

University of Kragujevac
Faculty of Engineering Sciences



Basics of Machine and Deep Learning

Project: **Breast Cancer Prediction Using Machine Learning Models**

Student:
Саша Ристић 635/2018

Course Professor:
Др. Владимир Миловановић

Kragujevac, 2022.

Contents

| | |
|---|-----------|
| Breast Cancer Prediction Using Machine Learning Models | 2 |
| Project documentation by Staša Ristić | 2 |
| 1. Exploratory Data Analysis (EDA)..... | 3 |
| 2. Preparing the data | 4 |
| 3. Relationship analysis..... | 4 |
| 4. Models and performance analysis..... | 7 |
| a) Logistic Regression | 7 |
| b) Decision Tree | 8 |
| c) Random Forest..... | 9 |
| 5. Literature | 10 |

Breast Cancer Prediction Using Machine Learning Models

Project documentation by Staša Ristić

I am a student from the Faculty of Engineering Sciences in Kragujevac, Serbia, currently in my fourth, and final, year of studies on a Computer Science and Software Engineering module. This documentation is for a project that I have done for a course called “Basics of Machine and Deep Learning”.

I started creating the project on 6th of February 2022. using Jupyter Notebook and completed it, a week and a half later, on 16th of February 2022, by working on it for several hours a day. Without further ado, let me introduce you to the problem I have been presented with.

Dataset that I’ve chosen for this project is called "Breast Cancer Wisconsin (Diagnostic) Data Set". It includes features that have been computed from a digitized image of a fine needle aspirate (FNA) of a breast mass. Those features describe certain characteristics of the cell nuclei present in the image.

It contains two attribute features:

1. ID number of the patient
2. Diagnosis, where M stands for Malignant, and B stands for Benign

It also contains ten real-valued features that have been computed for each cell nucleus:

- a) Radius (mean of distances from center to points on the perimeter)
- b) Texture (standard deviation of grey-scale values)
- c) Perimeter
- d) Area
- e) Smoothness (local variation in radius lengths)
- f) Compactness ($\text{perimeter}^2 / \text{area} - 1.0$)
- 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” or largest (mean of the three largest values) of these features were computed for each image resulting in 30 features.

On the first glance, it is apparent that this data set has input (**features**) that are already labeled, and output (**target**) – diagnosis, that indicates that the model that should be used falls into the group of Supervised Learning models. I have opt for Logistic Regression, Decision Tree Classifier and Random Forest Classifier.

1. Exploratory Data Analysis (EDA)

Exploratory data analysis or EDA is an approach of analyzing data sets to summarize their main characteristics, often using statistical graphics and other data visualization methods. Primarily EDA is for seeing what the data can tell us beyond the formal modeling or hypothesis testing task. Exploratory data analysis has been promoted by John Tukey since 1970 to encourage statisticians to explore the data and possibly formulate hypotheses that could lead to new data collection and experiments. It also helps us to filter data from redundancies and remove faulty data.

Before the analysis, I imported necessary libraries such as: pandas, seaborn, matplotlib, numpy and sklearn. Using pandas library, the data has been loaded from the csv file. First I took a look at the features and in which way they were formatted, as shown on the picture 1.1.

```
[3]: data = pd.read_csv("dataBC.csv", sep=",")
data.head()
```

```
[3]:
```

| | 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 | ... |

5 rows × 32 columns

Picture 1.1. Data formatted in a table

Next, by using describe function we check the basic computed summary of statistics pertaining to the DataFrame columns. This function gives mean, standard deviation and IQR values.

```
[17]: data.describe()
```

```
[17]:
```

| | diagnosis | radius_mean | texture_mean | perimeter_mean | area_mean | smoothness_mean | compactness_mean | concavity_mean | concave points_mean | symmetry_mean | ... |
|-------|------------|-------------|--------------|----------------|-------------|-----------------|------------------|----------------|---------------------|---------------|-----|
| count | 569.000000 | 569.000000 | 569.000000 | 569.000000 | 569.000000 | 569.000000 | 569.000000 | 569.000000 | 569.000000 | 569.000000 | ... |
| mean | 0.372583 | 14.127292 | 19.289649 | 91.969033 | 654.889104 | 0.096360 | 0.104341 | 0.088799 | 0.048919 | 0.181162 | ... |
| std | 0.483918 | 3.524049 | 4.301036 | 24.298981 | 351.914129 | 0.014064 | 0.052813 | 0.079720 | 0.038803 | 0.027414 | ... |
| min | 0.000000 | 6.981000 | 9.710000 | 43.790000 | 143.500000 | 0.052630 | 0.019380 | 0.000000 | 0.000000 | 0.106000 | ... |
| 25% | 0.000000 | 11.700000 | 16.170000 | 75.170000 | 420.300000 | 0.086370 | 0.064920 | 0.029560 | 0.020310 | 0.161900 | ... |
| 50% | 0.000000 | 13.370000 | 18.840000 | 86.240000 | 551.100000 | 0.095870 | 0.092630 | 0.061540 | 0.033500 | 0.179200 | ... |
| 75% | 1.000000 | 15.780000 | 21.800000 | 104.100000 | 782.700000 | 0.105300 | 0.130400 | 0.130700 | 0.074000 | 0.195700 | ... |
| max | 1.000000 | 28.110000 | 39.280000 | 188.500000 | 2501.000000 | 0.163400 | 0.345400 | 0.426800 | 0.201200 | 0.304000 | ... |

8 rows × 31 columns

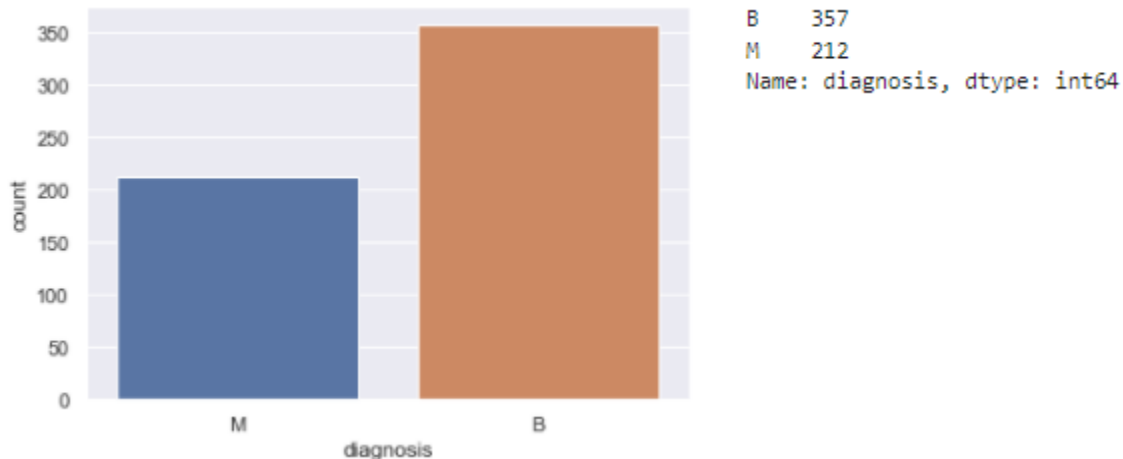
Picture 1.2. Computed summary of statistics

Shape of the data set is 569 instances (rows) and 32 columns. Using isnull().sum() functions to check if there are any holes in the data where there is values, or just null values showed that there is no such data. That means there is no need to fill possibly empty instances.

I have also checked to see how balanced the data is, and noticed that the ratio is around 60-40, which means that there is no need to further balance the data.

2. Preparing the data

Before preparing the data, I mapped out a graph of outputs. There are 357 instances in which diagnosis came out as benign, and 212 which came out as malignant.



Picture 2.1. Relation between malignant and benign tumors

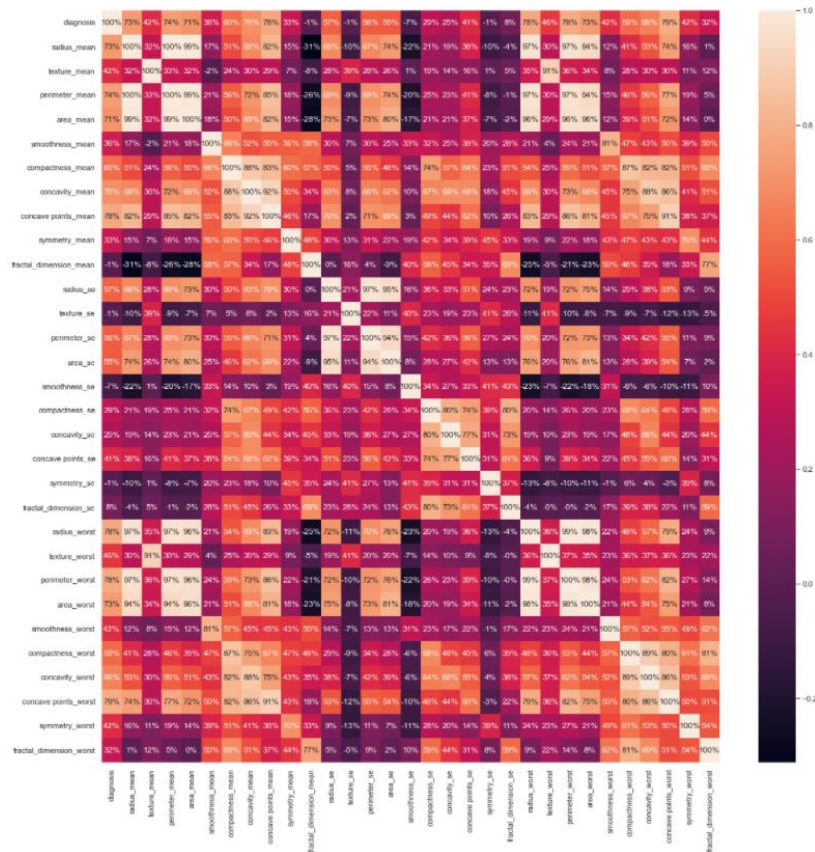
First step is to clean the data from information that has no relation to the outcome of tumor being benign or malignant. A column that should be excluded is the patient id, as the number under which a person is registered in a study or in a hospital should have no impact as to whether the tumor, that the said patient has, is benign or malignant.

Next step is mapping the diagnosis into a numeric form, as diagnosis column wasn't previously numeric it has to be converted into a numeric form so that the algorithm can understand the data. A numeric value that will be used as representation for a malignant tumor will be 1, and for benign it will be a 0.

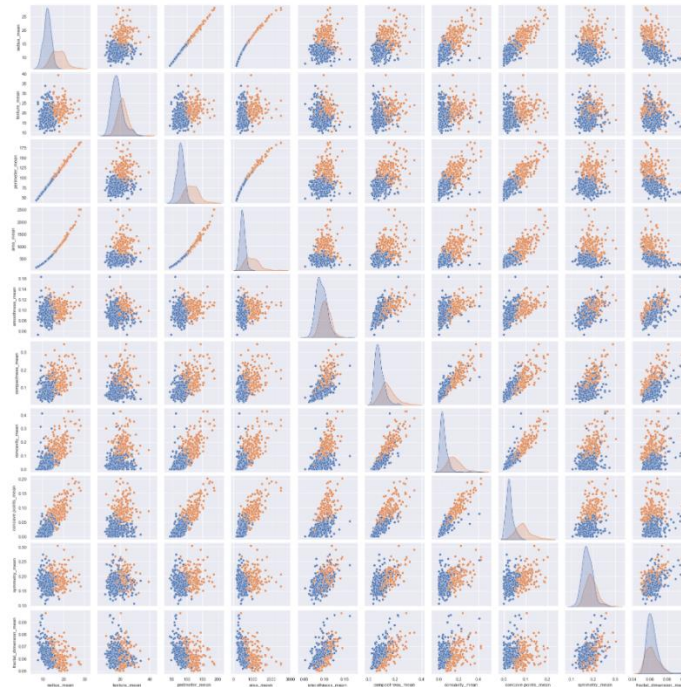
With function `.nunique()` I check for possible repeating values and notice that there is no or very little of them (not including the output). There is no need for removing duplicate instances.

3. Relationship analysis

By plotting a heatmap I can see that there is correlations between some features, so I decided to take a closer look by plotting pairplots of diagnosis feature and all mean features. Almost perfectly linear patterns between the radius, perimeter and area attributes are hinting at the presence of multicollinearity. Multicollinearity is the occurrence of high intercorrelations among two or more independent variables in a multiple regression model. In general, multicollinearity can lead to wider confidence intervals that can produce less reliable probabilities in terms of the effect of independent variables in a model. Another set of variables that can possibly imply multicollinearity are concavity, and concave_points and compactness.

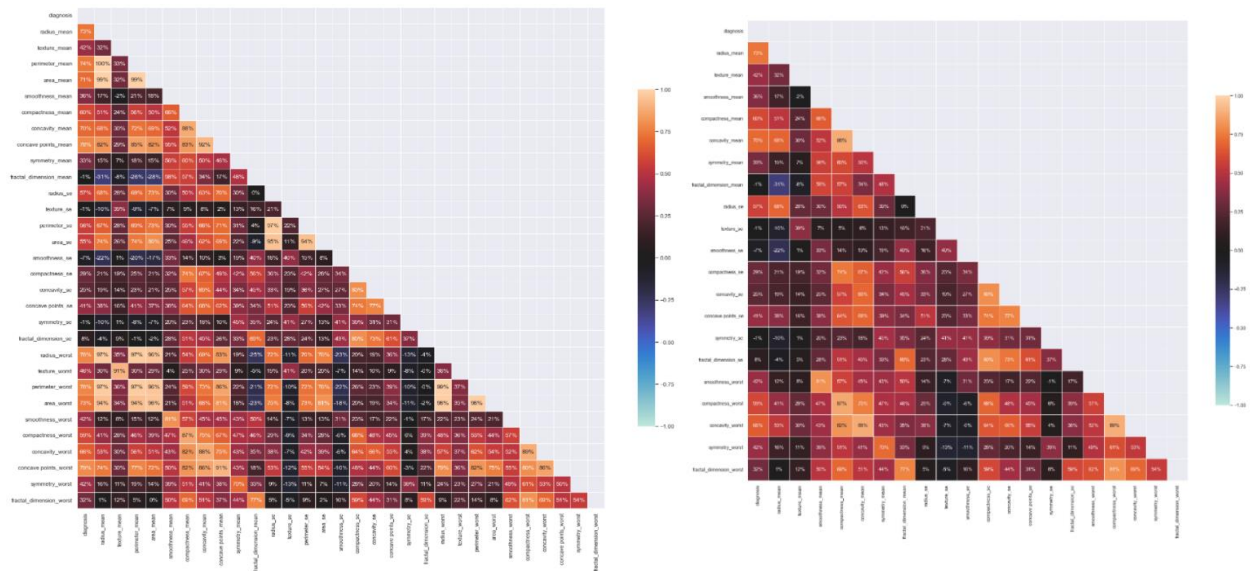


Picture 3.1 Correlation heatmap



Picture 3.2. Pairplots between features

Generating a correlation matrix I verified the presence of multicollinearity between some of the variables. For instance the radius_mean column has correlation of 1 and 0.99 with perimeter_mean and area_mean, respectively. This means that these three columns essentially contain the same information, which is physical size of the observed cell. I firstly tried removing all of the columns that may be showing high correlations by eye and that worsened later results of machine learning models, so I decided to create a for loop which catches all columns that have correlation higher than 0.9 (not less because there is not many features to begin with). Consequently, the columns caught were: perimeter_worst, concave points_worst, area_mean, perimeter_mean, concave points_mean, area_worst, texture_worst, perimeter_se, area_se and radius_worst, thereupon generating a new correlation matrix we can see the difference of the removed features.



Picture 3.4. Correlation matrix before and after removing correlated features

Once I've finished removing all the redundancies and fully cleaning the data, I've then split it into train and test groups, where train contains 70% of the original data and test 30%. Preprocessing is done when function `fit_transform()` is used on `X_train` and just transform on `X_test` data. That way `StandardScaler` standardizes features by using $\mu=0$ (mean), $\delta=1$ (standard deviation), which is learned from train data.

4. Models and performance analysis

a) Logistic Regression

Logistic regression is a statistical model that is used to model the probability of a certain class or event existing such as benign/malignant. Mathematically, a binary logistic model has a dependent variable with two possible values, such as benign/malignant which is presented by an indicator variable, where the two values are labeled “0” and “1”. Logistic regression in its basic form uses a logistic function also called the sigmoid function which is a mathematical function having a characteristic “S”-shaped curve.

```
[27]: logReg = LogisticRegression(random_state=0)
      modelLR = logReg.fit(X_train, Y_train)
      predictionLR = modelLR.predict(X_test)

[28]: cmLR = confusion_matrix(Y_test, predictionLR)
      cmLR

[28]: array([[112,  3],
            [ 2,  54]], dtype=int64)

[29]: sns.heatmap(cmLR,annot=True)

[29]: <AxesSubplot:>
```



```
[30]: print("Training accuracy: ", accuracy_score(Y_train, modelLR.predict(X_train)))
      print("Accuracy score function: ",accuracy_score(Y_test, predictionLR))
      print("Classification report\n", classification_report(Y_test, predictionLR))

Training accuracy: 0.9824120603015075
Accuracy score function: 0.9707602339181286
Classification report
      precision    recall  f1-score   support

      0       0.98      0.97      0.98       115
      1       0.95      0.96      0.96        56

   accuracy          0.97          171
  macro avg          0.96          171
 weighted avg          0.97          171
```

Analysing the performance is easiest by using the confusion matrix. Confusion matrix is a technique of summarizing the performance of a classification algorithm. On [0][0] position we get True Positive results – tumor is malignant and model is giving it the right diagnosis, [1][1] True Negative – tumor

is benign and model classified it so, [0][1] False Positive – tumor is benign, but the model isn't diagnosing it rightfully, [1][0] False Negative – tumor is malignant, but the model doesn't classify it as so.

Comparing training and testing accuracy I notice that there is not a huge difference between the scores, which means that there is no overfitting and logistic regression has a really high accuracy of 97%.

b) Decision Tree

Decision tree is a classifier that has nodes and branches in which it uses to split the data into different categories. It constructs different categories by recursive evaluation of data. This procedure repeats itself by updating the nodes till the data is correctly separated.

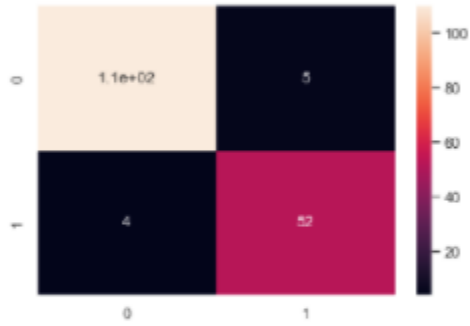
```
[31]: decTree = DecisionTreeClassifier(criterion='entropy', random_state=0)
      modelDT = decTree.fit(X_train, Y_train)
      predictionDT = modelDT.predict(X_test)

[32]: cmDT = confusion_matrix(Y_test, predictionDT)
      cmDT

[32]: array([[110,  5],
            [ 4, 52]], dtype=int64)

[33]: sns.heatmap(cmDT,annot=True)

[33]: <AxesSubplot:>
```



```
[34]: print("Training accuracy: ", accuracy_score(Y_train, modelDT.predict(X_train)))
      print("Accuracy score function: ",accuracy_score(Y_test, predictionDT))
      print("Classification report\n", classification_report(Y_test, predictionDT))

Training accuracy: 1.0
Accuracy score function: 0.9473684210526315
Classification report
      precision    recall  f1-score   support

      0       0.96       0.96       0.96        115
      1       0.91       0.93       0.92         56

   accuracy          0.95          171
  macro avg          0.94          171
 weighted avg          0.95          171
```

Decision Tree has a bit lower accuracy on this data set then Logistic Regression. Accuracy is 95%, and again the difference between training accuracy score and accuracy score of the testing data is not big, so the model is not overfitting.

c) Random Forest

Random Forest is, in a simplest way explained, a group of decision trees. Every decision tree gives a “vote” in the final decision by making a decision of the result on its own. The decision that is “voted” the most times gets picked for a result.

```
[51]: randForest = RandomForestClassifier(criterion='entropy', random_state=0)
      modelRF = randForest.fit(X_train,Y_train)
      predictionRF = modelRF.predict(X_test)

[52]: cmRF = confusion_matrix(Y_test, predictionRF)
      cmRF

[52]: array([[112,  3],
            [ 2,  54]], dtype=int64)

[53]: sns.heatmap(cmRF,annot=True)

[53]: <AxesSubplot:>
```



```
[54]: print("Training accuracy: ", accuracy_score(Y_train, modelRF.predict(X_train)))
      print("Accuracy score function: ",accuracy_score(Y_test, predictionRF))
      print("Classification report\n", classification_report(Y_test, predictionRF))

Training accuracy: 1.0
Accuracy score function: 0.9707602339181286
Classification report
              precision    recall  f1-score   support

     0       0.98       0.97       0.98       115
     1       0.95       0.96       0.96        56

 accuracy          0.97          0.97          0.97       171
 macro avg         0.96          0.97          0.97       171
 weighted avg      0.97          0.97          0.97       171
```

Random Forest Classifier has a higher accuracy than decision tree by 2%, and the same recall time. Logistic Regression, however has the same accuracy as Random Forest tree when there is 100 (default set) decision trees in the random forest. The only difference between these results is that training accuracy of Logistic regression scores a lower precision.

5. Literature

1. <https://www.kaggle.com/uciml/breast-cancer-wisconsin-data>
2. <https://scikit-learn.org/stable/>
3. http://theta.edu.pl/wp-content/uploads/2012/10/exploratorydataanalysis_tukey.pdf
4. https://en.wikipedia.org/wiki/Logistic_regression
5. https://en.wikipedia.org/wiki/Decision_tree
6. https://en.wikipedia.org/wiki/Random_forest