness_mean',
 'concavity_mean',
 'concave points_mean',
 'symmetry_mean',
 'fractal_dimension_mean']
```

Secolumns:

```
['radius_se',
 'texture_se',
 'perimeter_se',
 'area_se',
 'smoothness_se',
 'compactness_se',
 'concavity_se',
 'concave points_se',
 'symmetry_se',
 'fractal_dimension_se']
```

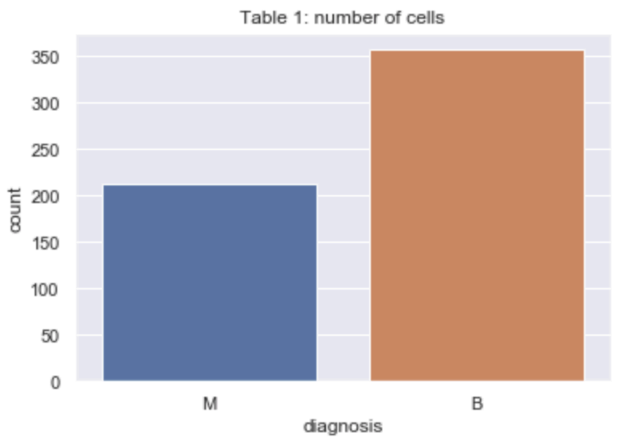
Worstcolumns:

```
['radius_worst',
 'texture_worst',
 'perimeter_worst',
 'area_worst',
 'smoothness_worst',
 'compactness_worst',
 'concavity_worst',
 'concave points_worst',
 'symmetry_worst',
 'fractal_dimension_worst']
```

Exploratory Analysis

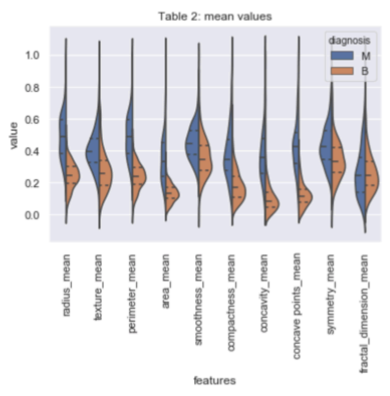
Before applying any classification algorithms, some exploration of the dataset is carried out. This involves different visualisations techniques which will help in gaining some insights and understanding the relationship between the features.

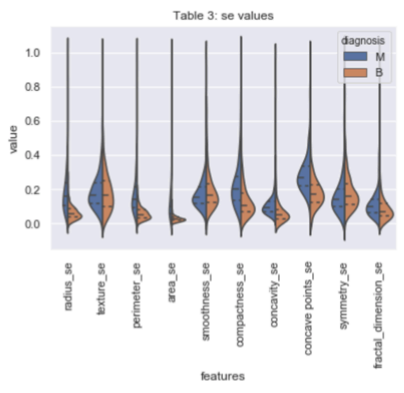
```
Benign cells:  357
Malignant cells:  212
```

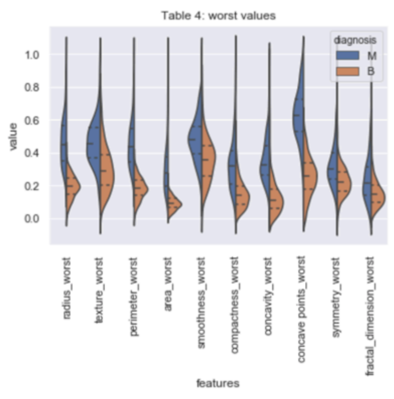


As it can be seen from table 1, the total number of cells is split into 212 malignant cells and 357 benign cells. So, there are less cancerous cells than non-cancerous.

Tables 2, 3, and 4 present different violin plots in order to better understand the relationship between the variables in the dataset. Those are split into 3 groups: mean values, standard error values, and worst values.

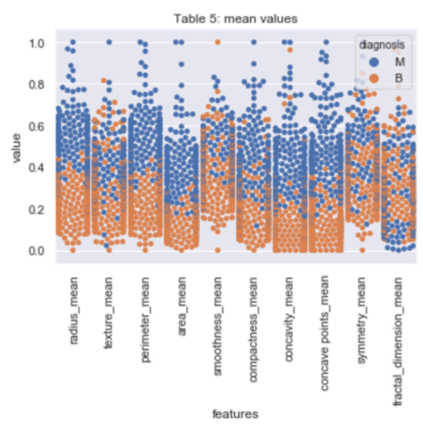


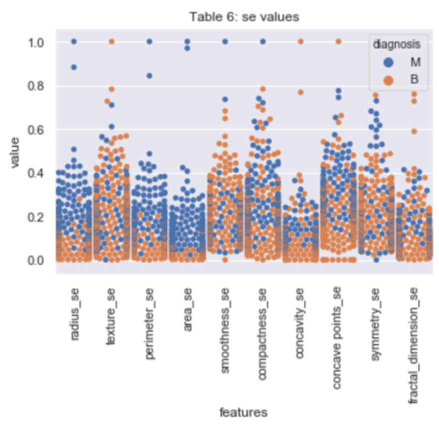


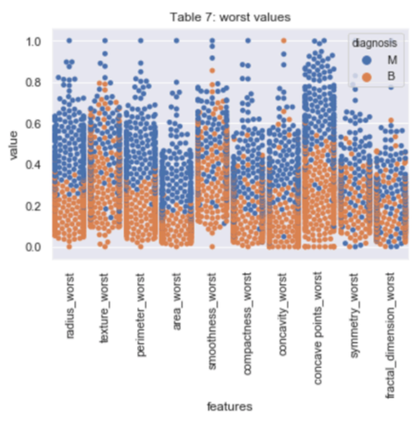


In the first group, it can be seen that for radius_mean, the median of the Malignant and Benign cells is separated, with a median of 0.5 on the left side and 0.25 on the right side. As there is a big difference in the median, it means that the radius_mean feature can play an important role in classifying the type of the cell. For example, if a new cell will have a radius_mean of 0.6, it is a high probability that the cell is Malignant. Applying the same rule, if a new analysed cell has a radius_mean of 0.2, there is a high probability of a Benign cell. Also, a high difference between the medians have the following features: texture_mean, perimeter_mean, area_mean, compactness_mean, concavity_mean, and concave points_mean. So all of those features play an important role in the classification of a cell. But there are also some features which have approximately the same median for both Malignant and Benign types. For example, fractal_dimension_mean, which has the same median of approximately 0.25, and so this feature is not that useful when one wants to classify the type of cell. Other features that don't have a significant difference in the median are smoothnes_mean, and symmetry_mean. In the standard error group, radius_se, perimeter_se, area_se, compactness_se, and concave points_se have separated medians, so they play an important role in classification, while the other 5 features don't have separated medians. Moving to the last group, worst values, almost all the features look like they have separated medians, with the exception of the symmetry_worst, and the fractal_dimension_worst features, which do not have a significant separation of the median so they do not provide good information for classification. As it can be seen, the violin plots are very useful tools in helping to decide which features are the most important when one tries to classify the type of a new cell.

Another technique for the dataset visualisation is the swarm plot, presented in tables 5, 6, and 7. In the three tables below, the variance can be seen with more clarity comparing to the violin plots and also it is much easier to have the feeling of which feature is useful for classification.







In the mean values group (Table 5), it can be seen that for the radius_mean, the Malignant and Benign cells are mostly separated so it can be stated that this feature can be used for the classification. The same as radius_mean are also the perimeter_mean, area_mean, concavity_mean, and concave points_mean. On the other hand, for the texture_mean, smoothness_mean, compactness_mean, symmetry_mean and fractal_dimension_mean it can be seen that there is not such a clear separation between the two types of cells, so it is hard to classify while using this feature. In the standard error groups (Table 6) it looks like only the radius_se, perimeter_se, and area_se are mostly separated, while all the other features have mixed types and so do not provide a clear separation. In the last groups, worst values (Table 7), the radius_worst, perimeter_worst, area_worst, concave points_worst present separation between the two types of cells while the others do not present a clear separation.

Bellow there are represented 3 correlation matrixes using python's heatmaps in order to understand the correlation between the variables. Again, this is made in 3 groups: standard error values in table 8, mean values in table 9, and the worst values in table 10.

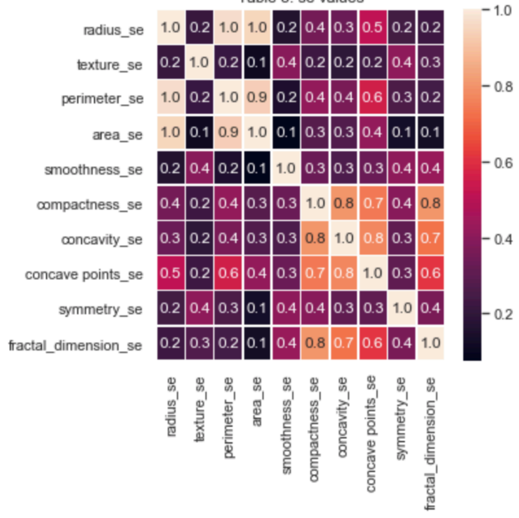


Table 8: se values

	radius_se	texture_se	perimeter_se	area_se	smoothness_se	compactness_se	concavity_se	concave points_se	symmetry_se	fractal_dimension_se
radius_se	1.0	0.2	1.0	1.0	0.2	0.4	0.3	0.5	0.2	0.2
texture_se	0.2	1.0	0.2	0.1	0.4	0.2	0.2	0.2	0.4	0.3
perimeter_se	1.0	0.2	1.0	0.9	0.2	0.4	0.4	0.6	0.3	0.2
area_se	1.0	0.1	0.9	1.0	0.1	0.3	0.3	0.4	0.1	0.1
smoothness_se	0.2	0.4	0.2	0.1	1.0	0.3	0.3	0.3	0.4	0.4
compactness_se	0.4	0.2	0.4	0.3	0.3	1.0	0.8	0.7	0.4	0.8
concavity_se	0.3	0.2	0.4	0.3	0.3	0.8	1.0	0.8	0.3	0.7
concave points_se	0.5	0.2	0.6	0.4	0.3	0.7	0.8	1.0	0.3	0.6
symmetry_se	0.2	0.4	0.3	0.1	0.4	0.4	0.3	0.3	1.0	0.4
fractal_dimension_se	0.2	0.3	0.2	0.1	0.4	0.8	0.7	0.6	0.4	1.0

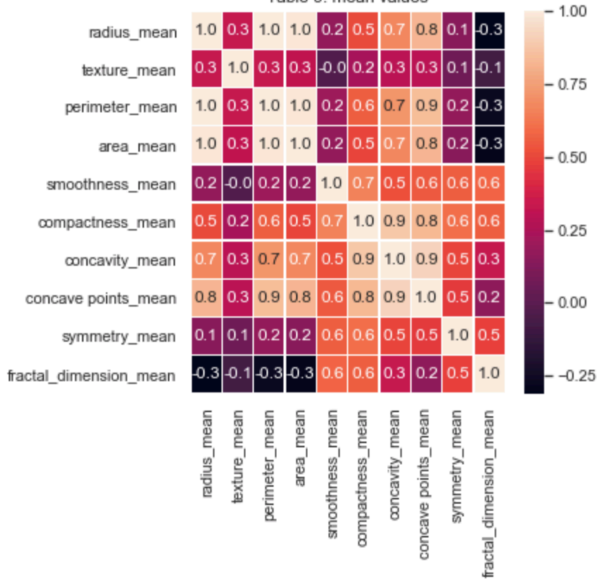


Table 9: mean values

	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	symmetry_mean	fractal_dimension_mean
radius_mean	1.0	0.3	1.0	1.0	0.2	0.5	0.7	0.8	0.1	-0.3
texture_mean	0.3	1.0	0.3	0.3	-0.0	0.2	0.3	0.3	0.1	-0.1
perimeter_mean	1.0	0.3	1.0	1.0	0.2	0.6	0.7	0.9	0.2	-0.3
area_mean	1.0	0.3	1.0	1.0	0.2	0.5	0.7	0.8	0.2	-0.3
smoothness_mean	0.2	-0.0	0.2	0.2	1.0	0.7	0.5	0.6	0.6	0.6
compactness_mean	0.5	0.2	0.6	0.5	0.7	1.0	0.9	0.8	0.6	0.6
concavity_mean	0.7	0.3	0.7	0.7	0.5	0.9	1.0	0.9	0.5	0.3
concave points_mean	0.8	0.3	0.9	0.8	0.6	0.8	0.9	1.0	0.5	0.2
symmetry_mean	0.1	0.1	0.2	0.2	0.6	0.6	0.5	0.5	1.0	0.5
fractal_dimension_mean	-0.3	-0.1	-0.3	-0.3	0.6	0.6	0.3	0.2	0.5	1.0

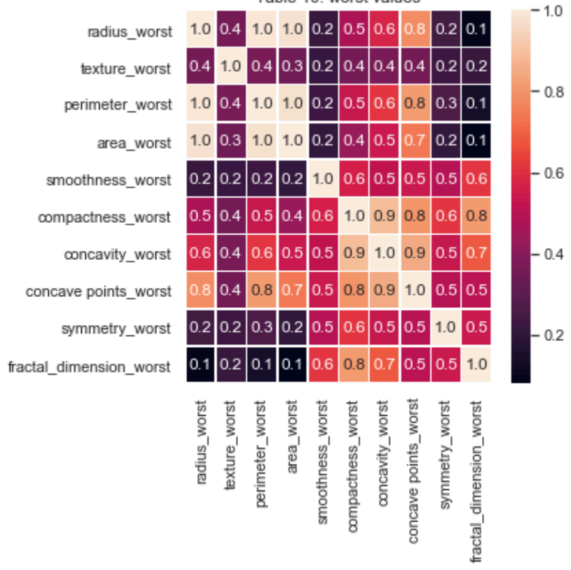


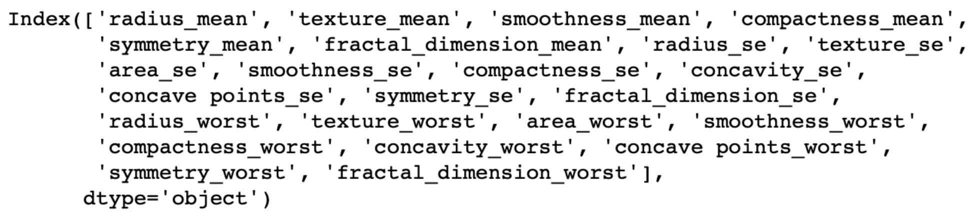
Table 10: worst values

	radius_worst	texture_worst	perimeter_worst	area_worst	smoothness_worst	compactness_worst	concavity_worst	concave points_worst	symmetry_worst	fractal_dimension_worst
radius_worst	1.0	0.4	1.0	1.0	0.2	0.5	0.6	0.8	0.2	0.1
texture_worst	0.4	1.0	0.4	0.3	0.2	0.4	0.4	0.4	0.2	0.2
perimeter_worst	1.0	0.4	1.0	1.0	0.2	0.5	0.6	0.8	0.3	0.1
area_worst	1.0	0.3	1.0	1.0	0.2	0.4	0.5	0.7	0.2	0.1
smoothness_worst	0.2	0.2	0.2	0.2	1.0	0.6	0.5	0.5	0.5	0.6
compactness_worst	0.5	0.4	0.5	0.4	0.6	1.0	0.9	0.8	0.6	0.8
concavity_worst	0.6	0.4	0.6	0.5	0.5	0.9	1.0	0.9	0.5	0.7
concave points_worst	0.8	0.4	0.8	0.7	0.5	0.8	0.9	1.0	0.5	0.5
symmetry_worst	0.2	0.2	0.3	0.2	0.5	0.6	0.5	0.5	1.0	0.5
fractal_dimension_worst	0.1	0.2	0.1	0.1	0.6	0.8	0.7	0.5	0.5	1.0

As some of the classification algorithms can suffer from correlation biases [15], this paper chose only the uncorrelated features as an input for the algorithms. As it can be seen from the 3 correlation matrixes tables, the radius, perimeter and area features are highly correlated with a correlation of 1. So there is no need to use all of them in the classification problem, and so later only one of them will be selected. Other features that are highly correlated are compactness_mean, concavity_mean and concavepoint_mean.

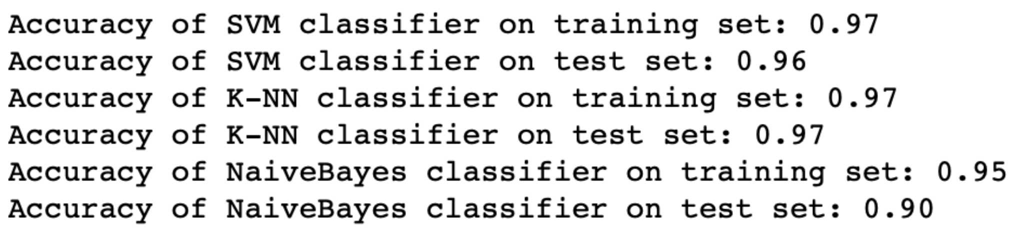
Classification results

Having in mind the computational costs, the time constraints for this study, and also the fact that some classification algorithms can suffer from correlation biases [14], not all the features are used as an input to the models, and so this paper only uses the uncorrelated features. After the features that are highly correlated have been extracted, the new dataset consists of only 24 features which can be seen in the picture below.



```
Index(['radius_mean', 'texture_mean', 'smoothness_mean', 'compactness_mean',
       'symmetry_mean', 'fractal_dimension_mean', 'radius_se', 'texture_se',
       'area_se', 'smoothness_se', 'compactness_se', 'concavity_se',
       'concave points_se', 'symmetry_se', 'fractal_dimension_se',
       'radius_worst', 'texture_worst', 'area_worst', 'smoothness_worst',
       'compactness_worst', 'concavity_worst', 'concave points_worst',
       'symmetry_worst', 'fractal_dimension_worst'],
      dtype='object')
```

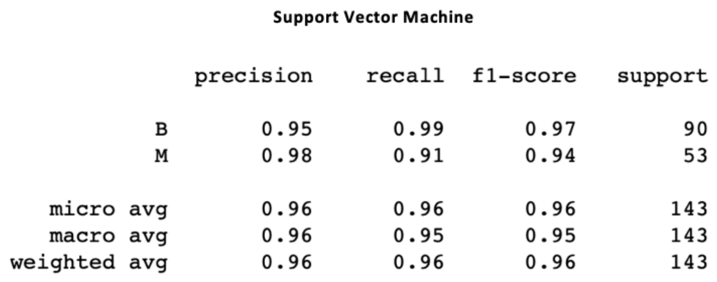
Of course, because the dataset is labelled, only supervised machine learning models are being used: Support Vector Machines, Naïve Bayes, and the K-nearest neighbour algorithm.



```
Accuracy of SVM classifier on training set: 0.97
Accuracy of SVM classifier on test set: 0.96
Accuracy of K-NN classifier on training set: 0.97
Accuracy of K-NN classifier on test set: 0.97
Accuracy of NaiveBayes classifier on training set: 0.95
Accuracy of NaiveBayes classifier on test set: 0.90
```

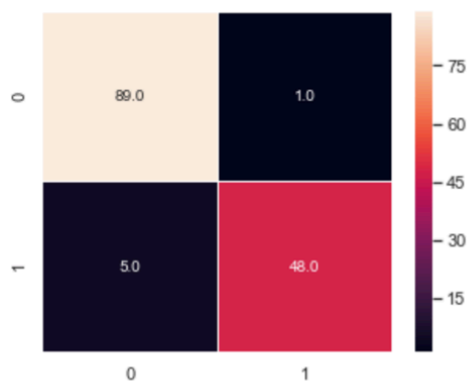
In the picture above it can be seen that all the three chosen classifiers presents high accuracy. It looks like the best one is the K-nearest neighbour algorithm, which has a 0.97 accuracy for both the training and the testing set.

Now, to have a better idea about the algorithms, a more in-depth analysis is being made.



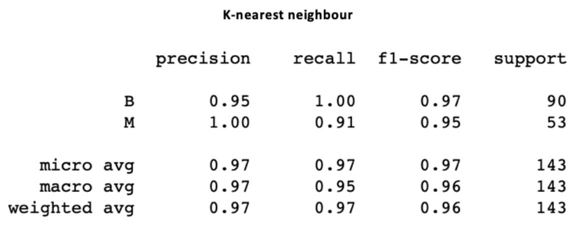
Support Vector Machine

```
              precision    recall  f1-score   support

           B       0.95      0.99      0.97        90
           M       0.98      0.91      0.94        53

   micro avg       0.96      0.96      0.96       143
   macro avg       0.96      0.95      0.95       143
weighted avg       0.96      0.96      0.96       143
```



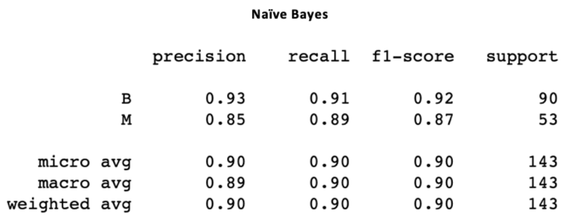
The picture above shows the classification report and the confusion matrix for the Support Vector Machine classifier. As it can be seen, there are 89 True Positives (TP), 48 True Negatives (TN), 5 False Positives (FP), and 1 False Negative (FN). The average precision, being the ratio of correctly predicted positive observations to the total predicted positive observations, is 0.96. The question that this metric answer is of all the cells that are labelled as Benign/ Malignant, how many actually are Benign/ Malignant. The average recall is 0.96, which is the ratio of correctly predicted positive observations to all the observations in actual class. The question the recall answers is of all the cells that actually are Benign/Malignant, how many did the algorithms labelled correctly. So, it can be seen that all the metrics show some good results, meaning that this algorithm worked well on this specific dataset.

Following the logic around the explanation of the first Algorithm, the pictures below show the results of the K-Nearest Neighbour and Naïve Bayes. It can be seen that all the algorithms showed some good metrics.



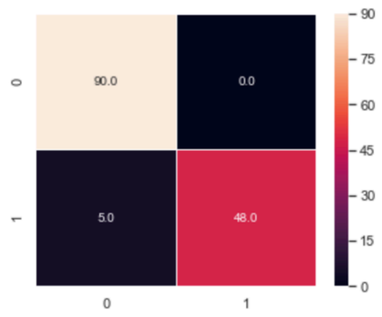
K-nearest neighbour

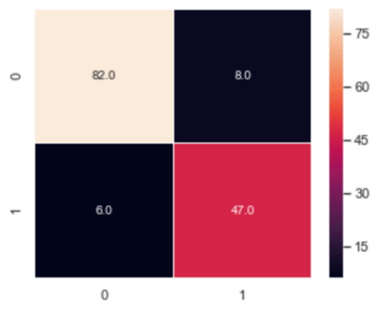
```
              precision    recall  f1-score   support

           B       0.95      1.00      0.97        90
           M       1.00      0.91      0.95        53

   micro avg       0.97      0.97      0.97       143
   macro avg       0.97      0.95      0.96       143
weighted avg       0.97      0.97      0.96       143
```



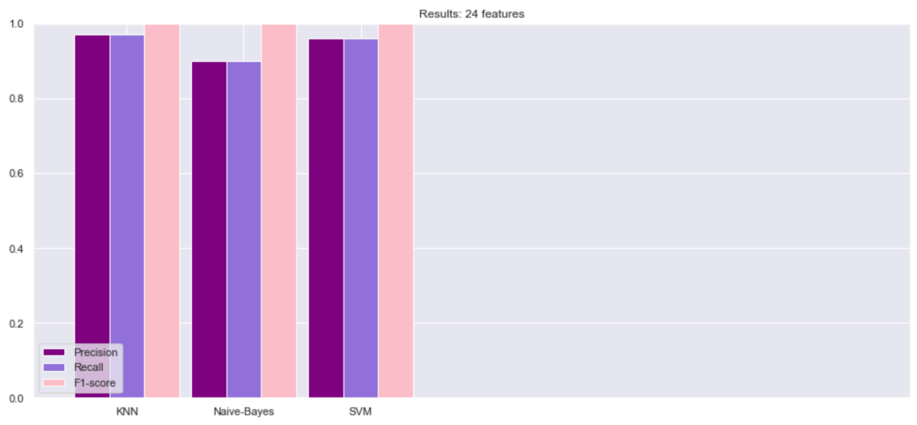
Naïve Bayes

```
              precision    recall  f1-score   support

           B       0.93      0.91      0.92        90
           M       0.85      0.89      0.87        53

   micro avg       0.90      0.90      0.90       143
   macro avg       0.89      0.90      0.90       143
weighted avg       0.90      0.90      0.90       143
```





For a better visualisation of the results, the table below presents the precision, recall and f1-score for all the three algorithms. And although all the algorithms returned similar results, it can be argued that the best one for this specific dataset is the KNN, as it had the biggest values for all the three measures. As it can be seen, the worst is the Naïve Bayes, with an average precision and recall of 0.90.



Conclusion

This paper implemented three classification algorithms (Support Vector Machines, K-Nearest Neighbour, and Naïve Bayes) for the Wisconsin Breast Cancer dataset [6] in order to see which one is the best in predicting the class of a new cell after visualising the dataset and explaining features correlation and how well can some of them explain the class of a cell. As Breast cancer is ranked first and second out of all diseases reported in women, there is a high necessity for further studies on how machine learning and data analytics can help doctors in decision making. In the first part of this paper, three visualisation tools have been used. First, 30 violin plots have been presented grouped in pairs of 10: mean values, standard error values, and worst values. By looking at those violin plots, it can be observed that some features have a separated median for the Benign and Malignant classes, and so those can explain the class from which the specific cell is. The ones who do not present a clear separation of the median for the two classes cannot help in classifying the cell. Another visualisation method used in this paper was the swarm plot. Comparing to the violin plots, the swarm plots express more clearly the separation between the two classes. As it has already been presented, there are some features who are clearly separated, like the radius_mean, and perimeter_mean, while for others there is not a clear separation between the classes. Lastly, the correlation matrix has been used in order to check for highly correlated features. Because of computational costs and time constraints, but also because of correlation biases [14], the highly correlated features have been removed, and only 24 features have been used as input to the three algorithms. The results found are similar, with the best accuracy for the KNN, with 0.97 for both the training and test sets. The last algorithm in terms of accuracy was the Naïve Bayes, with 0.95 accuracy on the training test, and 0.90 accuracy on the testing set. The biggest limitation for this study is the time constraint, and further research is needed in the implementation of other classification algorithms on larger datasets in order to have more precise findings of how machine learning can help in classifying cells into Benign or Malignant

References

1. Cancer Statistics (Mac 2017)
   https://www.cancer.gov/about-cancer/understanding/statistics

2. Gouda I Salama, M Abdelhalim, and Magdy Abd-Elghany Zeid, (2012) “Breast cancer diagnosis on three different datasets using multi-classifiers”, Breast Cancer (WDBC) 32, 569 (2012), 2.

3. Elias Zafiropoulous, Ilias Maglogiannis, and Ioannis Anagnostopului, (2016) “A support vector machine approach to breast cancer diagnosis and prognosis”, Artificial Intelligence Applications and Innovations, 500-507

4. William H Wolberg, W Nick Street, and Olvi L Mangasarian, (2009) “A deep learning approach for cancer detection and relevant gene identification”, The US National Library of Medicine, National Institute of health

5. Oneeb Rehman, Hanqi Qhuang, (2019) “Validation of miRNAs as Breast Cancer Biomarkers with a Machine Learning approach”, The Journal of Medicine, pp(200-240)

6. William H Wolberg, W Nick Street, and Olvi L Mangasrian, (1992) “Breast cancer Wisconsin (diagnostic) data set”
   https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Diagnostic)

7. Geeta Patel, (2013) “The psychosocial impact of breast cancer diagnosis and treatment in black and south Asian women”, Faculty of Health and Life Sciences, University of West of England, Bristol

8. Numpy.org. (2017) Available at: http://www.numpy.org/ [Accessed 2 Mar 2019].

9. Wes McKinney. “pandas: a foundational python library for data Analysis and Statistics”

10. Scikit-learn.org. (2017). scikit-learn: machine learning in Python — scikit-learn 0.18.1 documentation. [online] Available at: http://scikit-learn.org/stable/ [Accessed 4 Mar. 2019].
11. F. Pedregosa et al. “Scikit-learn: Machine Learning in Python”. In: Journal of Machine Learning Research 12 (2011), pp. 2825–2830.

12. “Violin Plot”, NIST DataPlot, National Institute of Standards and technology, 2015-10-13

13. Pryanka Sonkar, (2017) “Application of supervised machine learning to predict the mortality risk in elderly using biomarkers”, Dublin Institute of Technology

14. Abu Mostafa, Yaser S, (2010) “Data complexity in machine learning and novel classification algorithms”, California Institute of Technology

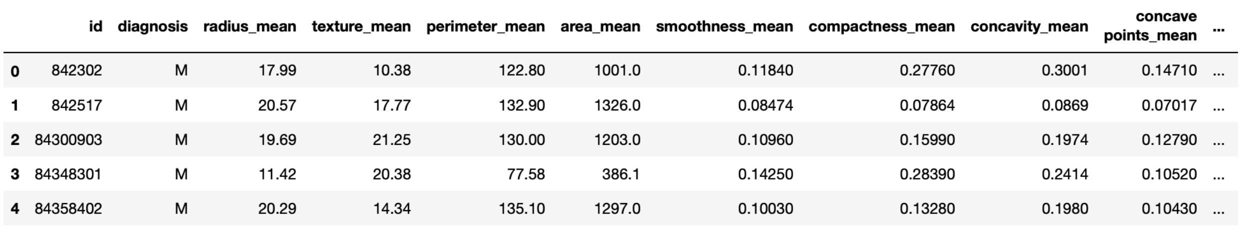
15. Michael David Rechenthin, (2014) “Machine learning classification techniques for the analysis of cancerous cells”, PhD (Doctor of Philosophy) thesis, University of Iowa

16. OpenCV, “Introduction to Support Vector Machines”, Accessed on March 2, 2019 http://docs.opencv.org/doc/tutorials/ml/introduction_to_svm/introduction_to_svm. html

17. K. P. Bennett and E. J. Bredensteiner, “Duality and geometry in SVM classifiers”, the 17th International Conference on Machine Learning, pages 57-64. Morgan Kaufmann, San Francisco, CA, 2000

Appendices

Dataset sample



	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	...
0	842302	M	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	...
1	842517	M	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	...
2	84300903	M	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	...
3	84348301	M	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	...
4	84358402	M	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	...

Python Code for the visualisation and algorithms implementation

```
#!/usr/bin/env python
# coding: utf-8
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import time
from sklearn.model_selection import train_test_split
sns.set(style="darkgrid")

data = pd.read_csv('breastcancer.csv')

data.info()

print(data.columns)

y = data.diagnosis #y represents the labels
listofcolumnsnotused = ['Unnamed: 32','id','diagnosis']
xwithoutnormalisation = data.drop(listofcolumnsnotused, axis = 1) #x
is the dataset without 3 columns

xwithoutnormalisation.describe()

#normalisation
```

```
x = (xwithoutnormalisation -
np.min(xwithoutnormalisation))/(np.max(xwithoutnormalisation)-
np.min(xwithoutnormalisation)).values
x.describe()

listmean = ['radius_mean', 'texture_mean', 'perimeter_mean',
       'area_mean', 'smoothness_mean', 'compactness_mean',
'concavity_mean',
       'concave points_mean', 'symmetry_mean',
'fractal_dimension_mean']
listse = ['radius_se', 'texture_se', 'perimeter_se', 'area_se',
'smoothness_se',
       'compactness_se', 'concavity_se', 'concave points_se',
'symmetry_se',
       'fractal_dimension_se']
listworst = ['radius_worst', 'texture_worst',
       'perimeter_worst', 'area_worst', 'smoothness_worst',
       'compactness_worst', 'concavity_worst', 'concave points_worst',
       'symmetry_worst', 'fractal_dimension_worst']
meancolumns = x.drop(listse, axis = 1).drop(listworst, axis = 1)
secolumns = x.drop(listmean, axis = 1).drop(listworst, axis = 1)
worstcolumns = x.drop(listse, axis=1).drop(listmean, axis =1)

listworst

ax = sns.countplot(y).set_title("Table 1: number of cells")

B, M = y.value_counts()
print("Benign cells: ", B)
print("Malignant cells: ", M)

#violin plots for 3 groups of 10 features

data_violin = y
data = pd.concat([y, x.iloc[:,0:10]], axis = 1)
data = pd.melt(data, id_vars = "diagnosis", var_name="features",
value_name="value")
```

```
plt.figure()
plt.xticks(rotation=90)
sns.violinplot(x="features", y="value", hue="diagnosis",
data=data,split=True, inner="quart", rotation =90).set_title("Table 2:
mean values")

data_violin = y
data = pd.concat([y, x.iloc[:,10:20]], axis = 1)
data = pd.melt(data, id_vars = "diagnosis", var_name="features",
value_name="value")
plt.figure()
plt.xticks(rotation=90)
sns.violinplot(x="features", y="value", hue="diagnosis",
data=data,split=True, inner="quart", rotation =90).set_title("Table 3:
se values")

data_violin = y
data = pd.concat([y, x.iloc[:,20:31]], axis = 1)
data = pd.melt(data, id_vars = "diagnosis", var_name="features",
value_name="value")
plt.figure()
plt.xticks(rotation=90)
sns.violinplot(x="features", y="value", hue="diagnosis",
data=data,split=True, inner="quart", rotation =90).set_title("Table 4:
worst values")

datay=y
data = pd.concat([y,x.iloc[:,0:10]],axis=1)
data = pd.melt(data,id_vars="diagnosis",
               var_name="features",
               value_name='value')
plt.xticks(rotation=90)
sns.swarmplot(x="features", y="value", hue="diagnosis",
data=data).set_title("Table 5: mean values")

# In[21]:
```

```
data = pd.concat([y,x.iloc[:,10:20]],axis=1)
data = pd.melt(data,id_vars="diagnosis",
               var_name="features",
               value_name='value')
plt.xticks(rotation=90)
sns.swarmplot(x="features", y="value", hue="diagnosis",
data=data).set_title("Table 6: se values")

# In[22]:

data = pd.concat([y,x.iloc[:,20:31]],axis=1)
data = pd.melt(data,id_vars="diagnosis",
               var_name="features",
               value_name='value')
plt.xticks(rotation=90)
sns.swarmplot(x="features", y="value", hue="diagnosis",
data=data).set_title("Table 7: worst values")

# In[24]:

f,ax = plt.subplots(figsize=(18, 18))
sns.heatmap(x.corr(), annot=True, linewidths=.5, fmt= '.1f',ax=ax)

# In[25]:

f,ax = plt.subplots(figsize=(5, 5))
sns.heatmap(secolumns.corr(), annot=True, linewidths=.5, fmt=
'.1f',ax=ax).set_title("Table 8: se values")

# In[26]:

f,ax = plt.subplots(figsize=(5, 5))
sns.heatmap(meancolumns.corr(), annot=True, linewidths=.5, fmt=
'.1f',ax=ax).set_title("Table 9: mean values")

f,ax = plt.subplots(figsize=(5, 5))
```

```
sns.heatmap(worstcolumns.corr(), annot=True, linewidths=.5, fmt=
'.1f',ax=ax).set_title("Table 10: worst values")

listtodrop = ['perimeter_mean', 'perimeter_se', 'perimeter_worst',
            'area_mean', 'perimeter_mean', 'perimeter_worst',
            'concavity_mean', 'concave points_mean'
      ]

xml = x.drop(listtodrop, axis = 1)


xml.describe()

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(xml, y,
random_state=0)

from sklearn.linear_model import LogisticRegression
from sklearn import metrics

# Create the model Support Vector Machine
from sklearn.svm import SVC
model_svm= SVC(kernel='linear', C = 1.0)
model_svm.fit(X=X_train,y=y_train)
predicion_svm = model_svm.predict(X_test)

#Clasification report
results_svm=metrics.classification_report(y_true=y_test,
y_pred=predicion_svm)
print(results_svm)

#Confusion matrix
cm_svm=metrics.confusion_matrix(y_true=y_test,
y_pred=predicion_svm)
f,ax = plt.subplots(figsize=(5, 4))
sns.heatmap(cm_svm, annot=True, linewidths=.5, fmt= '.1f',ax=ax);
```

```
from sklearn.neighbors import KNeighborsClassifier
# Create the model
model_knn= KNeighborsClassifier(n_neighbors=10)
model_knn.fit(X=X_train,y=y_train)
predicion_knn = model_knn.predict(X_test)
# Clasification report
results_knn=metrics.classification_report(y_true=y_test,
y_pred=predicion_knn)
print(results_knn)

# Confusion matrix
cm_knn= metrics.confusion_matrix(y_true=y_test,
y_pred=predicion_knn)
f,ax = plt.subplots(figsize=(5, 4))
sns.heatmap(cm_knn, annot=True, linewidths=.5, fmt= '.1f',ax=ax);


# In[76]:


def resultados_classification_report(cr):
    total=[]
    lines = cr.split('\n')
    total_aux=lines[5].split()
    for i in range(3,6):
        total.append(float(total_aux[i]))
    return total


from sklearn.naive_bayes import GaussianNB
model_NB= GaussianNB()
model_NB.fit(X=X_train,y=y_train)
predicion_NB = model_NB.predict(X_test)


# In[77]:
```

```
from sklearn import metrics
#Clasification report
results_NB=metrics.classification_report(y_true=y_test,
y_pred=predicion_NB)
print(results_NB)

#Confusion Matrix
cm_NB=metrics.confusion_matrix(y_true=y_test,
y_pred=predicion_NB)
f,ax = plt.subplots(figsize=(5, 4))
sns.heatmap(cm_NB, annot=True, linewidths=.5, fmt= '.1f',ax=ax);


# In[89]:


print('Accuracy of SVM classifier on training set: {:.2f}'
    .format(model_svm.score(X_train, y_train)))
print('Accuracy of SVM classifier on test set: {:.2f}'
    .format(model_svm.score(X_test, y_test)))
print('Accuracy of K-NN classifier on training set: {:.2f}'
    .format(model_knn.score(X_train, y_train)))
print('Accuracy of K-NN classifier on test set: {:.2f}'
    .format(model_knn.score(X_test, y_test)))
print('Accuracy of NaiveBayes classifier on training set: {:.2f}'
    .format(model_NB.score(X_train, y_train)))
print('Accuracy of NaiveBayes classifier on test set: {:.2f}'
    .format(model_NB.score(X_test, y_test)))


# In[91]:


names=['KNN','Naive-Bayes','SVM']
modelResults=[]
```

```
modelResults.append(results_knn)
modelResults.append(results_NB)
modelResults.append(results_svm)


totalResults=[]
totalPrecision=[]
totalRecall=[]
totalF=[]
for i in range(len(modelResults)):

totalResults.append(resultados_classification_report(modelResults[i])
)

for i in range(len(totalResults)):
    totalPrecision.append(totalResults[i][0])
    totalRecall.append(totalResults[i][1])
    totalF.append(totalResults[i][2])


# In[92]:


index = np.arange(7)
# Set position of bar on X axis
barWidth=0.3
r1 = np.arange(len(totalPrecision))
r2 = [x + barWidth for x in r1]
r3 = [x + barWidth for x in r2]

plt.subplots(figsize=(16, 7))
plt.bar(r1,totalPrecision,width=barWidth, label='Precision',
color="purple")
plt.bar(r2,totalRecall,width=barWidth, label='Recall',
color="mediumpurple")
plt.bar(r3,totalF,width=barWidth,  label='F1-score', color="pink")
plt.axis([-0.5,7,0, 1])
plt.legend(loc='lower left')
plt.title('Results: 30 features')

# Add xticks on the middle of the group bars
plt.xticks([r + barWidth for r in range(len(totalPrecision))], names)

# In[105]:

plt.title('KNN')
sns.heatmap(cm_knn, annot=True, linewidths=.5, fmt= '.1f')

plt.title('Naive-Bayes')
sns.heatmap(cm_NB, annot=True, linewidths=.5, fmt= '.1f')

plt.title('SVM')
sns.heatmap(cm_svm, annot=True, linewidths=.5, fmt= '.1f')

print(xml.columns)
```