Laboratory Assignment 1: Classification

MACHINE LEARNING I

MARTA SIMÓN PINACHO, ANA ROTELLA FERNANDEZ, ANA MIGUELEZ MARTINEZ

1. Exploratory data analysis and preprocesing

STEP 1: IMPORT DATASET

First of all, we imported the dataset Diabetes.csv and we saved it in the variable 'diabetes'. We have printed the data frame to observe the variables with which we are going to work.

]:

	PREGNANT	GLUCOSE	BLOODPRESS	SKINTHICKNESS	INSULIN	BODYMASSINDEX	PEDIGREEFUNC	AGE	DIABETES
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1
...	...	...	...	...	...	...	...	...	...
763	10	101	76	48	180	32.9	0.171	63	0
764	2	122	70	27	0	36.8	0.340	27	0
765	5	121	72	23	112	26.2	0.245	30	0
766	1	126	60	0	0	30.1	0.349	47	1
767	1	93	70	31	0	30.4	0.315	23	0

768 rows × 9 columns

We have 768 rows and 9 variables or columns. Of these 9 variables, there are 8 input variables and 1 output variable.

Input variables:

- PREGNANT
- GLUCOSE
- BLOOD PRESS
- BODYMASSINDEX
- INSULIN
- SKINTHICKNESS
- PEDIGREEFUNC
- AGE

Output variable:

- DIABETES

Here we can see that we have 9 variables, 7 of type int64 and 2 of type float64.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   PREGNANT       768 non-null    int64
 1   GLUCOSE        768 non-null    int64
 2   BLOODPRESS     768 non-null    int64
 3   SKINTHICKNESS  768 non-null    int64
 4   INSULIN        768 non-null    int64
 5   BODYMASSINDEX  768 non-null    float64
 6   PEDIGREEFUNC   768 non-null    float64
 7   AGE            768 non-null    int64
 8   DIABETES       768 non-null    int64
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
```

STEP 2: CHECK OUT THE MISSING VALUES

We see how many non-null values there are for each of the variables

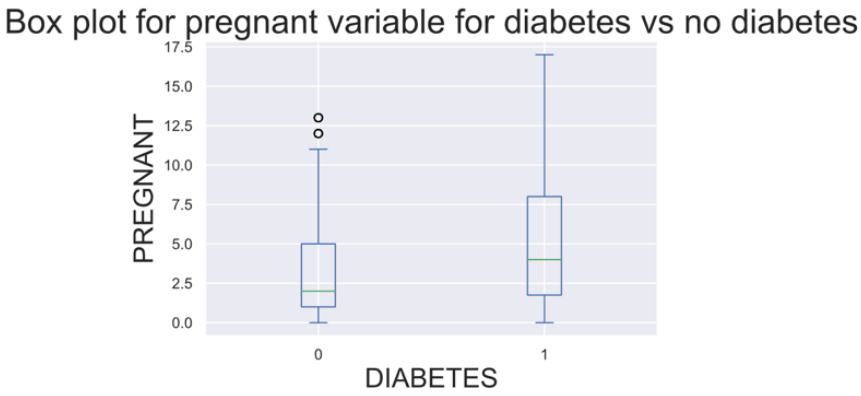
```
Number of NAs in dataframe: 0

PREGNANT         False
GLUCOSE          False
BLOODPRESS       False
SKINTHICKNESS    False
INSULIN          False
BODYMASSINDEX    False
PEDIGREEFUNC     False
AGE              False
DIABETES         False
dtype: bool
```

As we have seen, none of the variables have non-null values, so there is no need to remove them.

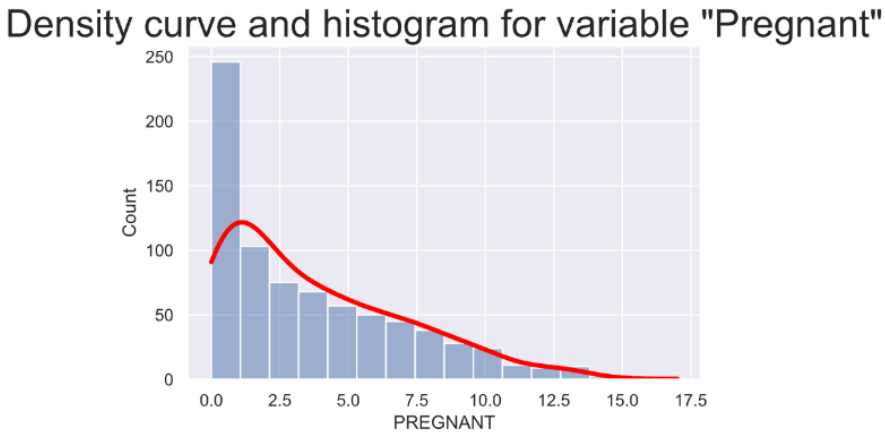
STEP 3: PLOT THE DATA AND CHECK OUT FOR OUTLIERS

We do an analysis of each of the output variables to decide whether to remove the outliers or not:

A. PREGNANT



This variable has a total of 4 outliers, all of them are above the upper limit, and they are 15, 17, 14 and 14.

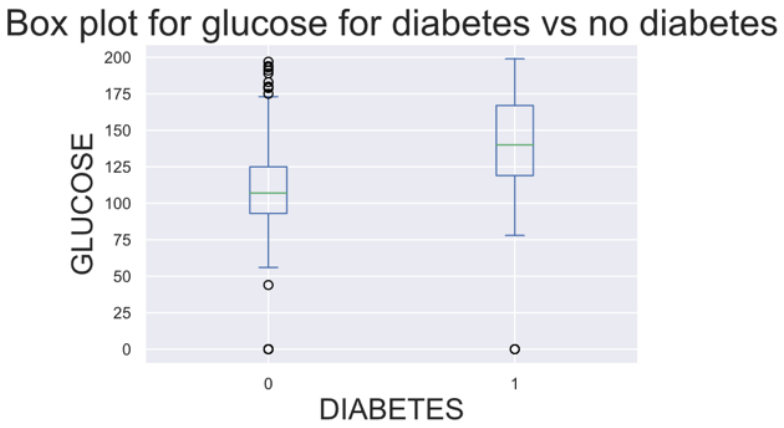


Conclusion of the outliers for PREGNANT

In this variable, most cases occur in 0, 1 and 2 pregnancies, as we can see in the histogram. 0 pregnancies are a normal case, and 14, 15 and 17 pregnancies are not very likely, but it can also be the case.

As these are values that can occur, so we have decided not to eliminate or replace the outliers.

B. GLUCOSE



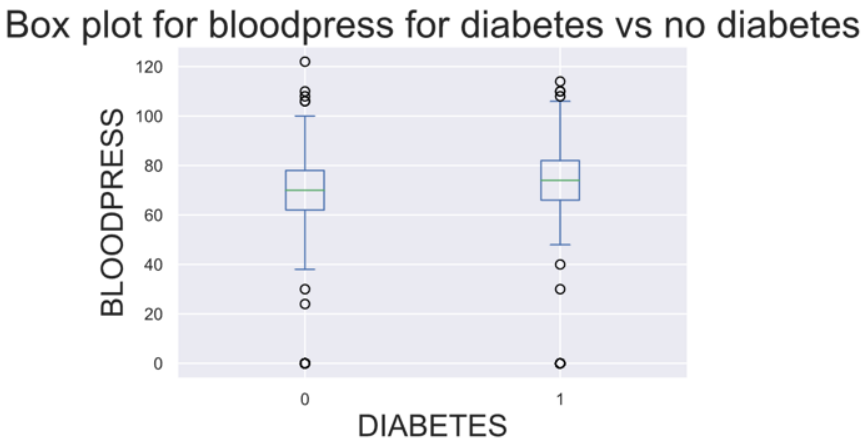
Conclusion of the outliers for GLUCOSE

The GLUCOSE variable has a total of 5 outliers, all with a value of 0, as we can see in the boxplot.

Low blood sugar level is a serious medical emergency. This can cause seizures and brain damage. A blood sugar level below 70 mg/dL is considered low.

For this reason, in this variable we have decided to eliminate the outliers, since a glucose level equal to 0 is impossible in a person who is alive.

C. BLOODPRESS



Conclusion of the outliers for BLOODPRESS

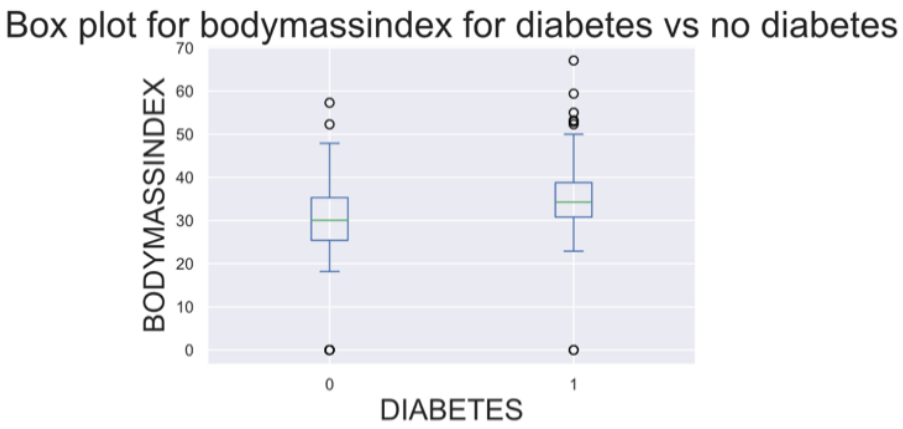
The BLOODPRESS variable has a total of 45 outliers. The values are above the upper limit and below the lower limit

The minimum value of this variable is 0, this means that the blood pressure is too low, which can cause the body's oxygen levels to decrease and cause damage to the heart and brain.

On the other hand, the maximum blood pressure value is 122, around 80 and above is already considered high blood pressure.

For both reasons, we have decided to eliminate the outliers of this variable.

D. BODYMASSINDEX



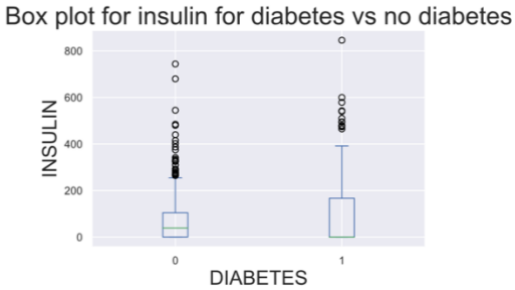
Conclusion of the outliers for BODYMASSINDEX

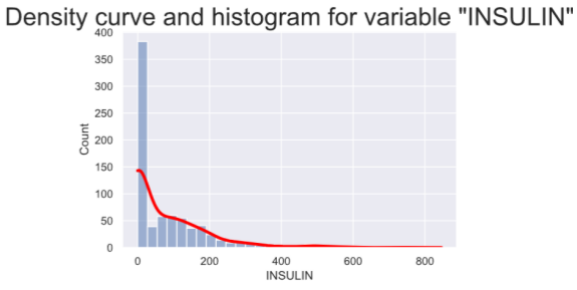
The variable BODYMASSINDEX indicates the body mass index, which is measured considering the weight and height of each person.

There are a total of 19 outliers, of which 11 have a value of 0, this would mean that the person would lack body mass, which is impossible.

For this reason, we have decided to eliminate the outliers of this variable.

E. INSULIN





Conclusion of the outliers for INSULIN

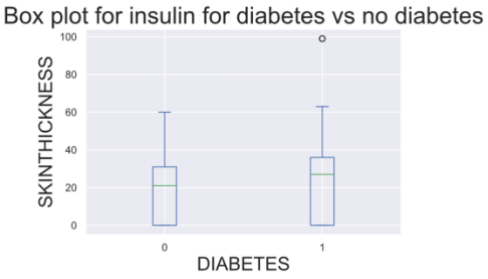
The INSULIN variable has a total of 34 outliers.

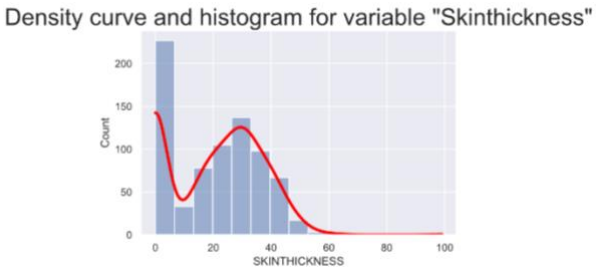
In this variable, it is normal for the values to be between 5-26 U/ml. Therefore, in the graph of this variable it is normal to find values close to 0.

If a person has 300 U/ml or more of insulin in the bloodstream it can cause hypoglycemia.

The outliers that are located at values greater than 318,125 U/ml (which is the upper limit) are cases that can occur, so we have decided not to eliminate or replace the outliers of this variable.

F. SKINTHICKNESS





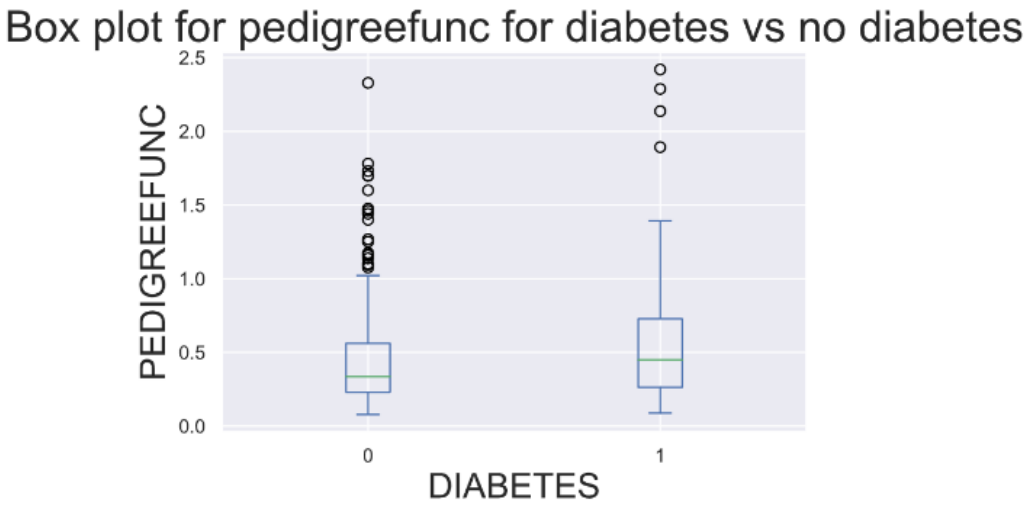
Conclusion of the outliers for SKINTHICKNESS

The SKINTHICKNESS variable has only one outlier, with a value of 99, which, as we can see in the boxplot, is quite far from the rest of the data. As it is an isolated case compared to the rest of the values of the variable, we decided to eliminate the outlier of this variable.

On the other hand, we have observed that we have many zeros in the variable (227 in total), so we have decided to graph it. This measurement is not a normal value to measure the thickness of the skin, so we assume that they are measurements that were not taken at the time of analyzing the patient.

There are 227 variables that do not provide information to the variable, that is, 29.55% of the total. For this reason, it is a variable that we will delete it for the new data frame.

G. PEDIGREEFUNC



Conclusion of the outliers for PEDIGREEFUNC

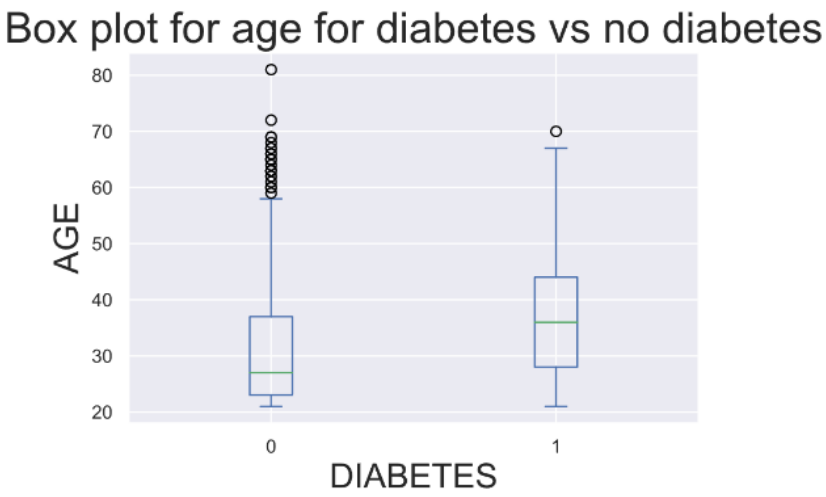
The PEDIGREEFUNC variable has a total of 29 outliers.

As we can see in the boxplot, all the outliers have values above 1.2, which is the upper limit.

This variable indicates the relationship that the person has at the genetic level with diabetes, the higher the value, the higher the relationship with the pathology.

So, in this case we have decided not to remove the outliers.

H. AGE



Conclusion of the outliers for AGE

The AGE variable has a total of 9 outliers, which are above the upper limit (66.5 years).

This variable indicates the ages of women from 21 years. All the outliers show older ages (69, 67, 72…) which are cases that occur, so we have decided not to eliminate the outliers.

CONCLUSION OF THE OUTLIERS

The next step is to eliminate the outliers of the variables that we have decided.

- GLUCOSE
- BLOOD PRESS
- BODYMASSINDEX
- SKINTHICKNESS -> We eliminate the column

It is important to mention that we are going to train the models with two data frames, one with the original data frame (diabetes), and another with the modified data, so the modifications that we are going to make, are going to be on the new data frame (diabetes_new).

STEP 4: ENCODE CATEGORICAL VARIABLES

There are no categorical input variables, but 'DIABETES' would be a categorical value. In this step we convert output variable to factor, in this case 'DIABETES' is as a categorical variable. We make this transformation on both data frames (diabetes, diabetes_new).

STEP 5: EXPLORATORY ANALYSIS

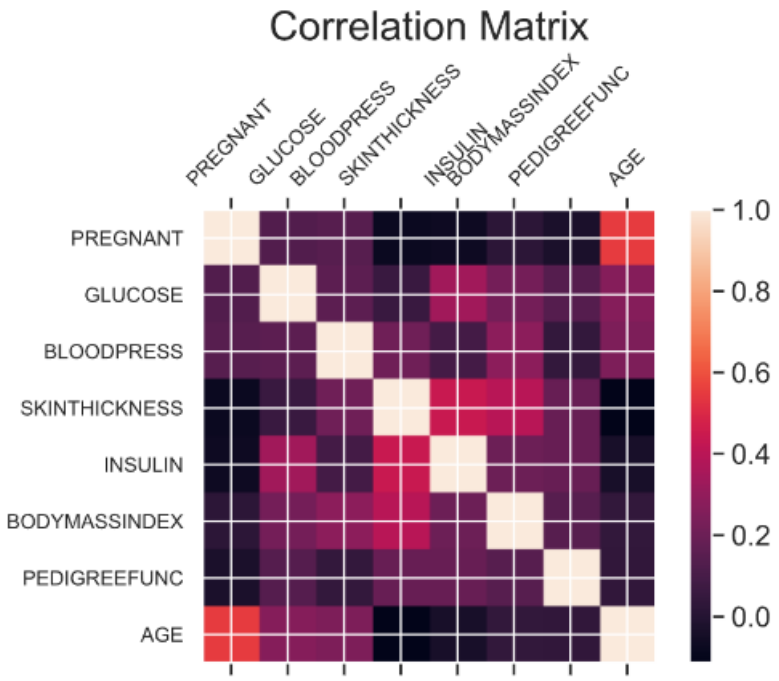
PAIRPLOT:

In this step we get a correlation plot of numeric variables with 'pairplot' and with de correlation matrix.

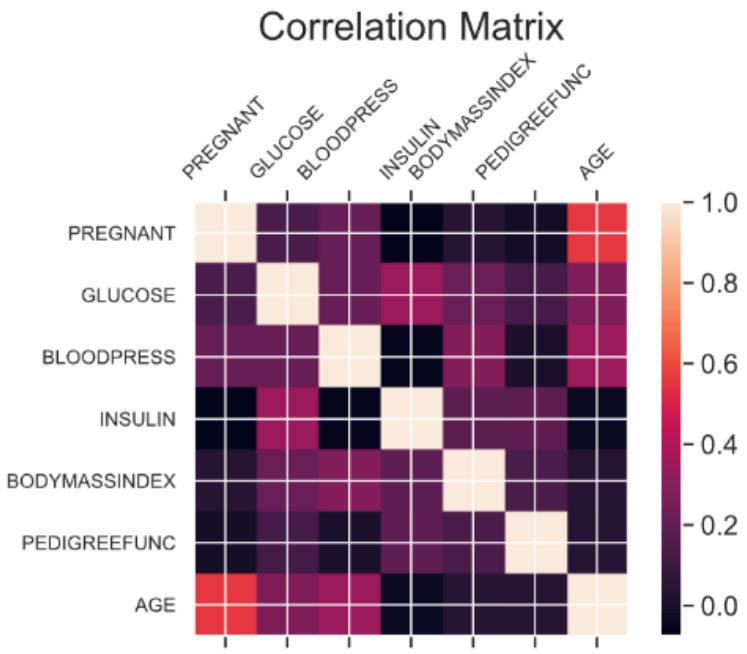
CORRELATION MATRIX:

We elaborate a correlation matrix to detect the correlation between each one of the variables, and the importance of each one of them in the dataset (the more correlated they are between them, the less information they will contribute to our model).

- Correlation matrix with the original dataset (diabetes)



- Correlation matrix with new dataset (diabetes_new)



Conclusion for both correlation matrix

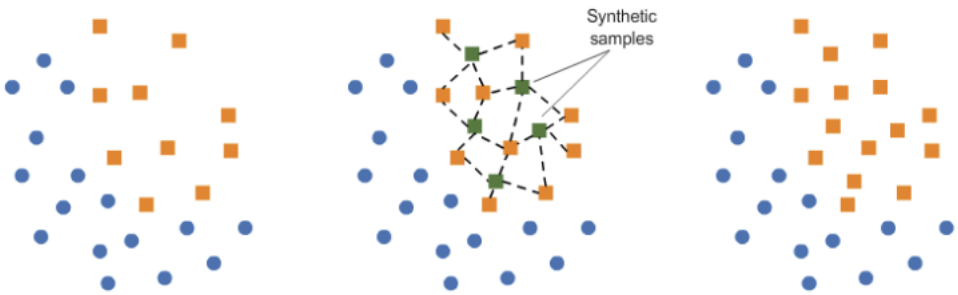
Both present almost the same correlation between the variables. Noteworthy, the highest correlation is AGE and PREGNANT, with a correlation of 0.544. This amount of correlation between two variables is considered moderate, so they continue to contribute information to the analysis.

STEP 6: CHECK OUT FOR CLASS IMBALANCES

As we can see, in the frequency table of the output variable there are many more records of class 0 than of class 1, specifically, we have 65% of the data in class 0, and 35% of the data in class 1. So, we are dealing with unbalanced data.

```
0    468
1    241
Name: DIABETES, dtype: int64
```

To balance the classes, we are going to use the **imblearn** library, specifically, the oversampling method called SMOTE. This method generates synthetic data of the minority class, to obtain samples like our data and to balance the classes of the dataset.



After balancing the data, we have obtained a dataset with 936 observations, while the previous one had a total of 709.

```
Original dataset shape 709
Resample dataset shape 936
```

As we can see, when balancing the data, values have been added in class 1 of the DIABETES variable, in this way, having 50% of the data at 0 and 50% of the data at 1.

```
: 0    468
  1    468
  Name: DIABETES, dtype: int64
```

COMPARE THE METRICS OF THESE TWO DATASETS

Once the outliers have been eliminated, the data has been balanced, the correlation between the variables has been detected, and the importance they have for the dataset, we check how the metrics of the variables have changed before and after balancing the data and eliminating the outliers.

- Original dataset (diabetes)

|:

	PREGNANT	GLUCOSE	BLOODPRESS	SKINTHICKNESS	INSULIN	BODYMASSINDEX	PEDIGREEFUNC	AGE
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000
mean	3.845052	120.894531	69.105469	20.536458	79.799479	31.992578	0.471876	33.240885
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	0.331329	11.760232
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.078000	21.000000
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	0.243750	24.000000
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	0.372500	29.000000
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	0.626250	41.000000
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	2.420000	81.000000

- Modificated dataset (diabetes_new)

|:

	PREGNANT	GLUCOSE	BLOODPRESS	INSULIN	BODYMASSINDEX	PEDIGREEFUNC	AGE
count	936.000000	936.000000	936.000000	936.000000	936.000000	936.000000	936.000000
mean	4.042735	126.322650	72.774573	87.320513	32.801580	0.487708	34.293803
std	3.291261	31.188483	11.081392	119.161688	6.165188	0.322424	11.467518
min	0.000000	44.000000	38.000000	0.000000	18.200000	0.078000	21.000000
25%	1.000000	103.000000	65.000000	0.000000	28.525933	0.258000	25.000000
50%	3.000000	122.000000	73.000000	48.500000	32.900000	0.402000	31.000000
75%	6.000000	147.000000	80.000000	142.000000	36.725000	0.637386	42.000000
max	17.000000	199.000000	106.000000	846.000000	50.000000	2.329000	81.000000

Here we have the metrics of the output variables before and after balancing the data and removing outliers.

The main difference is the number of observations of the variables. Before balancing the data and removing the outliers we had a total of 768 variables (468 for 0 and 241 for 1), and after balancing we have a total of 936 variables (468 for 0 and 468 for 1).

On the other hand, in the new dataframe, we do not have the SKINTHICKNESS variable, since we have removed it.

As for the rest of the metrics, they have hardly changed, except for the minimum and maximum, which after eliminating the outliers have been modified in the GLUCOSE, BLOODPRESS AND BODYMASSINDEX variables.

- GLUCOSE: the minimum has passed from 0 to 44, and the maximum remains
- BLOODPRESS: The minimum has gone from 0 to 38, and the maximum from 122 to 106. This is because the outliers were above the upper limit, and below the lower limit.
- BODYMASSINDEX: the minimum has changed from 0 to 18.2, and the maximum from 67 to 50. Both values have been modified for the same reason as with the BLOODPRESS variable.

STEP 7: Split the dataset into train and test

We split the data randomly into train and test. The train has 80% of the data and the test 20%.

We do this division to have a training set (train) in which we have the data from which we start to train the model, and then we validate it with the validation set (test).

Overfitting must be avoided, that is, the train set is overlearned, and then it does not provide good results for the test.

We are creating 2 input variables and 2 output variables.

- Input and output variables with the original dataset. It has unbalanced data and has outliers
- Input and output variables with the new dataset. It has balanced data and no outliers

2. Identification and fitting process of classification models

Next, we will briefly discuss each of the models that we have applied to our dataset, focusing on the choice of the parameters of each of the models and commenting on the results that we obtain with each of them by applying them to the two datasets that we have generated, the original and the clean one.

Decission Trees

Configuration of the model. Selecting parameters

The first model we are going to make is a decision tree since it will give us information about the importance of the variables in our model. It will allow us to make some first decisions.

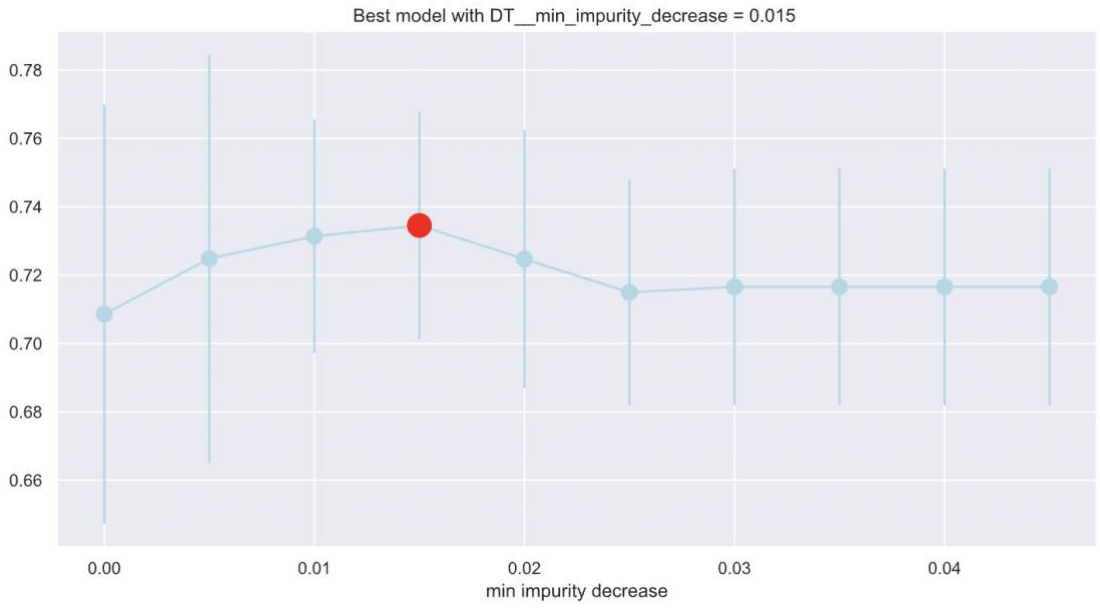
We are going to make a decision tree for each dataset we have, the first one will be for the original dataset (with all values and without modifying anything) and the second model will be with the modified dataset data (without outliers and balanced).

- *TEST WITH ORIGINAL DATASET*

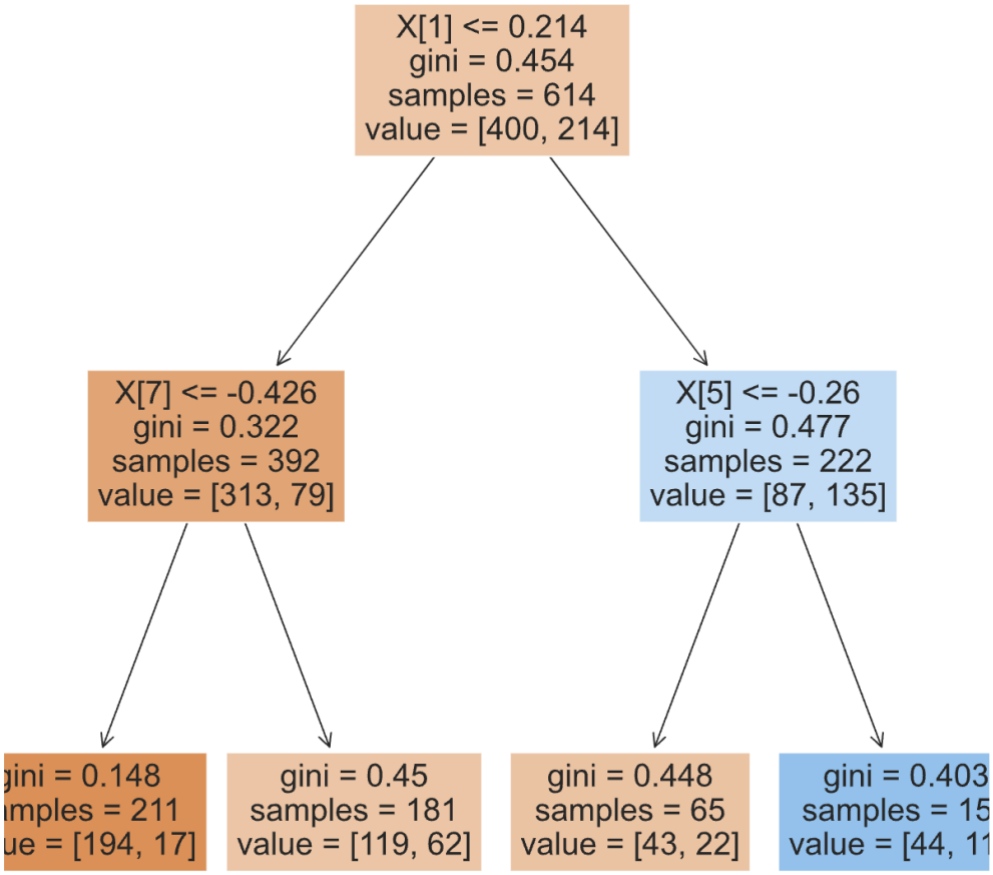
  For this model we will use as INPUTS all the variables of the dataset, and as output we will use the response variable 'DIABETES'.

  Other extra parameters we have defined for the model are the. 'DT__min_impurity_decrease', we have tried with several ranges but we have chosen the 'DT__min_impurity_decrease' parameter.

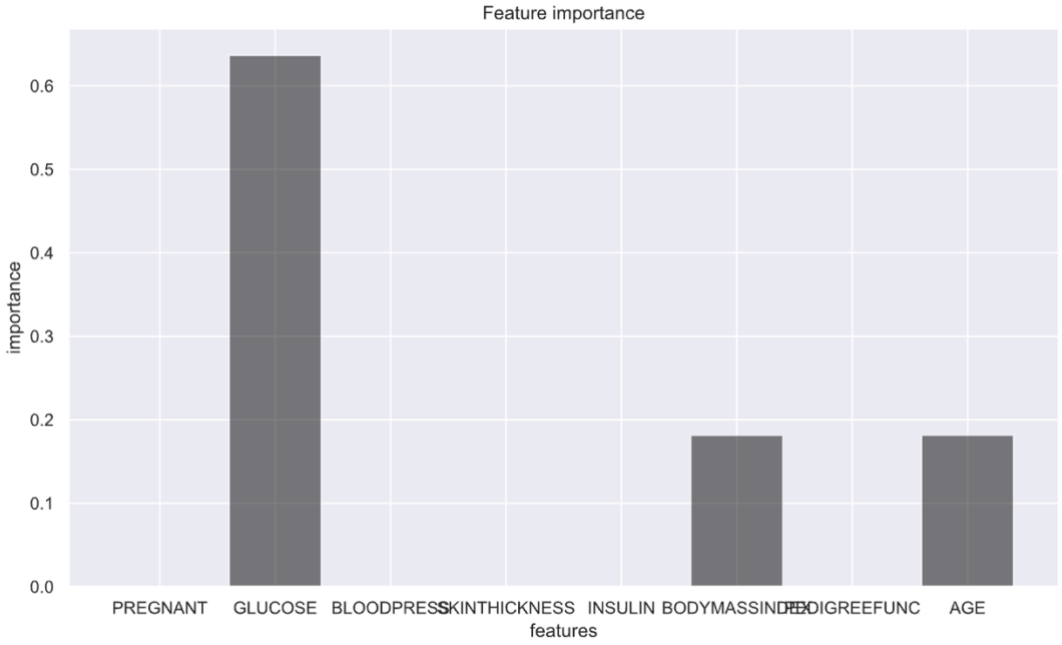
```
param = {'DT__min_impurity_decrease': np.arange(0,0.05,0.005)}
```



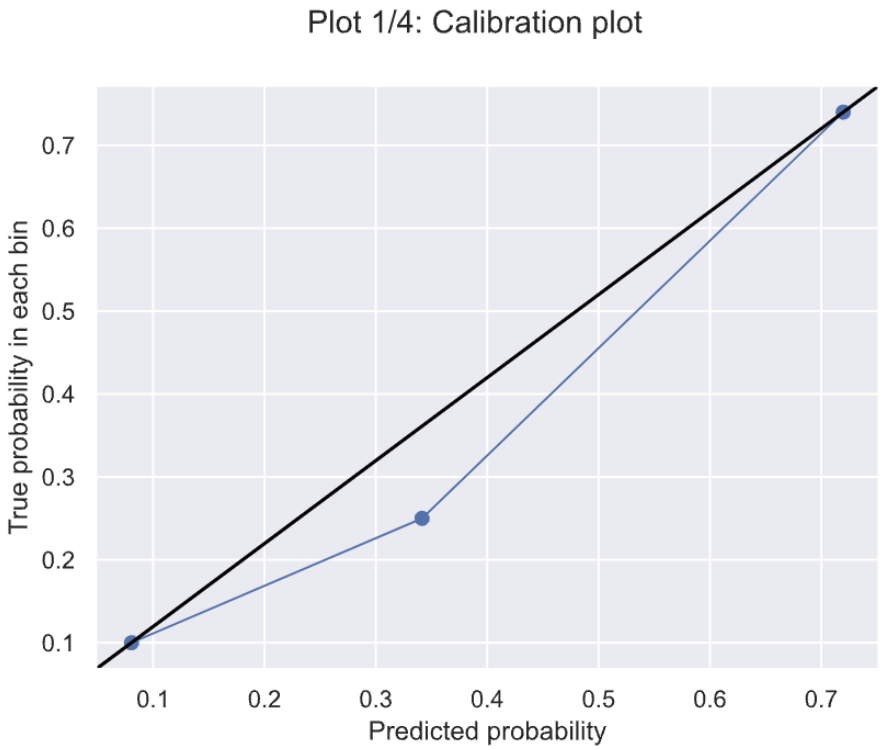
We can see that it has reached the maximum for the value 0.015 of the range.



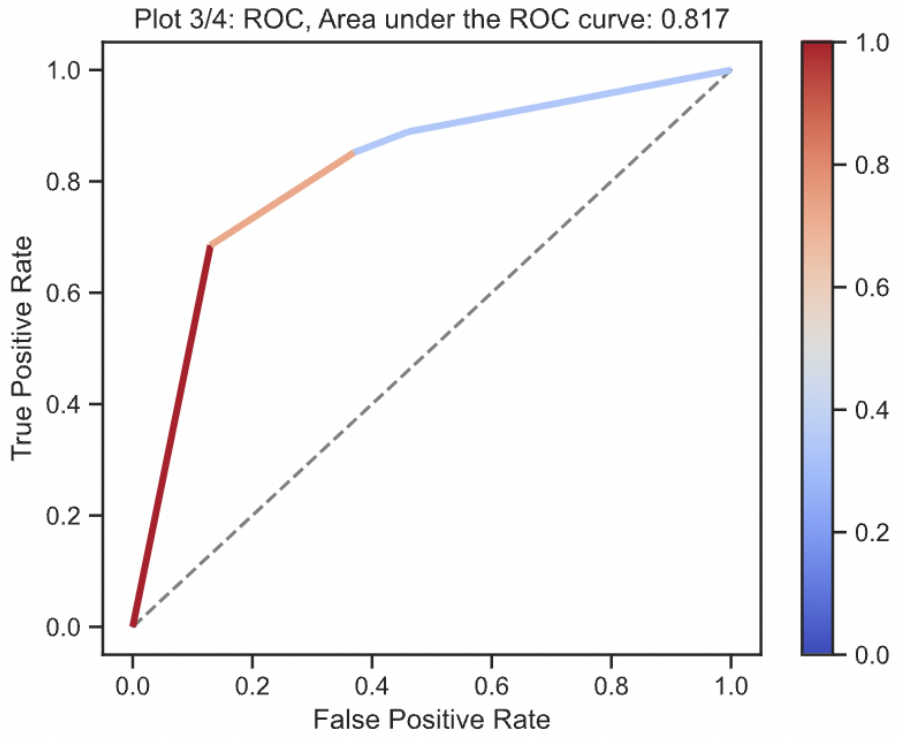
When we look at the tree we see that the most significant variables are X[1], X[7] and X[5]. Another way to see this is with the following graph. We can see that it is a tree that is simple.



We see that variable 1 is GLUCOSE, variable 5 is BMI and variable 7 is AGE.



The ROC curve has a value of 0.817, which is quite positive for the first model.



- *TEST WITH MODIFICATED DATASET*

In this new model, what we have done is to use the dataset that is balanced and without outliers. The most logical thing would be to have a similar result to the previous decision tree model.

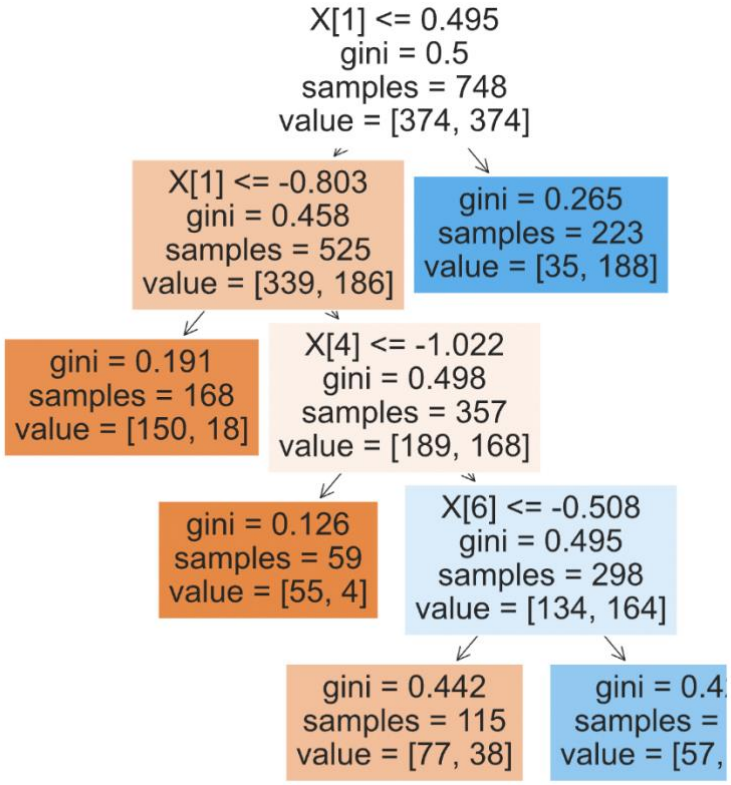
In this new model we will use as explanatory variable the following INPUTS and as response variable 'DIABETES'.

```
INPUTS_NEW =
['PREGNANT','GLUCOSE','BLOODPRESS','INSULIN','BODYMA
SSINDEX','PEDIGREEFUNC','AGE']
OUTPUT_NEW = 'DIABETES'
```

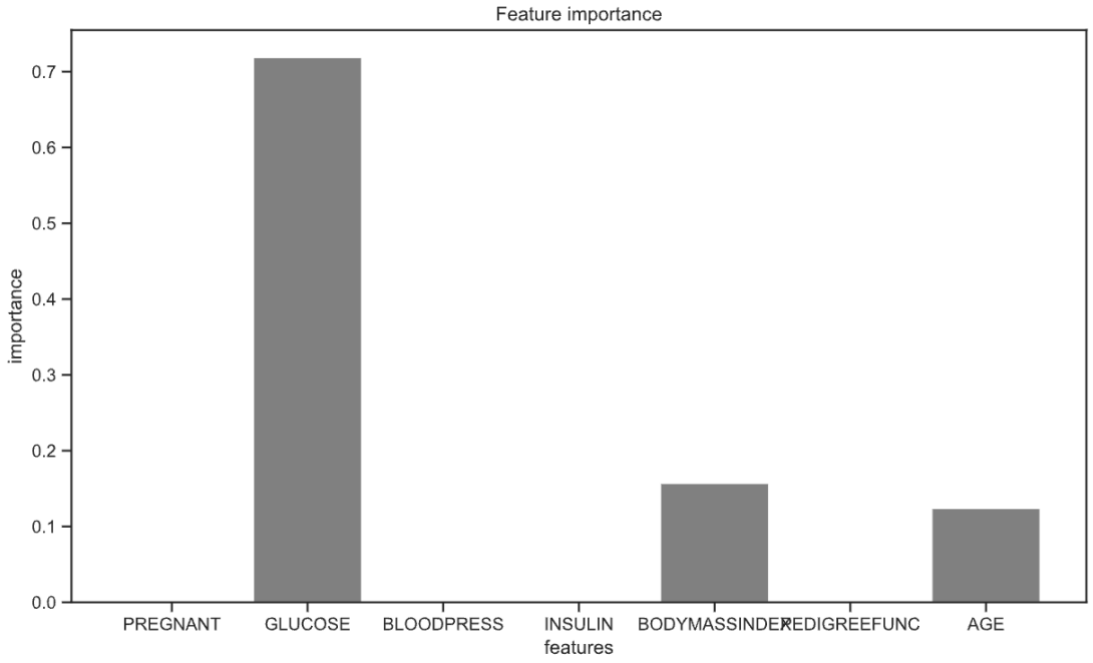
We have used the same hyperparameters as in the previous model since we have seen that it maximizes in that range.

```
param = {'DT__min_impurity_decrease':
np.arange(0,0.05,0.005)}
```

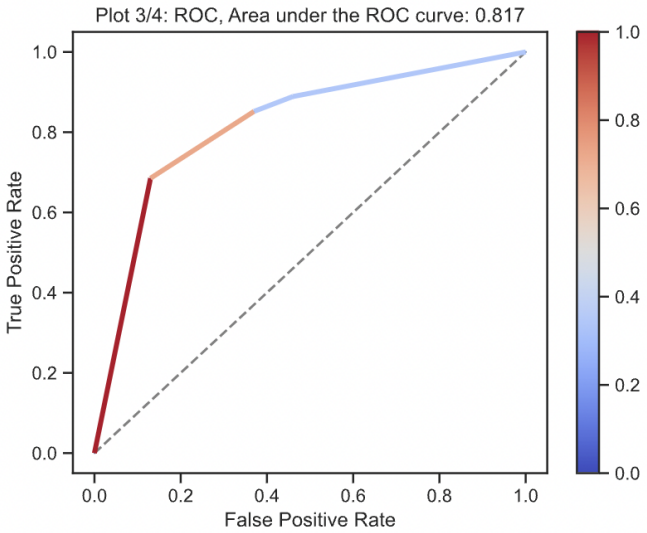
The decision tree we have obtained is as follows:

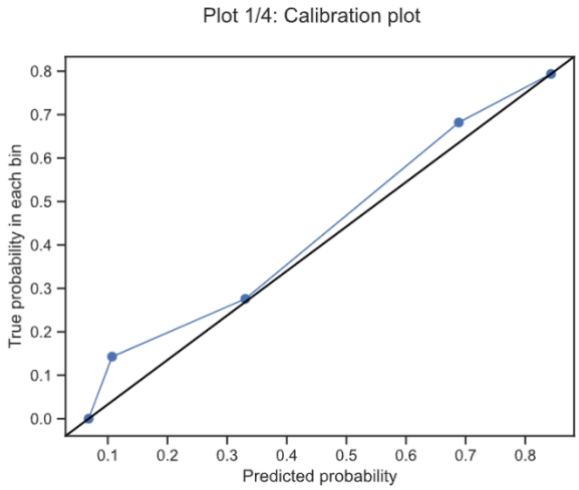


We see that we have exactly the same variables that are significant for the orther model to decision tree:



Next, we can see how the calibration graph and the ROC curve of this model look like.





As we can see, the values we have obtained are very similar to each other. So it is a good sign, since one of them is unbalanced and the other balanced and that means that we have managed to balance it without modifying the data

Logistic Regression

Configuration of the model. Selecting parameters

For this type of classification we have followed the same dynamics as in the previous ones, we have made a study for the original dataset and then we have made a model on the modified data, eliminating outliers and balancing the classes.

- *TEST WITH ORIGINAL DATASET*

The first model, as we can see in the code, has been a model with all the data of the pure dataset. Taking into account this, and seeing which variables were the most significant for this model, we have made a second model with only the most significant variables, and we can see that the same results are obtained with a simpler model.

As inputs to the model we have used all the variables mentioned above and as output the diabetes variable, which is our response variable.

```
INPUTS_ALL =
['PREGNANT','GLUCOSE','BLOODPRESS','SKINTHICKNESS','INSULIN','BO
DYMASSINDEX','PEDIGREEFUNC','AGE']
OUTPUT = 'DIABETES'
```

Once we have trained the model (see notebook), we have obtained which are the most significant variables (***), this will help us to make decisions for other models.

```
CT.summaryLogReg(LogReg_fit_Original, X_train[INPUTS_ALL],
y_train)
```

```
Deviance Residuals:
        Min         1Q     Median         3Q        Max
0 -0.960377 -0.247754 -0.095737  0.268054  0.980029

Coefficients:
              Estimate
Intercept     -0.847913
PREGNANT       0.392084
GLUCOSE        0.986186
BLOODPRESS    -0.227106
SKINTHICKNESS -0.034883
INSULIN       -0.047038
BODYMASSINDEX  0.625000
PEDIGREEFUNC   0.306933
AGE            0.243356
              Estimate  Std. Err   t-value      Pr(>|t|) Signif
Intercept     -0.847913  0.105634 -8.026912  1.110223e-15    ***
PREGNANT       0.392084  0.116019  3.379468  7.262638e-04    ***
GLUCOSE        0.986186  0.127060  7.761575  8.437695e-15    ***
BLOODPRESS    -0.227106  0.110483 -2.055566  3.982434e-02      *
SKINTHICKNESS -0.034883  0.119559 -0.291765  7.704666e-01
INSULIN       -0.047038  0.112906 -0.416610  6.769636e-01
BODYMASSINDEX  0.625000  0.128958  4.846542  1.256320e-06    ***
PEDIGREEFUNC   0.306933  0.109628  2.799767  5.113945e-03     **
AGE            0.243356  0.118128  2.060107  3.938827e-02      *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

As we can see in the picture above INSULIN and SKINTHICKNESS are not at all significant for the model, AGE and BLOODPRESS are not very significant for the model, so we have tested both situations. Seeing the results we have seen that it was better to remove the 4, since the others hardly contribute to the model.

The next model I have done for linear regression is by removing these variables mentioned above where now the inputs have changed.

```
OUTPUT = 'DIABETES'

INPUTS_LR_MOD =
['PREGNANT','GLUCOSE','BODYMASSINDEX','PEDIGREEFUNC']
```

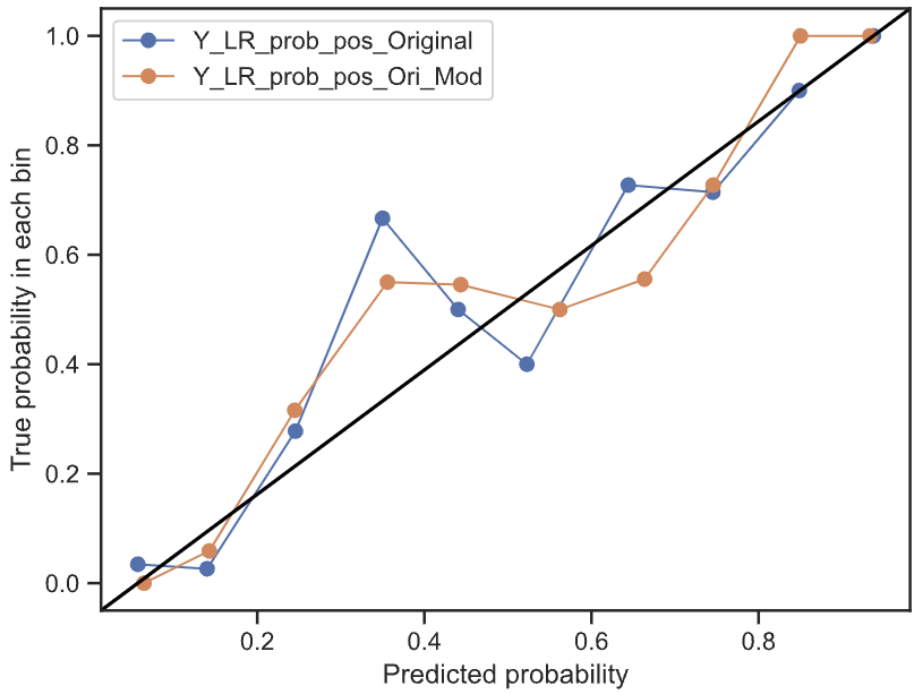
Once this model is done, we see that all the variables are quite significant for the model.

```
CT.summaryLogReg(LogReg_fit_Original_Clean,
X_train[INPUTS_LR_MOD], y_train)
```

```
Deviance Residuals:
        Min         1Q    Median        3Q       Max
0 -0.971827 -0.252159 -0.10039   0.27053  0.981695

Coefficients:
               Estimate
Intercept     -0.829764
PREGNANT       0.489057
GLUCOSE        1.002663
BODYMASSINDEX  0.522501
PEDIGREEFUNC   0.303820
               Estimate  Std. Err   t-value      Pr(>|t|) Signif
Intercept     -0.829764  0.103932 -7.983719  1.332268e-15    ***
PREGNANT       0.489057  0.100176  4.881960  1.050364e-06    ***
GLUCOSE        1.002663  0.117116  8.561247  0.000000e+00    ***
BODYMASSINDEX  0.522501  0.115453  4.525671  6.020406e-06    ***
PEDIGREEFUNC   0.303820  0.107194  2.834289  4.592781e-03     **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

To compare these two models we will look at the calibration plot and the ROC curve.

```
CT.calibration_plot(y_test,
dfTS_eval[['Y_LR_prob_pos_Original','Y_LR_prob_pos_Ori_Mod'
]])
```

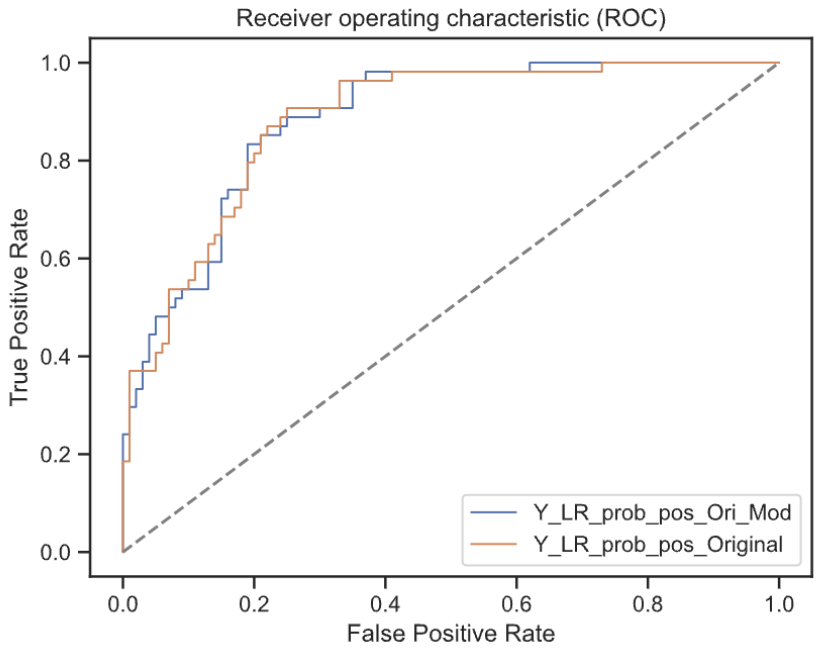


Looking at the previous graph we see that the ideal would be that it fits as much as possible to the line, as we see the orange graph fits better than the blue one, being the blue the original

one (dataset with all INPUTS) and the orange the graph belonging to the second case (ONLY SIGNIFICANT INPUTS).

```
CT.roc_curve(y_test,dfTS_eval[['Y_LR_prob_pos_Ori_Mod','Y_LR_pro
b_pos_Original']], 1)
```

```
Area under the ROC curve of Y_LR_prob_pos_Ori_Mod : 0.885
Area under the ROC curve of Y_LR_prob_pos_Original : 0.884
```



Looking at the above graph we see that the ROC curve of the second model is better than the first one, although not much better.

- *TEST WITH MODIFICATED DATASET*

Now we are going to make the same models with the modified dataset. To modify the dataset what we have done is to balance the data and eliminate the outliers (see exploratory analysis section).

The first model is with the whole dataset, where we will see which are the most significant variables, which will help us to make the second model.

We used the following as model variables:

```
OUTPUT = 'DIABETES'

INPUTS_NEW =
['PREGNANT','GLUCOSE','BLOODPRESS','INSULIN','BODYMASSINDEX',
'PEDIGREEFUNC','AGE']
```

```
CT.summaryLogReg(LogReg_fit_Modificado,
X_train_new[INPUTS_LR_NEW], y_train_new)
```

```
Deviance Residuals:
       Min       1Q    Median        3Q       Max
0 -0.98802 -0.24374 -0.002403  0.225671  0.878594

Coefficients:
              Estimate
Intercept      0.042675
PREGNANT       0.276284
GLUCOSE        1.289440
BLOODPRESS    -0.118481
INSULIN       -0.205059
BODYMASSINDEX  0.749069
PEDIGREEFUNC   0.271008
AGE            0.290952
              Estimate  Std. Err    t-value      Pr(>|t|) Signif
Intercept      0.042675  0.093374   0.457032  6.476479e-01
PREGNANT       0.276284  0.108047   2.557078  1.055556e-02      *
GLUCOSE        1.289440  0.125647  10.262366  0.000000e+00    ***
BLOODPRESS    -0.118481  0.103699  -1.142546  2.532269e-01
INSULIN       -0.205059  0.105531  -1.943113  5.200246e-02      .
BODYMASSINDEX  0.749069  0.111111   6.741646  1.566014e-11    ***
PEDIGREEFUNC   0.271008  0.101082   2.681080  7.338502e-03     **
AGE            0.290952  0.114485   2.541402  1.104090e-02      *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

As we can see in the previous image the results are very similar to the first model with the whole dataset. The only change we can see is that in this PREGNANT and AGE are equally significant, these were the only two variables we saw in the correlation matrix that were slightly correlated.

We are going to make a second model keeping only the variables that contribute more meaning to the model.

```
INPUTS_LR_MOD_CLEAN =
['GLUCOSE','BODYMASSINDEX','PEDIGREEFUNC','AGE']
```

Once we have made the model we see that the variables that are most significant are very similar to the result we obtained for the model with the original dataset.

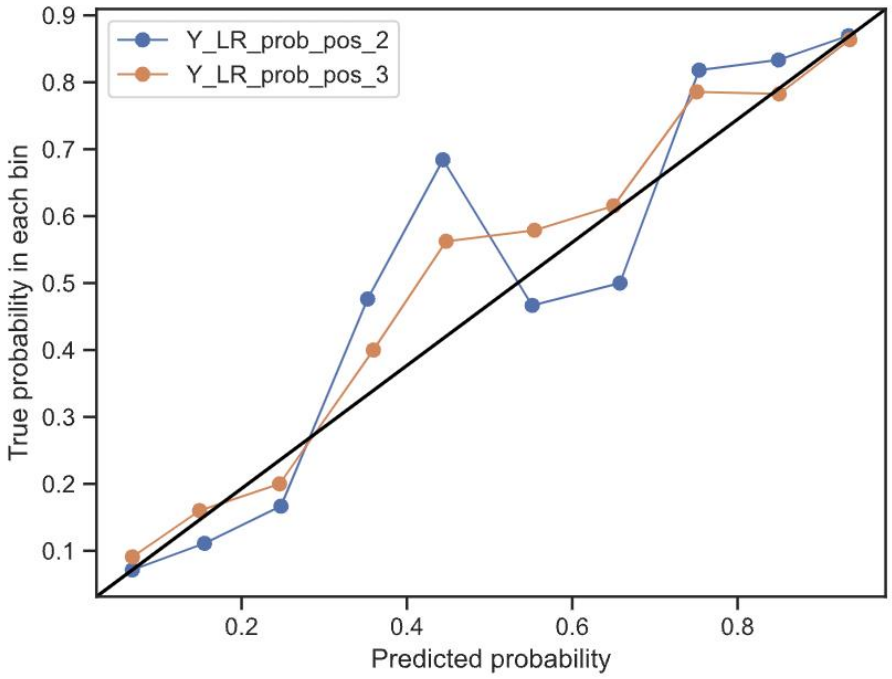
```
CT.summaryLogReg(LogReg_fit_Clean,
X_train_new[INPUTS_LR_MOD_CLEAN], y_train_new)
```

```
Deviance Residuals:
       Min         1Q     Median         3Q        Max
0 -0.98979 -0.260406 -0.001463   0.247681   0.886481

Coefficients:
               Estimate
Intercept      0.036201
GLUCOSE        1.189027
BODYMASSINDEX  0.673754
PEDIGREEFUNC   0.232235
AGE            0.428759
               Estimate  Std. Err    t-value      Pr(>|t|) Signif
Intercept      0.036201  0.092216   0.392566  6.946399e-01
GLUCOSE        1.189027  0.114440  10.389932  0.000000e+00    ***
BODYMASSINDEX  0.673754  0.101646   6.628463  3.391998e-11    ***
PEDIGREEFUNC   0.232235  0.098748   2.351791  1.868326e-02      *
AGE            0.428759  0.095640   4.483041  7.358692e-06    ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

The fact that we have obtained this result is a positive thing, since even if we have modified the dataset, the results are the same.

To compare these two models we will look at the calibration plot and the ROC curve.

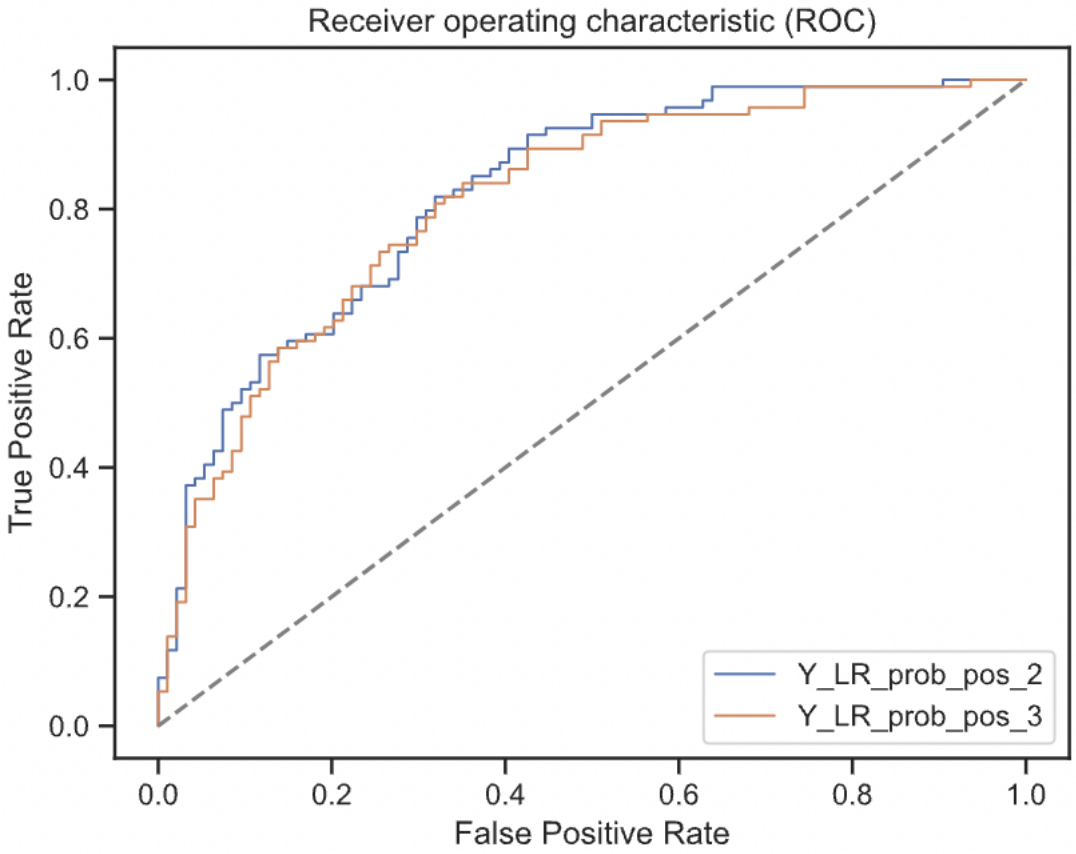
```
CT.calibration_plot(y_test_new,
dfTS_eval_new[['Y_LR_prob_pos_2','Y_LR_prob_pos_3']])
```



Looking at the previous graph we see that the ideal would be that it fits as much as possible to the line, as we see the orange graph fits better than the blue one, being the blue the original one (dataset with all INPUTS) and the orange the graph belonging to the second case (ONLY SIGNIFICANT INPUTS).

```
CT.roc_curve(y_test_new,dfTS_eval_new[['Y_LR_prob_pos_2','Y_LR_p
rob_pos_3']], 1)
```

We see that the ROC has decreased with respect to the two previous models.



We can see that the models we have obtained are not very good because the accuracy is not very good, and the results are very different between test and train. Using other models the results will be better.

SVM

Configuration of the model. Selecting parameters

There are several types of kernels to apply to SVM models. In particular, to test which of them best fits our data, we are going to use three of them:

- Linear
- Polynomial
- Radial

We will apply all of them on the two datasets we have generated, the original one, with all the data, outliers and unbalanced classes; and the clean one, without outliers, eliminating the "SKINTHICKNESS" column and with the balanced classes. We will do this to check if by cleaning the dataset we obtain better results with this model.

We know that the classes in our dataset are not linearly separable, yet we will apply the model with the linear kernel, to check if it is able to separate the classes using sufficient margins to help it classify.

- *TEST WITH ORIGINAL DATASET*
    - **Kernel linear**

      First, we are going to apply a SVM model with linear kernel. To do so, we first define the input and output variables of our model. For this first test, we will use the original dataset with all its columns.

```
INPUTS_ALL =
['PREGNANT','GLUCOSE','BLOODPRESS','SKINTHICKNESS','INSULIN','BODYM
ASSINDEX','PEDIGREEFUNC','AGE']
OUTPUT = 'DIABETES
```

      Once we have chosen the input variables and the dataset we are going to use, we are going to define the parameters of our model. We will use GridSearch to apply several parameters to our model and keep the one that gives us the best results.

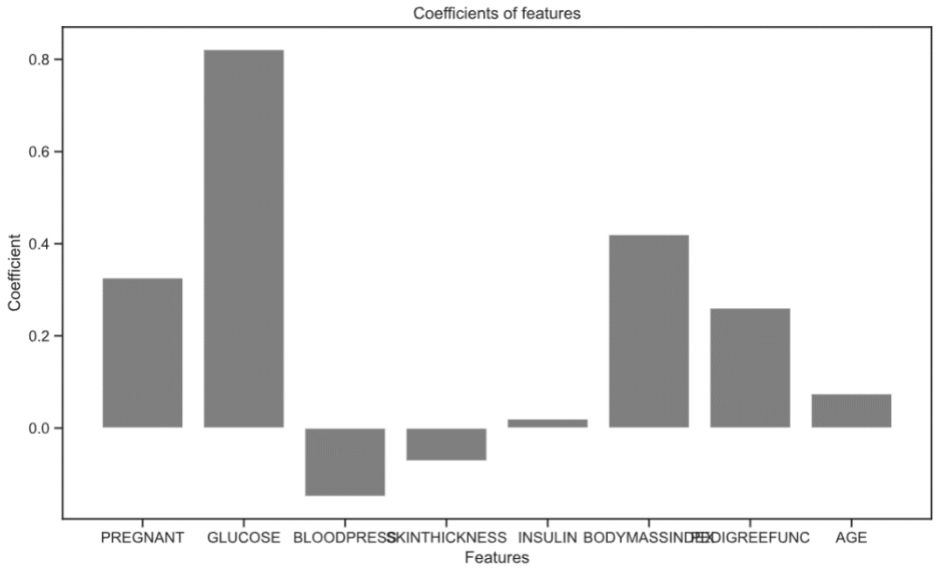
      In this case, let's try the following cost values:

```
[0.00001,0.0001,0.001,0.01,0.1,1,10,100,1000]
```

      With these values we specify the number of points that can be wrong within our predictions. When the value of our cost is small, we are telling the model that there are fewer support vectors that can be used to determine the hyperplane. When the value is larger, we are including more points to determine the hyperplane.

      Checking the accuracy of this model in train, we obtain a value of `0.76708`.

      Once the model has been entered, we can obtain the coefficients of the variables of our dataset, to see to which of them it has given more importance:



As can be seen in the graph, the most important variables for this model are 'GLUCOSE', 'BMI', 'PREGNANT' and 'PEDIGREEFUNC'.

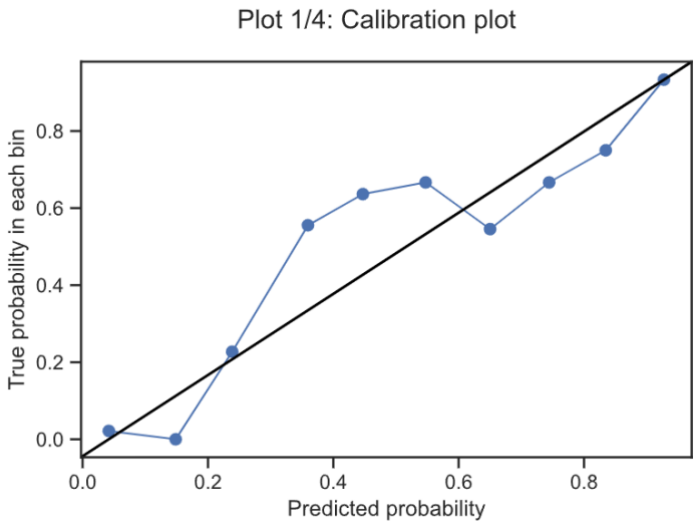
Once these variables have been ascertained, we will generate the confusion matrices for both train and test, to see how much accuracy our model has.

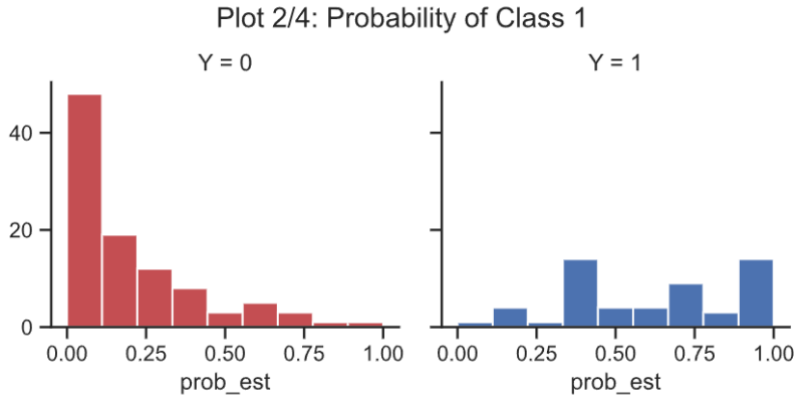
```
----- TRAINING CONFUSION MATRIX -----
Confusion Matrix and Statistics
          Prediction
 Reference   0   1
         0 355  45
         1  89 125

Accuracy: 0.78
No Information Rate: 0.57
P-Value [Acc > NIR]: 0.0
Kappa: 0.5
Mcnemar's Test P-Value: 0.0
Sensitivity: 0.58
Specificity: 0.89
Precision: 0.8
Recall: 0.89
Prevalence: 0.35
Detection Rate: 0.2
Detection prevalence: 0.28
Balanced accuracy: 0.74
F1 Score: 0.84
Positive label: 0
```

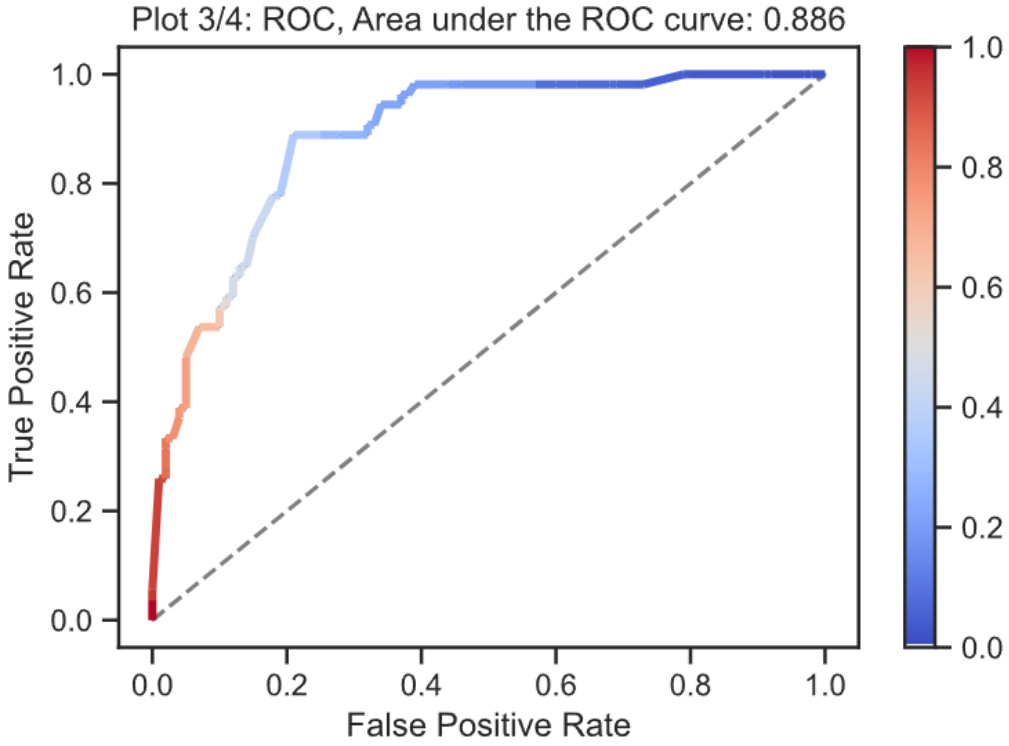
```
----- TESTING CONFUSION MATRIX -----
Confusion Matrix and Statistics
          Prediction
 Reference  0  1
         0 89 11
         1 23 31

Accuracy: 0.78
No Information Rate: 0.57
P-Value [Acc > NIR]: 0.0
Kappa: 0.49
Mcnemar's Test P-Value: 0.06
Sensitivity: 0.57
Specificity: 0.89
Precision: 0.79
Recall: 0.89
Prevalence: 0.35
Detection Rate: 0.2
Detection prevalence: 0.27
Balanced accuracy: 0.73
F1 Score: 0.84
Positive label: 0
```

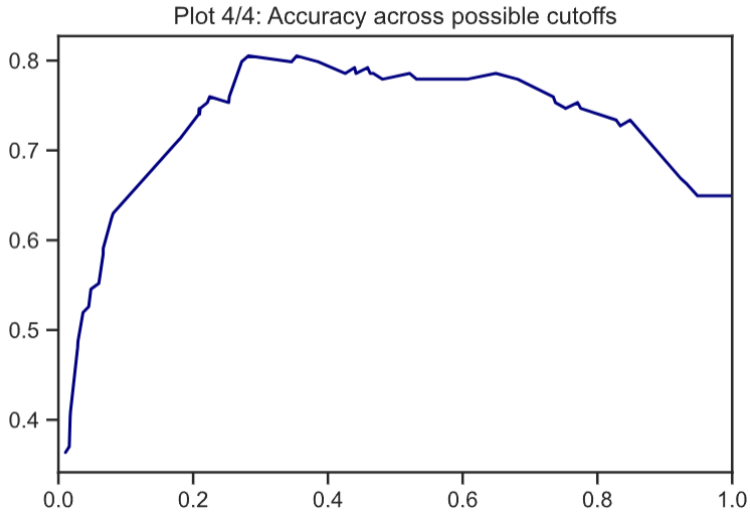
As we can see, we have an accuracy of 0.78 in train and test.

To compare these two models, we will look at the calibration plot and the ROC curve.


Plot 1/4: Calibration plot
True probability in each bin
Predicted probability
0.0
0.2
0.4
0.6
0.8


Plot 2/4: Probability of Class 1
Y = 0
Y = 1
40
20
0
0.00
0.25
0.50
0.75
1.00
prob_est


Plot 3/4: ROC, Area under the ROC curve: 0.886
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0



We can see that the roc curve fits quite well, although looking at the prediction graph, we can see that the model is not fitting quite well.

- Polynomial Kernel

By applying this type of kernel to the model, we can separate classes that are not linearly separable with greater accuracy.

As with the previous kernel, we use the same variables and the same dataset to train our model.

In turn, in this model we must indicate a number of other hyperparameters, so that the model is able to predict correctly.

First, we must indicate the degrees of the polynomial function we want to apply. In our case, we have chosen a function of degree 8, since this is the number of variables we are dealing with in our dataset.

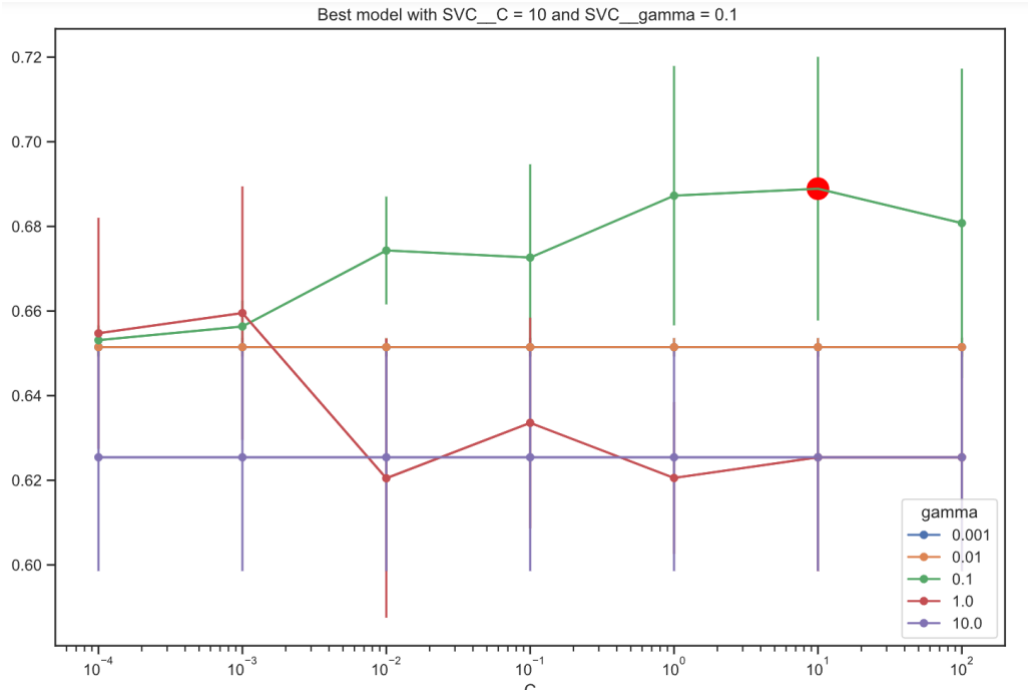
Another hyperparameter that we must indicate is the gamma, which delimits the width of the function, allowing us to see more local effects. In turn, as in the linear kernel, we must indicate the cost of the function, to specify the number of vectors we need to define our hyperplane.

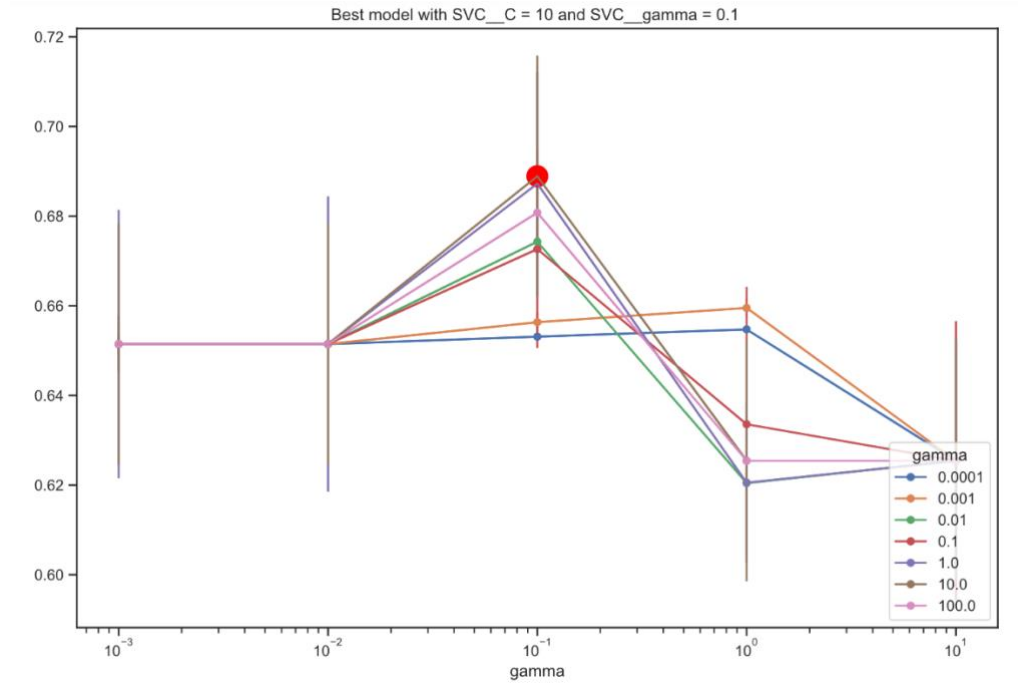
In summary, the parameters we have selected for this model are:

```
C: [0.0001,0.001,0.01,0.1,1,10,100]
Gamma:[0.001,0.01,0.1,1,10]
Degree: 8
```

For this model with these parameters we obtain an accuracy of `0.6922`.

Once the model has been trained, we can see the error plots, and which model has chosen the cross validation method as the best parameters:





We see that the best model is the one with cost 10 and a gamma of 0.1, parameters large enough to encompass a significant number of vectors defining the hyperplane.

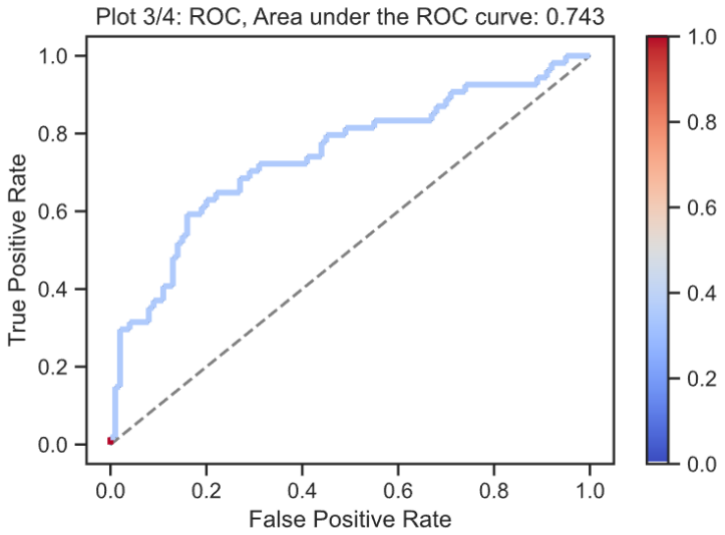
Once the best model has been selected, we can generate confusion matrices to check the accuracy of the model on the train and test sets.

```
----- TRAINING CONFUSION MATRIX -----
Confusion Matrix and Statistics
          Prediction
 Reference   0   1
         0 400   0
         1 107 107

Accuracy: 0.83
No Information Rate: 0.6
P-Value [Acc > NIR]: 0.0
Kappa: 0.57
Mcnemar's Test P-Value: 0.0
Sensitivity: 0.5
Specificity: 1.0
Precision: 0.79
Recall: 1.0
Prevalence: 0.35
Detection Rate: 0.17
Detection prevalence: 0.17
Balanced accuracy: 0.75
F1 Score: 0.88
Positive label: 0
```

```
----- TEST CONFUSION MATRIX-----
Confusion Matrix and Statistics
          Prediction
 Reference  0  1
         0 97  3
         1 38 16

Accuracy: 0.73
No Information Rate: 0.61
P-Value [Acc > NIR]: 0.02
Kappa: 0.31
Mcnemar's Test P-Value: 0.0
Sensitivity: 0.3
Specificity: 0.97
Precision: 0.72
Recall: 0.97
Prevalence: 0.35
Detection Rate: 0.1
Detection prevalence: 0.12
Balanced accuracy: 0.63
F1 Score: 0.83
Positive label: 0
```

We can see that for our model, we obtain an accuracy of 0.83 in train and 0.73 in test.

We now move on to paint the roc curve of this model. It can be observed that it does not give very good results:



- **Radial Kernel**

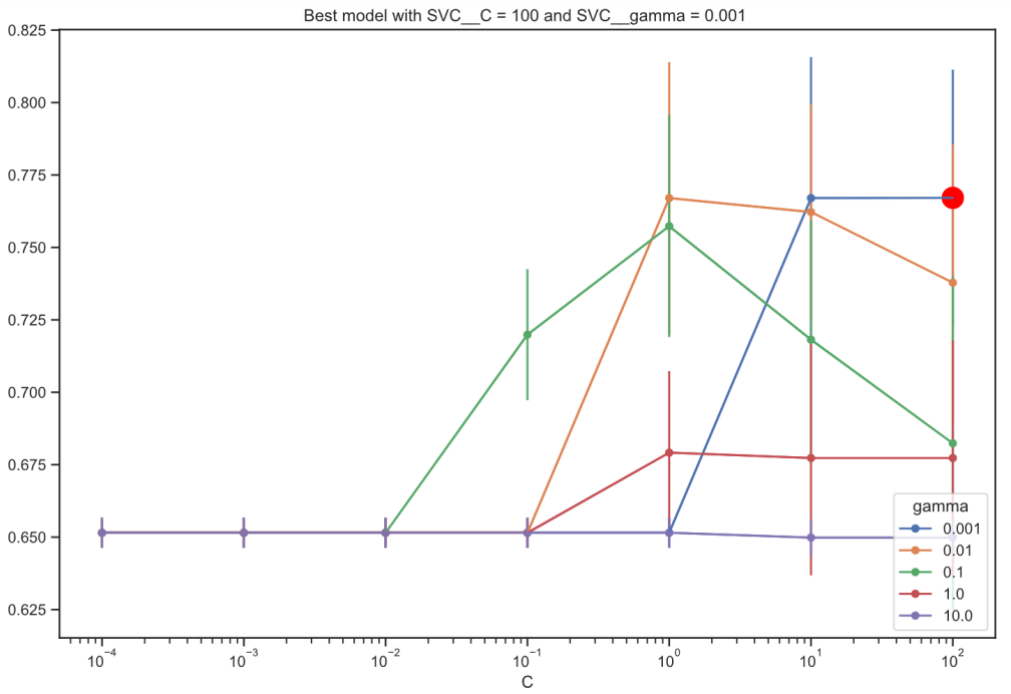
Finally, we are going to configure the parameters of the same model, training it with the same data, but this time using a linear kernel. In this case, it is not necessary to indicate the number of degrees, since what this model will do is to separate the classes by a 'circular' hyperplane.

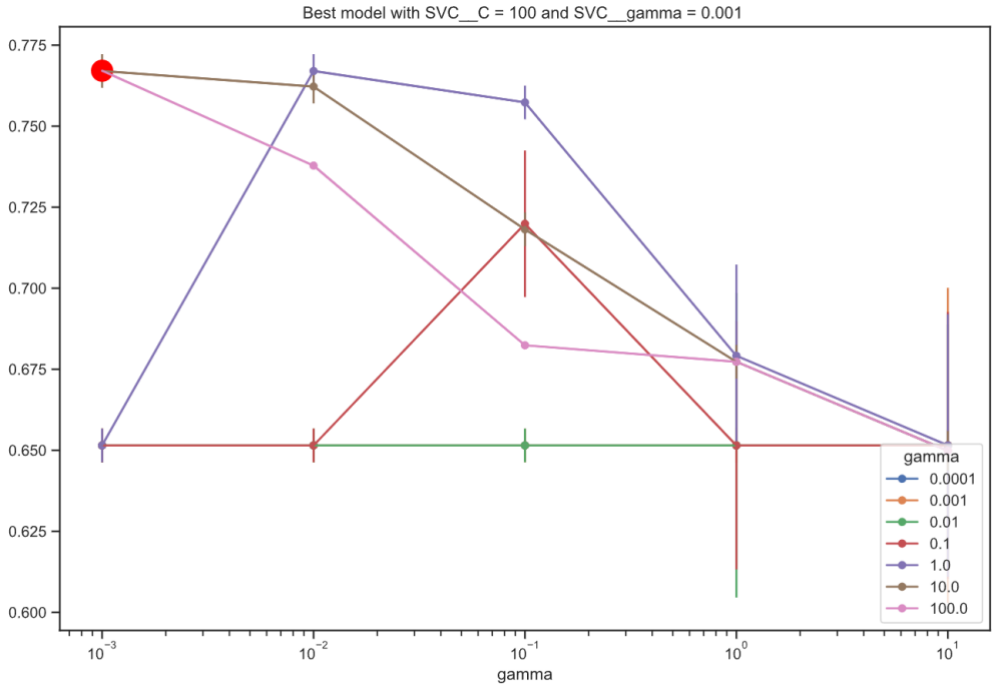
The parameters we have selected for this model are:

```
C: [0.0001,0.001,0.01,0.1,1,10,100]
Gamma:[0.001,0.01,0.1,1,10]
```

As in the previous models, we will perform a grid search to perform a cross validation and select the parameters with which our model best fits. We can see that the model gives us an accuracy of `0.7622`.

Once the model has been trained, we can see the error plots, and which model has chosen the cross-validation method as the best parameters.





We can see that the cross validation has chosen as best model the one with a cost of 100 and a gamma of 0.01, something different from our polynomial model.

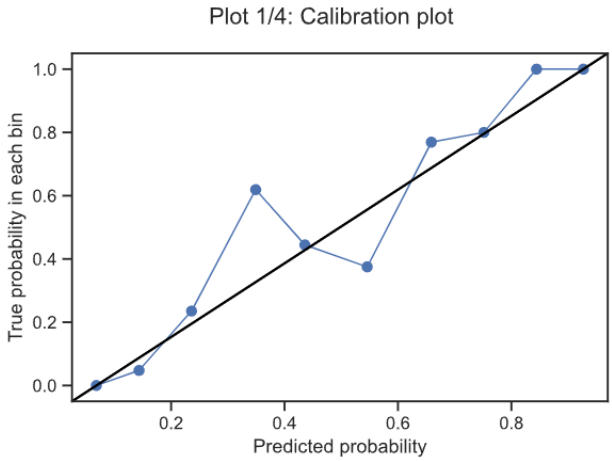
Once the best model has been selected, we can generate confusion matrices to check the accuracy of the model on the train and test sets.

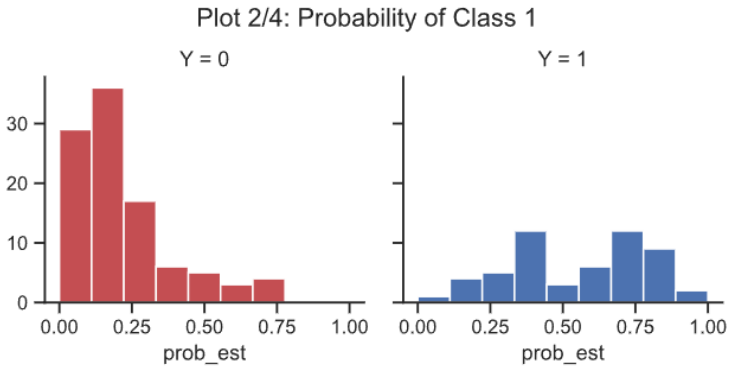
```
----- TRAINING CONFUSION MATRIX -----
Confusion Matrix and Statistics
          Prediction
 Reference   0   1
         0 366  34
         1 101 113

Accuracy: 0.78
No Information Rate: 0.58
P-Value [Acc > NIR]: 0.0
Kappa: 0.48
Mcnemar's Test P-Value: 0.0
Sensitivity: 0.53
Specificity: 0.92
Precision: 0.78
Recall: 0.92
Prevalence: 0.35
Detection Rate: 0.18
Detection prevalence: 0.24
Balanced accuracy: 0.72
F1 Score: 0.84
Positive label: 0
```

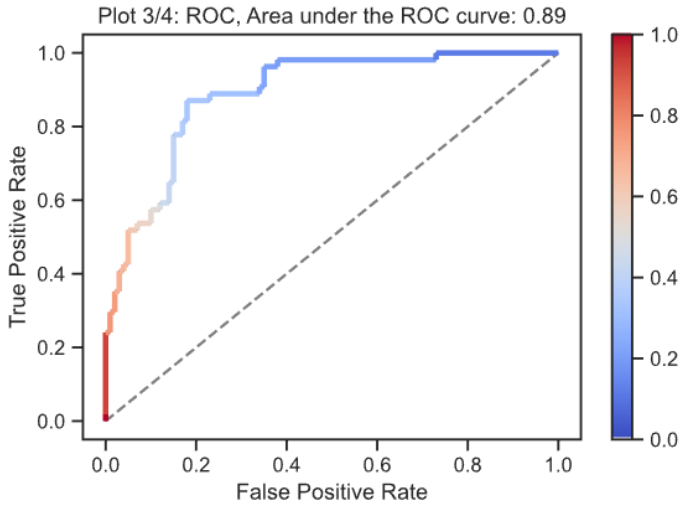
```
----- TEST CONFUSION MATRIX -----
Confusion Matrix and Statistics
          Prediction
 Reference  0  1
         0 90 10
         1 23 31

Accuracy: 0.79
No Information Rate: 0.57
P-Value [Acc > NIR]: 0.0
Kappa: 0.5
Mcnemar's Test P-Value: 0.04
Sensitivity: 0.57
Specificity: 0.9
Precision: 0.8
Recall: 0.9
Prevalence: 0.35
Detection Rate: 0.2
Detection prevalence: 0.27
Balanced accuracy: 0.74
F1 Score: 0.85
Positive label: 0
```

We can see that for our model, we obtain an accuracy of 0.78 in train and 0.79 in test.

Now, we will look at the calibration plot and the ROC curve.







We can see from the area under the roc curve that with this model and radial kernel pretty good results are obtained.

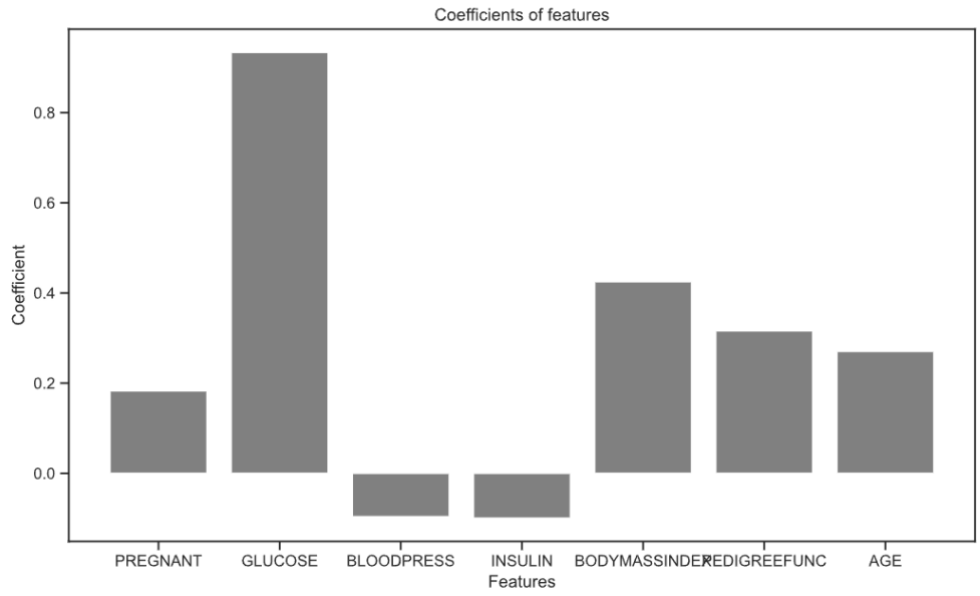
- *TEST WITH MODIFICATED DATASET*
  We are going to use exactly the same parameters described before for all the models, the only thing that will change now is that instead of using the original dataset, we are going to apply the models on the clean dataset, without outliers, with balanced classes, and excluding the 'SKINTHICKNESS' column which, as we have already mentioned, does not give us good results since most of the data are erroneous. The variables we are going to use are:

```
INPUTS_ALL = ['PREGNANT','GLUCOSE','BLOODPRESS'
,'INSULIN','BODYMASSINDEX','PEDIGREEFUNC','AGE']
OUTPUT = 'DIABETES
```

  - Linear kernel

    Using exactly the same parameters as with the original dataset, we obtain the following results:

    - Accuracy of `0.7581`.
    - Importance of the variables: In the graph we can see that now our model gives more importance to all the variables we have in the dataset, as we have eliminated the column 'SKINTHICKNESS', we have less noise in the data and we can be more accurate with our model.



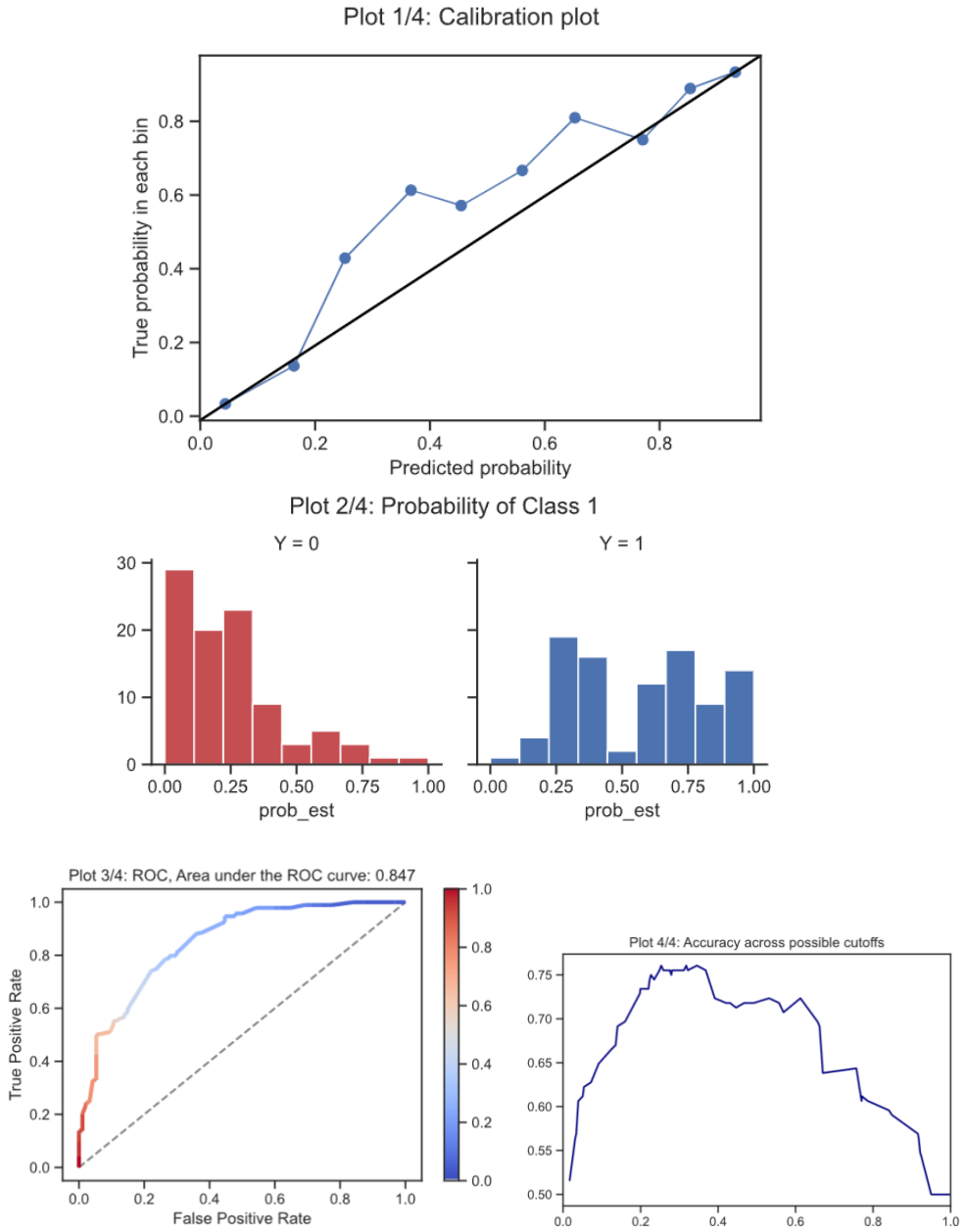
- Confusion matrices:

```
----- TRAINING CONFUSION MATRIX -----
Confusion Matrix and Statistics
          Prediction
 Reference   0   1
         0 336  38
         1 158 216

Accuracy: 0.74
No Information Rate: 0.5
P-Value [Acc > NIR]: 0.0
Kappa: 0.48
Mcnemar's Test P-Value: 0.0
Sensitivity: 0.58
Specificity: 0.9
Precision: 0.68
Recall: 0.9
Prevalence: 0.5
Detection Rate: 0.29
Detection prevalence: 0.34
Balanced accuracy: 0.74
F1 Score: 0.77
Positive label: 0
```

```
----- TESTING CONFUSION MATRIX -----
Confusion Matrix and Statistics
          Prediction
 Reference  0  1
         0 83 11
         1 42 52

Accuracy: 0.72
No Information Rate: 0.5
P-Value [Acc > NIR]: 0.0
Kappa: 0.44
Mcnemar's Test P-Value: 0.0
Sensitivity: 0.55
Specificity: 0.88
Precision: 0.66
Recall: 0.88
Prevalence: 0.5
Detection Rate: 0.28
Detection prevalence: 0.34
Balanced accuracy: 0.72
F1 Score: 0.76
Positive label: 0
```

As we can see in the confussion matrices, we have a 0.74 of accuracy in train and 0.72 in test.

- The calibration plot and the ROC curve: We can see that the area under the curve and the results shown in the graphs in general are better than the ROC curve.

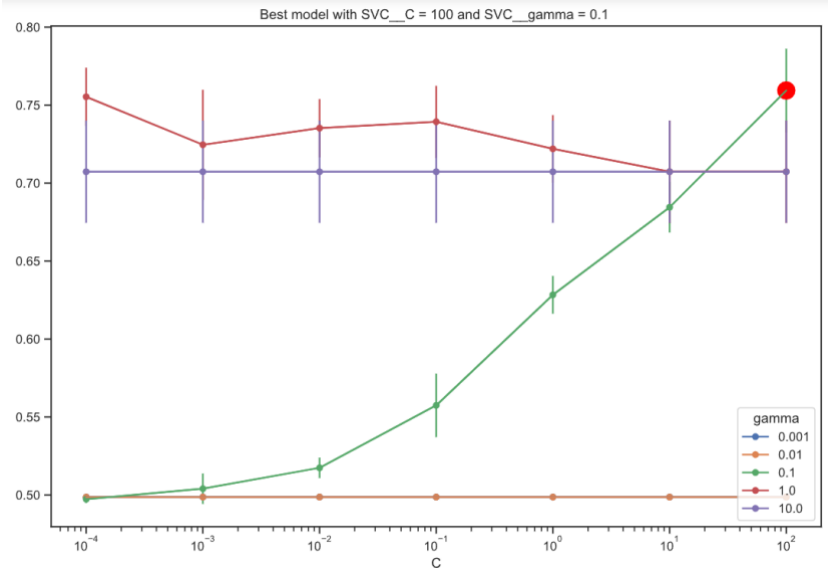


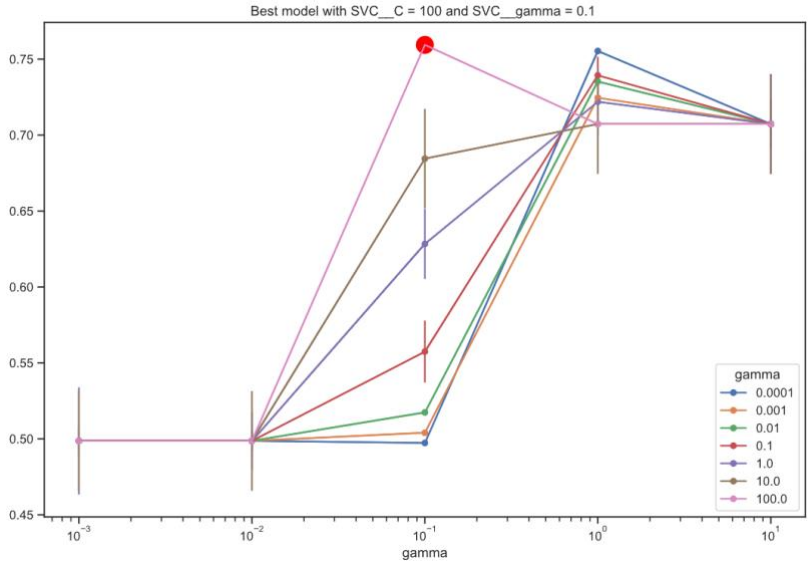
Both the area under the curve and the predictions of this model are quite tight, it is a good model.

- Polynomial Kernel

  Using exactly the same parameters as with the original dataset, only changing the degree of the polynomial from 8 to 7 (number of variables that our dataset now has after having eliminated the column), we obtain the following results:

  - Accuracy of `0.7768`.
  - Error plots: Running a cross validation with 5 folds, we obtain that the best model is the one using a cost of 100 and a gamma of 0.1, large enough parameters to obtain a wider band.





- Confusion matrices:

```
----- TRAINING CONFUSION MATRIX -----
Confusion Matrix and Statistics
          Prediction
 Reference   0   1
         0 327  47
         1  23 351

Accuracy: 0.91
No Information Rate: 0.5
P-Value [Acc > NIR]: 0.0
Kappa: 0.81
Mcnemar's Test P-Value: 0.01
Sensitivity: 0.94
Specificity: 0.87
Precision: 0.93
Recall: 0.87
Prevalence: 0.5
Detection Rate: 0.47
Detection prevalence: 0.53
Balanced accuracy: 0.91
F1 Score: 0.9
Positive label: 0
```

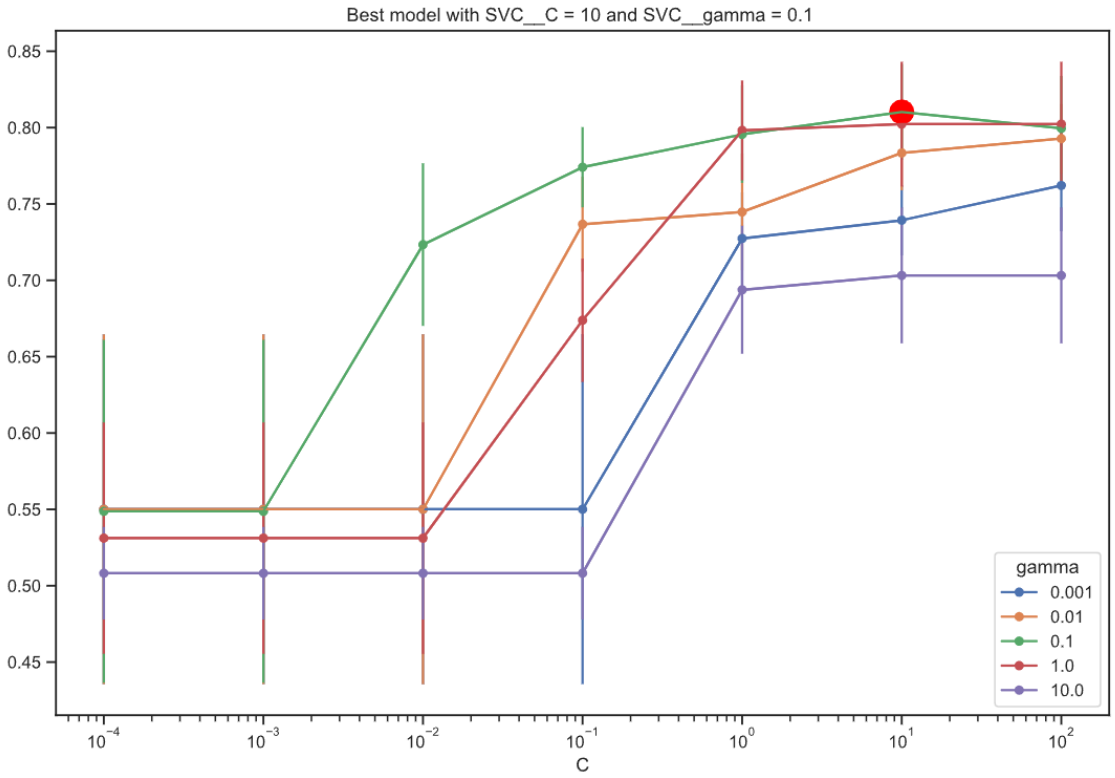
```
----- TESTING CONFUSION MATRIX -----
Confusion Matrix and Statistics
          Prediction
 Reference  0  1
         0 73 21
         1 15 79

Accuracy: 0.81
No Information Rate: 0.5
P-Value [Acc > NIR]: 0.0
Kappa: 0.62
Mcnemar's Test P-Value: 0.41
Sensitivity: 0.84
Specificity: 0.78
Precision: 0.83
Recall: 0.78
Prevalence: 0.5
Detection Rate: 0.42
Detection prevalence: 0.53
Balanced accuracy: 0.81
F1 Score: 0.8
Positive label: 0
```

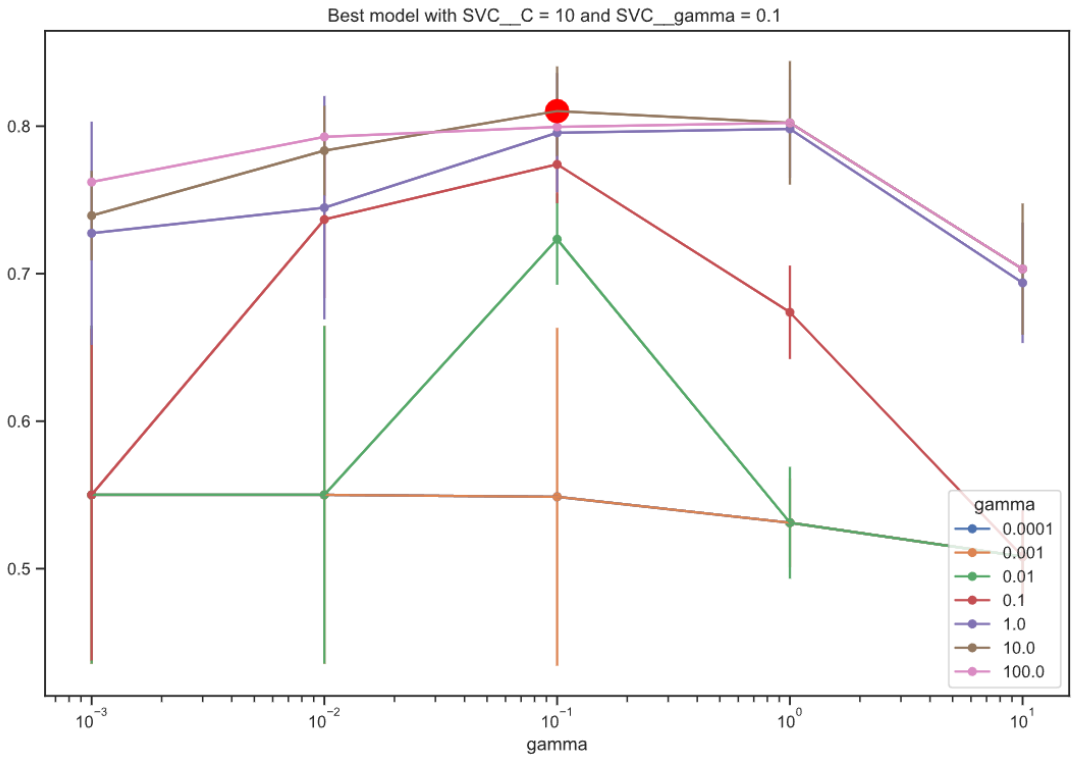
As we can see, with this model we are getting a 0.91 of accuracy on train and a 0.81 on test.

- Radial kernel

  Using exactly the same parameters as with the original dataset, we obtain the following results:

  - Accuracy of `0.8062`.
  - Error graph: It indicates that the best model for this kernel is the one with cost 10 and gamma 0.1.





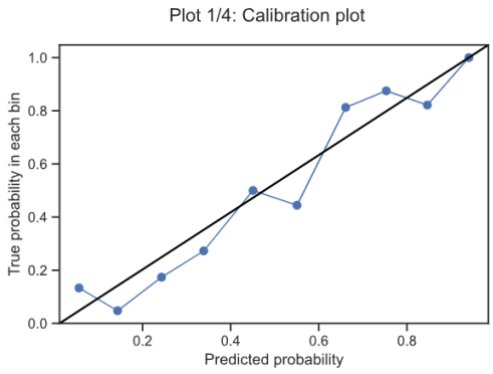
- Confusion matrices:

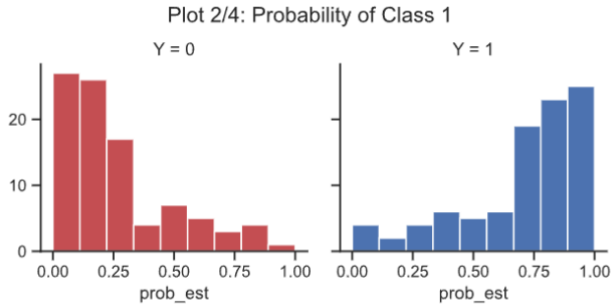
```
----- TRAINING CONFUSION MATRIX -----
Confusion Matrix and Statistics
          Prediction
 Reference   0   1
         0 325  49
         1  31 343

Accuracy: 0.89
No Information Rate: 0.5
P-Value [Acc > NIR]: 0.0
Kappa: 0.79
Mcnemar's Test P-Value: 0.06
Sensitivity: 0.92
Specificity: 0.87
Precision: 0.91
Recall: 0.87
Prevalence: 0.5
Detection Rate: 0.46
Detection prevalence: 0.52
Balanced accuracy: 0.89
F1 Score: 0.89
Positive label: 0
```

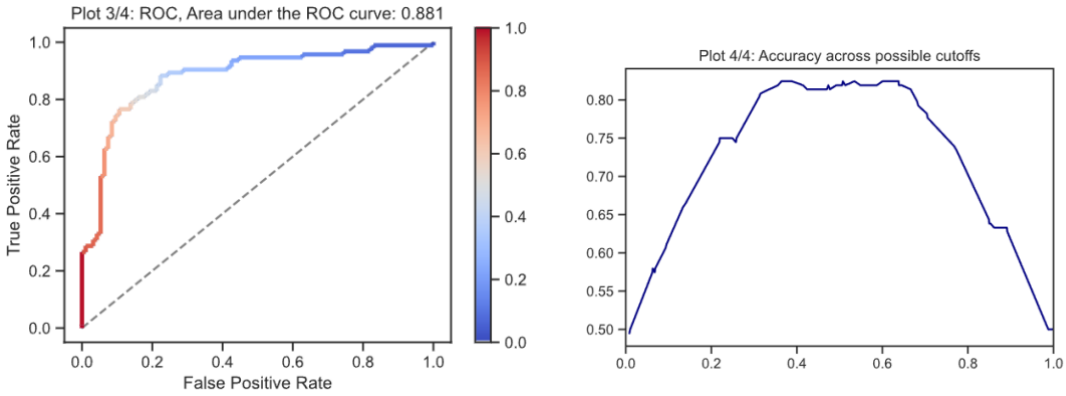
```
----- TEST CONFUSION MATRIX -----
Confusion Matrix and Statistics
          Prediction
 Reference  0  1
         0 79 15
         1 18 76

Accuracy: 0.82
No Information Rate: 0.5
P-Value [Acc > NIR]: 0.0
Kappa: 0.65
Mcnemar's Test P-Value: 0.73
Sensitivity: 0.81
Specificity: 0.84
Precision: 0.81
Recall: 0.84
Prevalence: 0.5
Detection Rate: 0.4
Detection prevalence: 0.48
Balanced accuracy: 0.82
F1 Score: 0.83
Positive label: 0
```

As we can see, we are getting an accuracy of 0.89 on train and a 0.82 on test.

- The calibration plot and the ROC curve: We can see that the model fits the data quite well, and that the area under the ROC curve is quite large. This is the best model





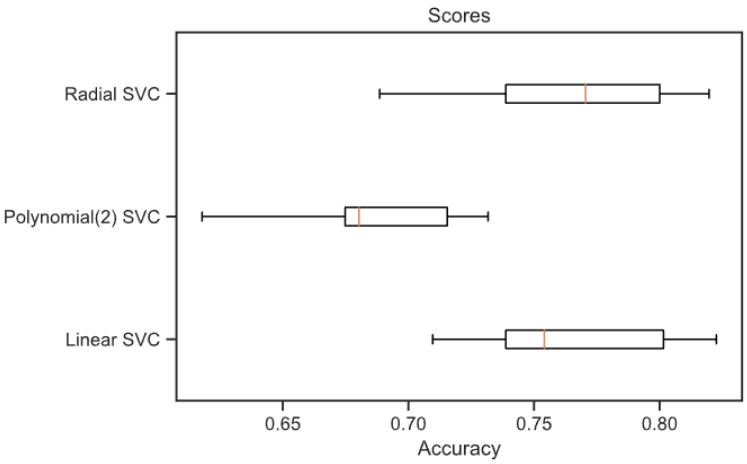


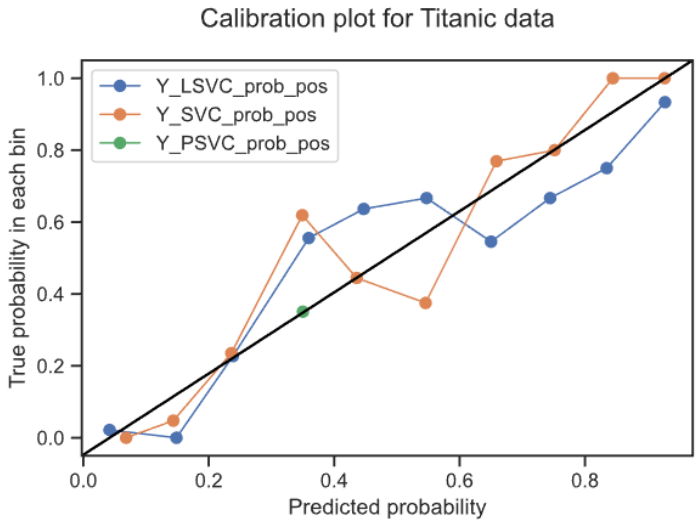
The area under the curve and the predictions are really good, we could say this is the best approach for all models of SVC that we have tested.

Results and conclussions

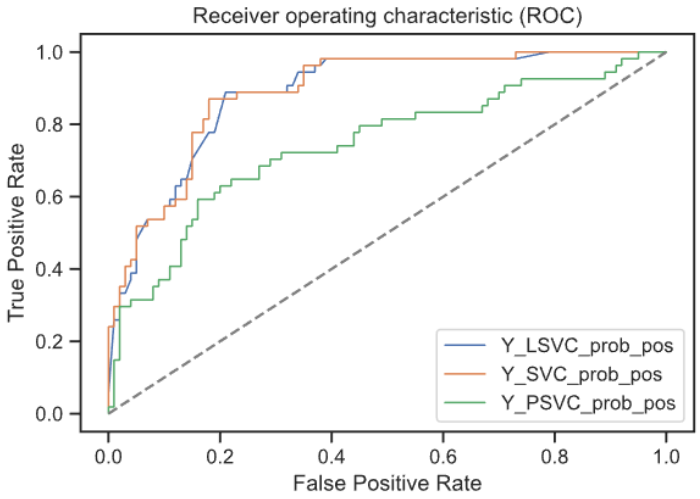
Once all the results have been obtained, we can compare the models according to the dataset on which we have trained them, to see which one gives a better result depending on the dataset.

- *Comparison of the results for the original dataset*



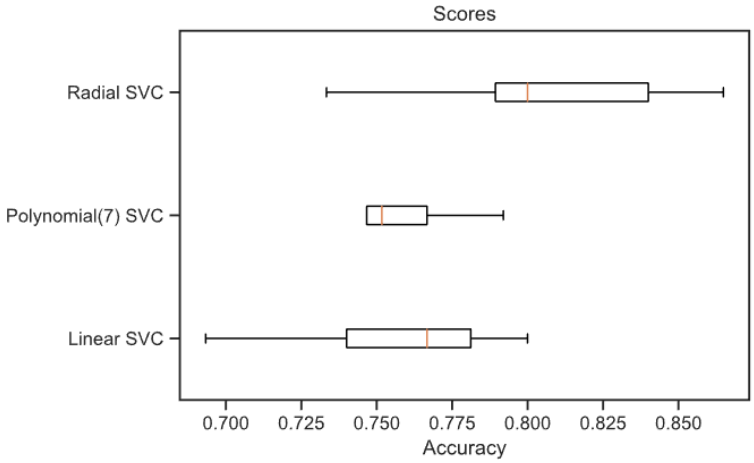


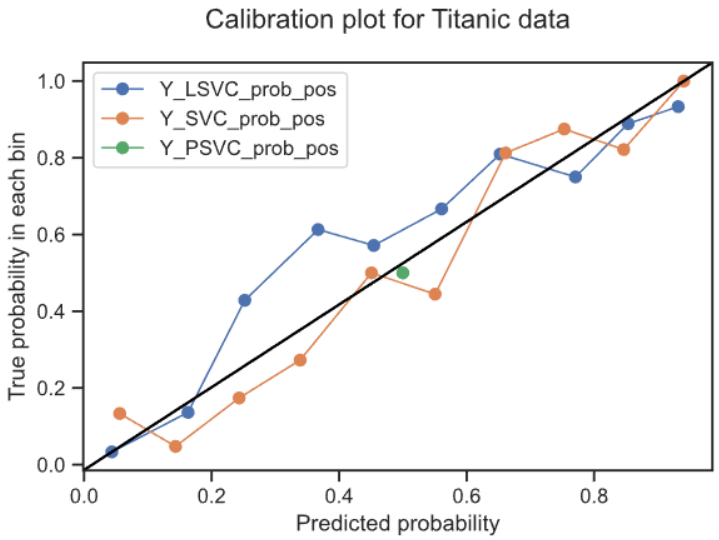

```
Area under the ROC curve of Y_LSVC_prob_pos : 0.886
Area under the ROC curve of Y_SVC_prob_pos : 0.89
Area under the ROC curve of Y_PSVC_prob_pos : 0.743
```



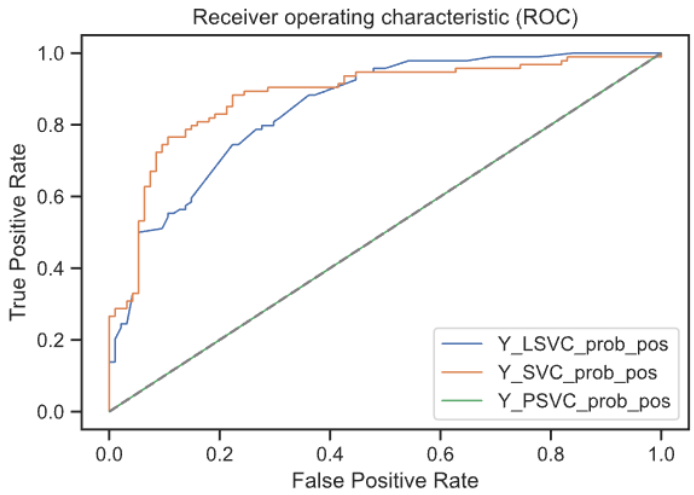
Comparing these three models, we can see that the best fit to our data is the radial model, followed by the linear model with a fairly wide margin, while the polynomial model lags behind and is not able to correctly predict the data.

- *Comparison of the results for the clean dataset*






```
Area under the ROC curve of Y_LSVC_prob_pos : 0.847
Area under the ROC curve of Y_SVC_prob_pos : 0.881
Area under the ROC curve of Y_PSVC_prob_pos : 0.5
```



Similarly, for the clean data set, we can observe the same trend. The polynomial model still does not fit the data well. While the radial model and the linear model fit quite well, giving good enough results.

KNN

The kNN model uses information about the k-nearest neighbors to classify the data. It is a non-parametric supervised learning classifier.

If k is too small, you may be overtraining the model, and therefore, it would not give good results in the test part. A midpoint must be found to obtain good results both in train and in test, that is, that the model is generalized.

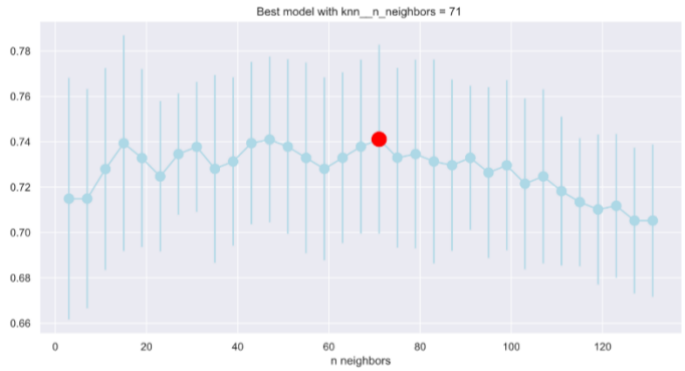
The K-Nearest Neighbors model has been trained 3 times:

- **TEST WITH ORIGINAL DATASET**: with the original Data Frame, 'diabetes'
- **TEST WITH MODIFIED DATASET**: with the new Data Frame, 'diabetes_new'
- **TEST WITH MOST IMPORTANT VARIABLES**: with the new Data Frame 'diabetes_new' but with the variables GLUCOSE, PEDIGREEFUNC, BODYMASSINDEX and AGE.

Now we will explain each of the trained models, with their respective results.

- **TEST WITH ORIGINAL DATASET**

This first model has been trained with all the data of the pure dataset (diabetes), and with all variables (PREGNANT, GLUCOSE, BLOODPREDD, SKINTHICKNESS, INSULIN, BODYMASSINDEX, PEDIGREEFUNC and AGE).



In this model with the original dataset, it has been detected that the optimum point is obtained with k neighbors equal to 71.

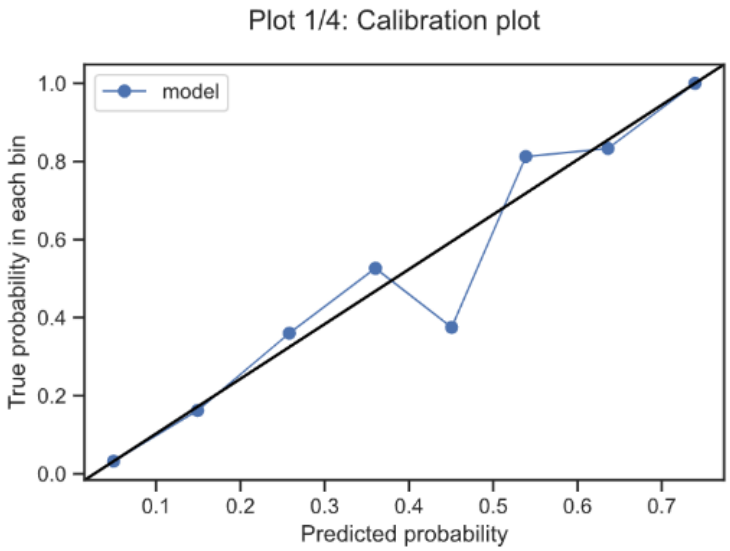
The smaller the number of k neighbors, the more complex the model. In this case, with k equal to 71, we are dealing with a model that is not really complicated. The higher the accuracy, and the less complex the model, the better.

Train part Test part

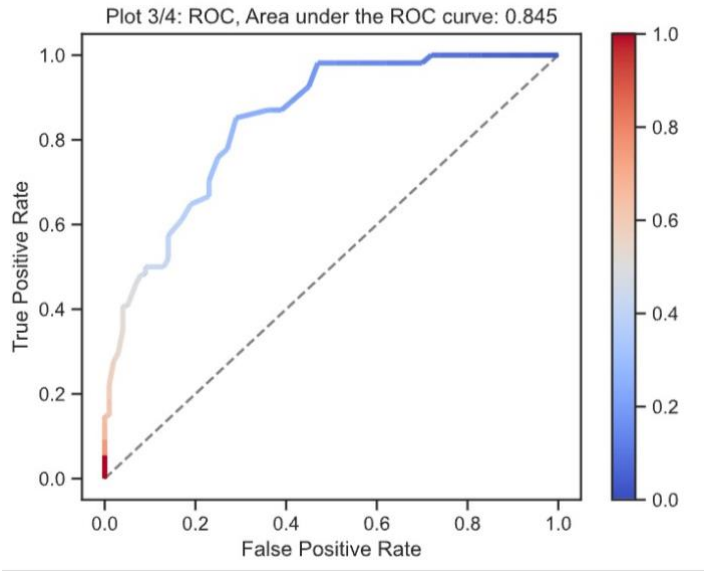
```
Confusion Matrix and Statistics
          Prediction
 Reference   1   0
         1  75 139
         0  24 376

Accuracy: 0.73
No Information Rate: 0.6
P-Value [Acc > NIR]: 0.0
Kappa: 0.33
Mcnemar's Test P-Value: 0.0
Sensitivity: 0.94
Specificity: 0.35
Pos pred value: 0.73
Neg pred value: 0.76
Prevalence: 0.65
Detection Rate: 0.61
Detection prevalence: 0.84
Balanced accuracy: 0.65
F Score: 0.82
Positive class: 0
```

```
Confusion Matrix and Statistics
          Prediction
 Reference  0  1
         0 96  4
         1 32 22

Accuracy: 0.77
No Information Rate: 0.6
P-Value [Acc > NIR]: 0.0
Kappa: 0.42
Mcnemar's Test P-Value: 0.0
Sensitivity: 0.41
Specificity: 0.96
Pos pred value: 0.85
Neg pred value: 0.75
Prevalence: 0.35
Detection Rate: 0.14
Detection prevalence: 0.17
Balanced accuracy: 0.68
F Score: 0.55
Positive class: 1
```

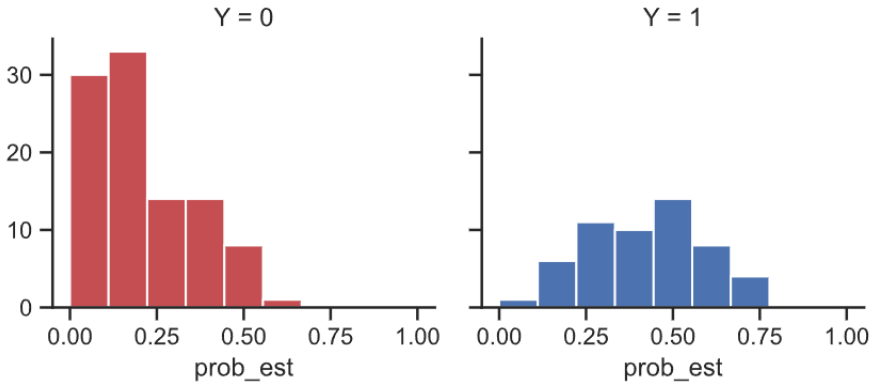
We obtained an accuracy of 0.73 in train, and 0.77 in test. The model is not overtrained, since the values of train and test are quite close. Also, the cross-Validation accuracy es 0.725.



Observing the previous graph and considering that the ideal case is that the line fits perfectly with the line that crosses the diagonal, we can see that in this case it does not fit well.



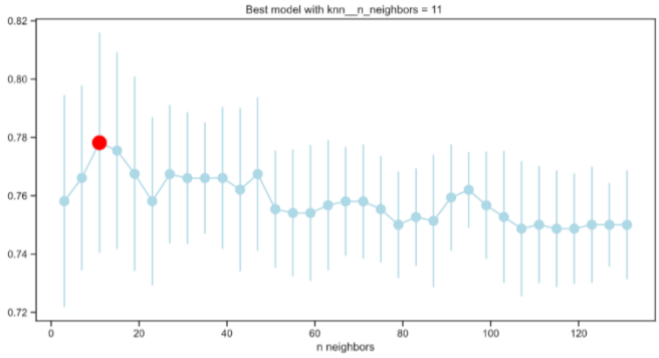
The ROC curve of this variable has an area of 0.845, considering that the ideal case would be in which the curve reaches the corner, and that the area is quite large, it can be considered a good ROC curve.



The sensitivity is at 0.41, this means that it is hitting 41% of all the positives. Bearing in mind that the ideal case is for the histogram to be tilted sideways, in this case we can see that this is not the case. It is above all in DIABETES=1. That is, the model does not perform a good classification.

- *TEST WITH MODIFIED DATASET*

This second model has been trained with all the data of the modificated dataset (diabetes_new), and with all variables (PREGNANT, GLUCOSE, BLOODPREDD, INSULIN, BODYMASSINDEX, PEDIGREEFUNC and AGE).



In this model with modificated dataset, it has been detected that the optimum point is obtained with k neighbors equal to 11. In this case we are dealing with a complex model, because k is too small.

Train part

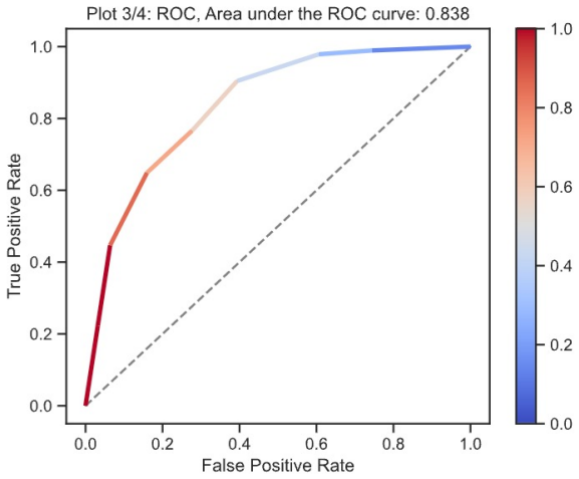
```
Confusion Matrix and Statistics
          Prediction
 Reference   0   1
         0 283  91
         1  50 324

Accuracy: 0.81
No Information Rate: 0.5
P-Value [Acc > NIR]: 0.0
Kappa: 0.62
Mcnemar's Test P-Value: 0.0
Sensitivity: 0.87
Specificity: 0.76
Pos pred value: 0.78
Neg pred value: 0.85
Prevalence: 0.5
Detection Rate: 0.43
Detection prevalence: 0.55
Balanced accuracy: 0.81
F Score: 0.82
Positive class: 1
```

Test part

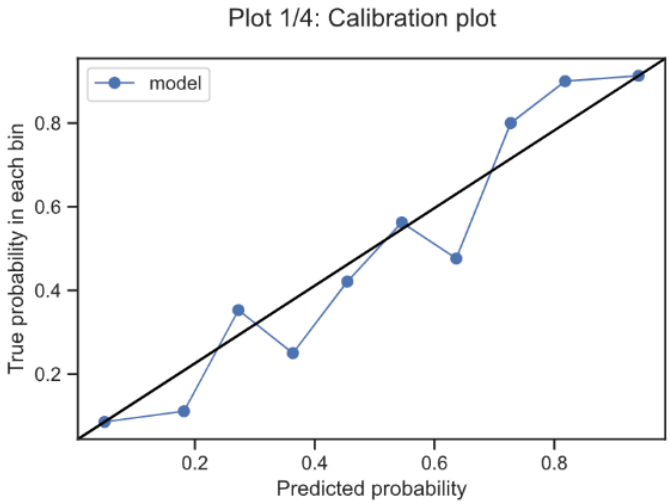
```
Confusion Matrix and Statistics
          Prediction
 Reference  0  1
         0 68 26
         1 20 74

Accuracy: 0.76
No Information Rate: 0.5
P-Value [Acc > NIR]: 0.0
Kappa: 0.51
Mcnemar's Test P-Value: 0.46
Sensitivity: 0.79
Specificity: 0.72
Pos pred value: 0.74
Neg pred value: 0.77
Prevalence: 0.5
Detection Rate: 0.39
Detection prevalence: 0.53
Balanced accuracy: 0.76
F Score: 0.76
Positive class: 1
```

We obtained an accuracy of 0.81 in train, and 0.76 in test. The model is a bit overtrained, and we can see the difference between the accuracy between the train and test. On the other hand, the test accuracy (0.79) is a high value, but if the model is overtrained, it is not a very good model.

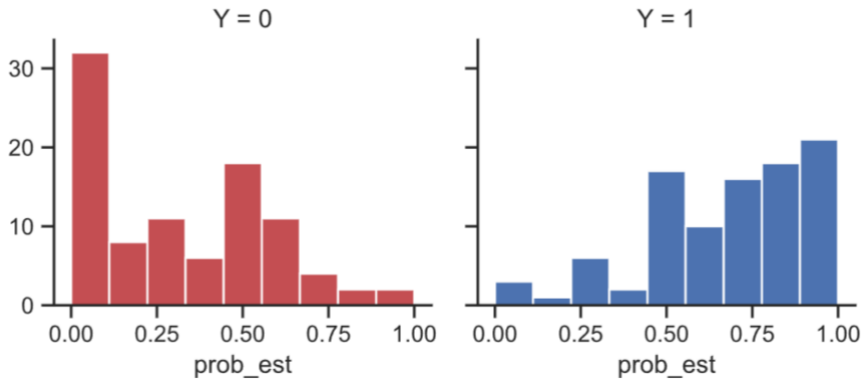
The cross-validation score in this model is 0.743



In this case, the ROC curve is quite like the previous case (original dataset), but the area under the curve (0.838) is a little smaller, so it is a slightly worse curve than the previous one.



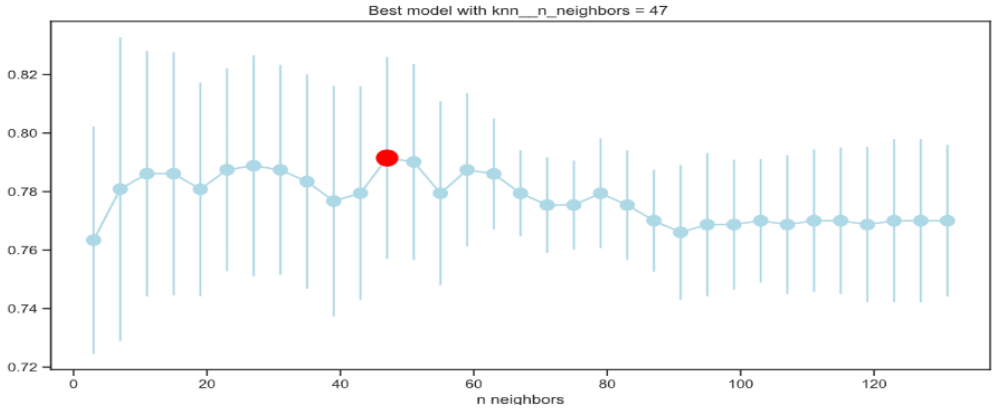
As we can see in the calibration plot, this model does not fit very well to the diagonal straight line, so as we said before, it is not a very good model.



The sensitivity is at 0.79, that means that it is hitting 79% of all positives. In this case, it is failing to match some values of DIABETES_0 and DIABETES=1 but having such a high sensitivity means that it is not doing anything wrong.

- ***TEST WITH MOST IMPORTANT VARIABLES***

This third model has been trained with some important variables of modificated dataset (diabetes_new). These important variables are GLUCOSE, PEDIGREEFUNC, BODYMASSINDEX and AGE.



In this model with the most important variables of the modificated dataset (diabetes_new), it has been detected that the optimum point is obtained with k neighbors equal to 47. In this case, we are dealing with a not very complicated model, and this is going to be good for the test part.

Train part

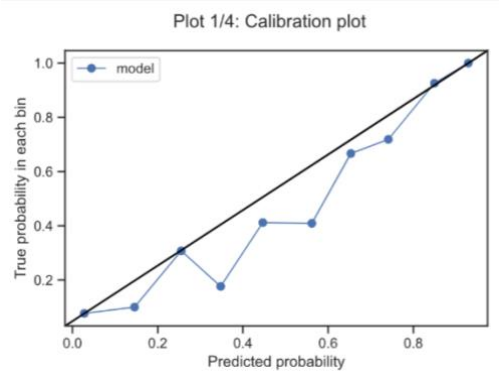
```
Confusion Matrix and Statistics
          Prediction
 Reference   0   1
         0 273 101
         1  52 322

Accuracy: 0.8
No Information Rate: 0.5
P-Value [Acc > NIR]: 0.0
Kappa: 0.59
Mcnemar's Test P-Value: 0.0
Sensitivity: 0.86
Specificity: 0.73
Pos pred value: 0.76
Neg pred value: 0.84
Prevalence: 0.5
Detection Rate: 0.43
Detection prevalence: 0.57
Balanced accuracy: 0.8
F Score: 0.81
Positive class: 1
```

Test part

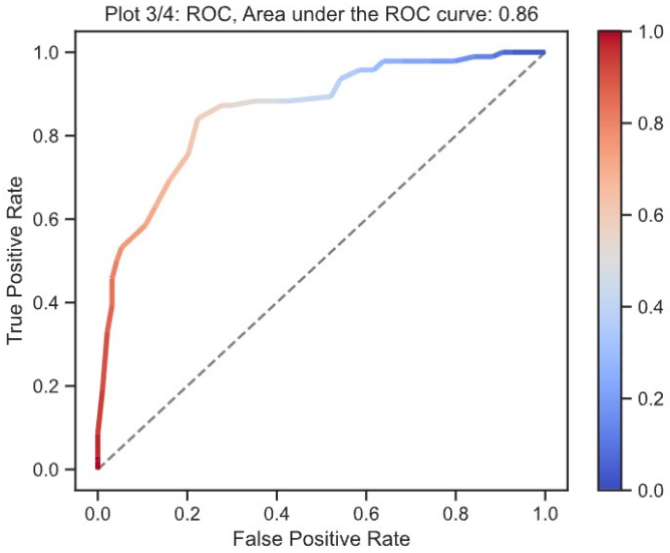
```
Confusion Matrix and Statistics
          Prediction
 Reference  0  1
         0 63 31
         1 17 77

Accuracy: 0.74
No Information Rate: 0.5
P-Value [Acc > NIR]: 0.0
Kappa: 0.49
Mcnemar's Test P-Value: 0.06
Sensitivity: 0.82
Specificity: 0.67
Pos pred value: 0.71
Neg pred value: 0.79
Prevalence: 0.5
Detection Rate: 0.41
Detection prevalence: 0.57
Balanced accuracy: 0.74
F Score: 0.76
Positive class: 1
```

We obtained an accuracy of 0.8 in train, and 0.74 in test. The values seem to be good metrics, but test accuracy is not a high value (0.74).

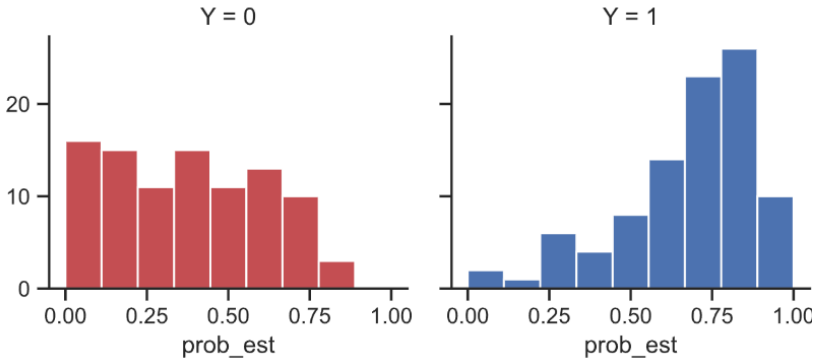
The cross-validation score here is 0.771.



This model does not fit very well to the diagonal straight line, and as we talk before, this is not good.



In this last model, the area under the ROC curve is the largest of the 3 models, which would help us to deduce that it is a good model, it is almost 0.9, with which we conclude that it is a good ROC curve.



The sensitivity is at 0.82, that means that it is hitting 82% of all positives. In this case it is failing to classify values above all of DIABETES=0, ideally, we would give the class of 0 very low probabilities, and the class of 1 very high probability.

CONCLUSION OF KNN

Original data set:

- k neighbors equal to 71
- 0.73 in train
- 0.77 in test
- accuracy in cross-validation of 0.725
- Sensibility 0.41

Modificated dataset:

- k neighbors equal to 11
- 0.81 in train
- 0.76 in test
- accuracy in cross-validation of 0.743
- Sensibility 0.79

Modificated dataset, but with the most significant variables

- k neighbors equal to 47
- 0.8 in train
- 0.74 in test
- accuracy in cross-validation of 0.771
- Sensibility 0.82

In this case it is a bit difficult to choose which would be the best model since each one has its pros and cons. The one that offers the highest accuracy in the test is the first (original data set), but it has a sensitivity of 41%, which is quite low.

On the other hand, the second model (modified data set) is a bit complicated model, since the lower the number of k, the more complicated the model will be, and 11 is quite a small number compared to the other models. This is reflected in the fact that there is a bit of overfitting in the accuracy of train and test. However, the test accuracy is the second highest compared to the other two models (0.76). In addition, it explains 79% of the data correctly, which is quite a high value.

As for the third model, there is overfitting between train and test, we can see this in the difference in the accuracy of both. But the test accuracy is a good value (0.74) and considering that it is a flexible model (it is not complex) because the number of k neighbors is large (47), I have decided that this model is the best. In addition, to correctly explain 82% of the data.

In conclusion, we have decided that **the third model (modified dataset, but with the most significant variables) is the best** compared to the other two.

Multi-Layer Perceptron (MLP)

Configuration of the model. Selecting parameters

For this model, as in all the previous ones, we are going to test a series of parameters, in this case, layers of neurons and moment penalty (Alpha), so that, by means of grid search with cross validation, we are able to identify the best parameters for our model.

In this case, we have also trained the model with the two datasets we have available, the original and the clean one, in order to compare the results of both and to see how the models behave with the different datasets.

- *TEST WITH ORIGINAL DATASET*

To train the model with the original dataset, we are going to configure the input and output variables as shown in the table below

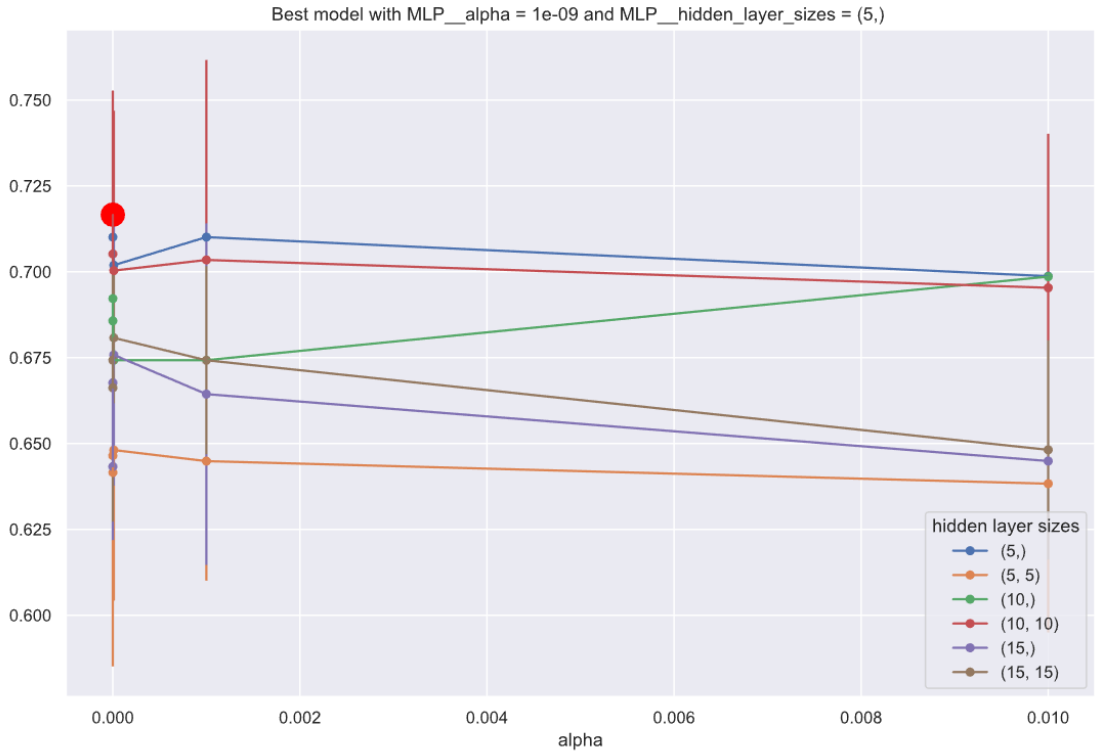
```
INPUTS_ALL =
['PREGNANT','GLUCOSE','BLOODPRESS','SKINTHICKNESS','INSULIN',
'BODYMASSINDEX','PEDIGREEFUNC','AGE']
OUTPUT = 'DIABETES
```

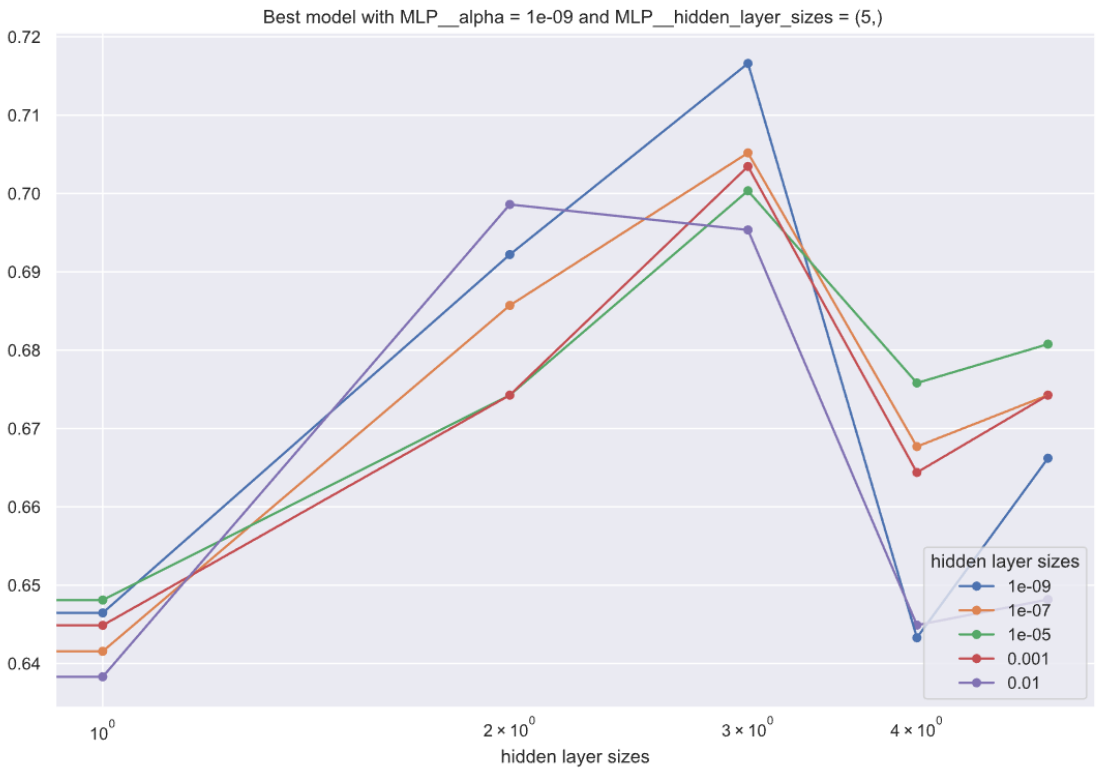
In addition, in order to set up the model, we are going to test the following neural network structures, as well as different L2 penalties.

```
MLP_alpha: [1e-9,1e-7,1e-5,0.001,0.01]# L2 regularization term

MLP_hidden_layer_sizes':[(5,),(10,),(15,),(5,5),(10,10),(15,15)
```

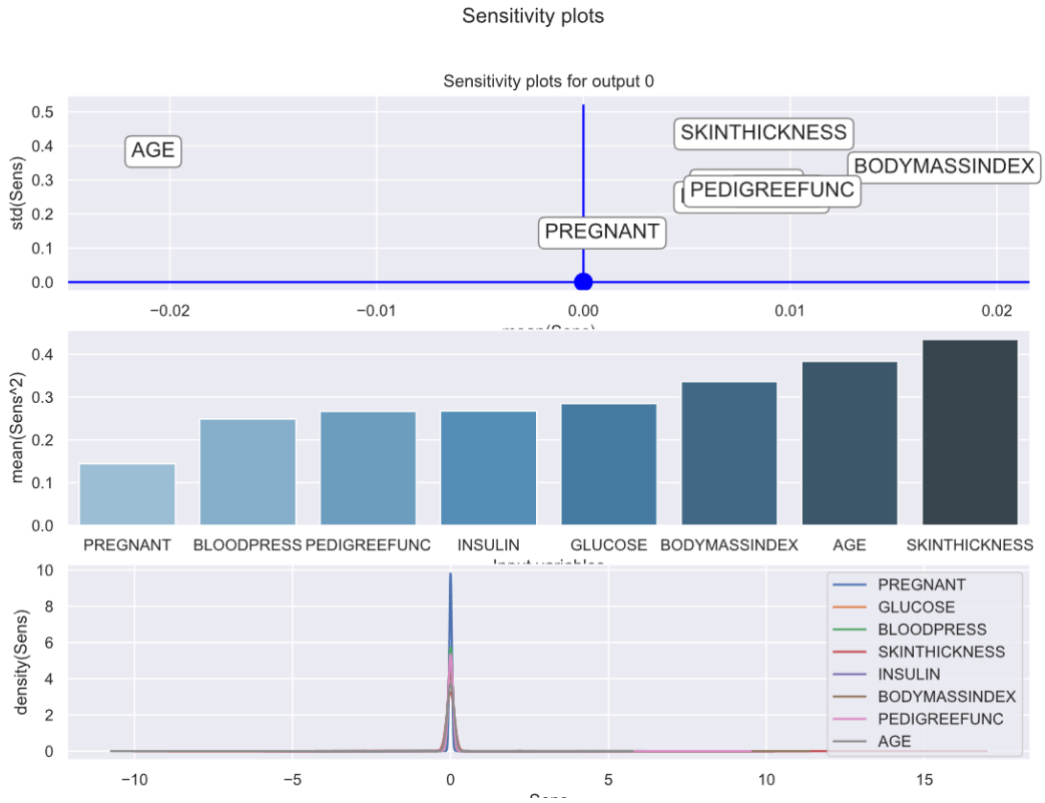
The meaning of the hidden layer sizes variable can be seen as tuples of layers in the neural network. Each tuple value shows the number of neurons in each layer. For example, for the value (5,), it represents that there are 5 neurons in the first and only layer. For the value (10,10), it implies that we have two layers, each with 10 neurons.

With these parameters, the Grid search method with cross validation has chosen the best model with the parameters alpha = $1e^{-09}$ and a single hidden layer with 5 neurons.





The following graphs show the sensitivity of the model. In the first one we can see the importance that the model has assigned to each of the variables of our dataset. As can be seen, the variable 'SKINTHICKNESS' has taken an important role in the model, when we know that most of its data are erroneous. This can penalize us when it comes to prediction.



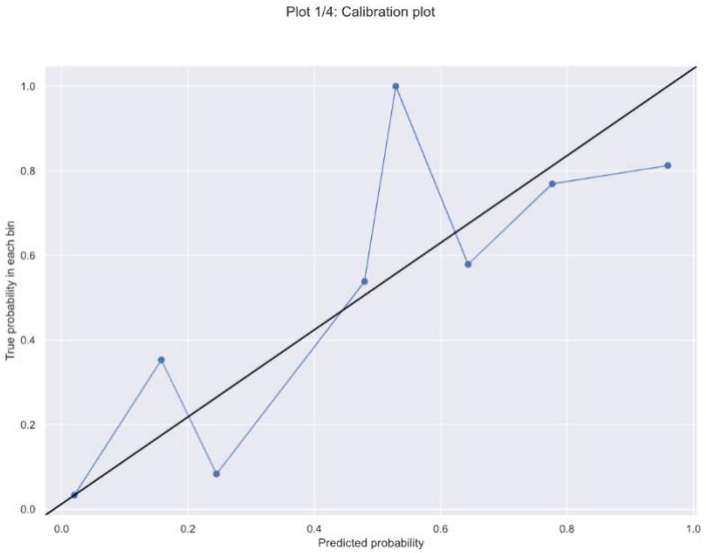
The confusion matrices we have obtained for the best model are:

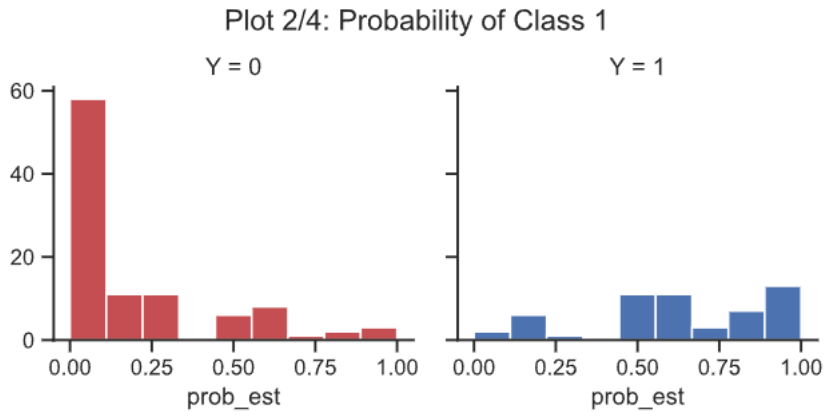
```
----- TRAINING CONFUSION MATRIX -----
Confusion Matrix and Statistics
          Prediction
 Reference   0   1
         0 359  41
         1  63 151

Accuracy: 0.83
No Information Rate: 0.56
P-Value [Acc > NIR]: 0.0
Kappa: 0.62
Mcnemar's Test P-Value: 0.04
Sensitivity: 0.71
Specificity: 0.9
Precision: 0.85
Recall: 0.9
Prevalence: 0.35
Detection Rate: 0.25
Detection prevalence: 0.31
Balanced accuracy: 0.8
F1 Score: 0.87
Positive label: 0
```

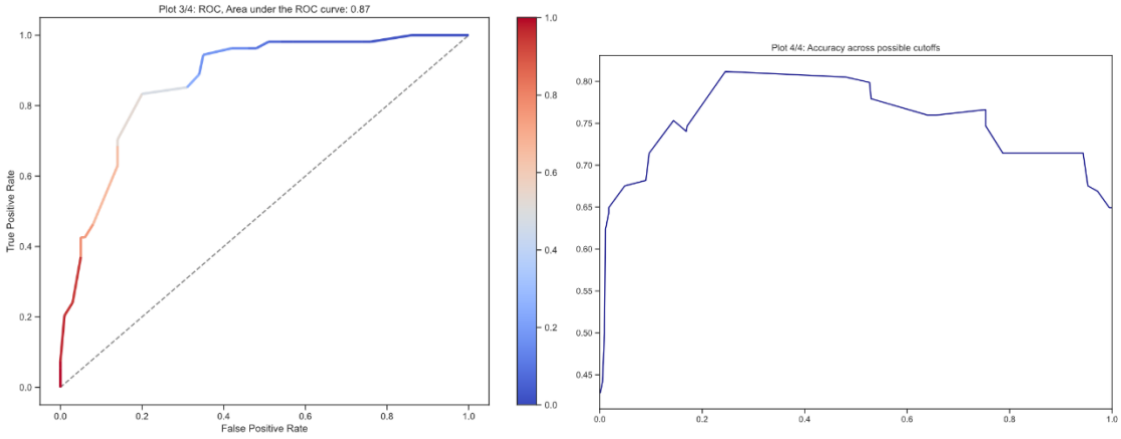
```
----- TESTING CONFUSION MATRIX -----
Confusion Matrix and Statistics
          Prediction
 Reference  0  1
         0 86 14
         1 16 38

Accuracy: 0.81
No Information Rate: 0.55
P-Value [Acc > NIR]: 0.0
Kappa: 0.57
Mcnemar's Test P-Value: 0.86
Sensitivity: 0.7
Specificity: 0.86
Precision: 0.84
Recall: 0.86
Prevalence: 0.35
Detection Rate: 0.25
Detection prevalence: 0.34
Balanced accuracy: 0.78
F1 Score: 0.85
Positive label: 0
```

We can see that we have a 0.83 accuracy in training and a 0.81 in test.

The calibration graphs and the roc curve are shown below, to see how good the model is.





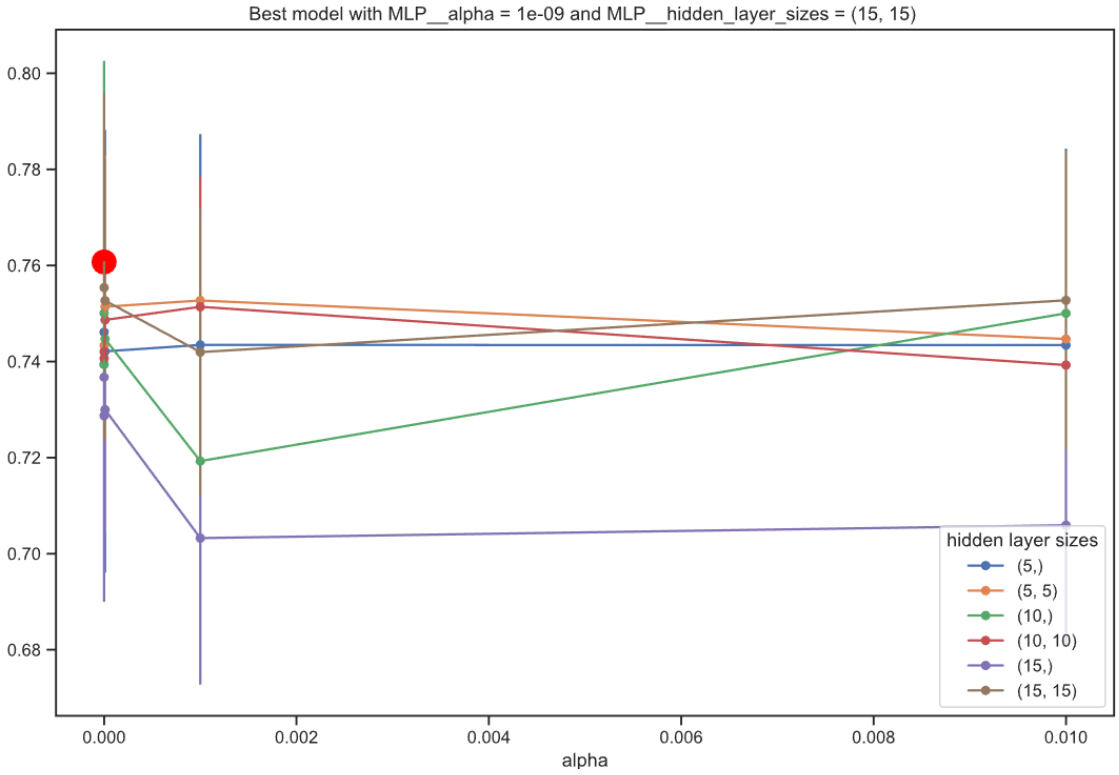


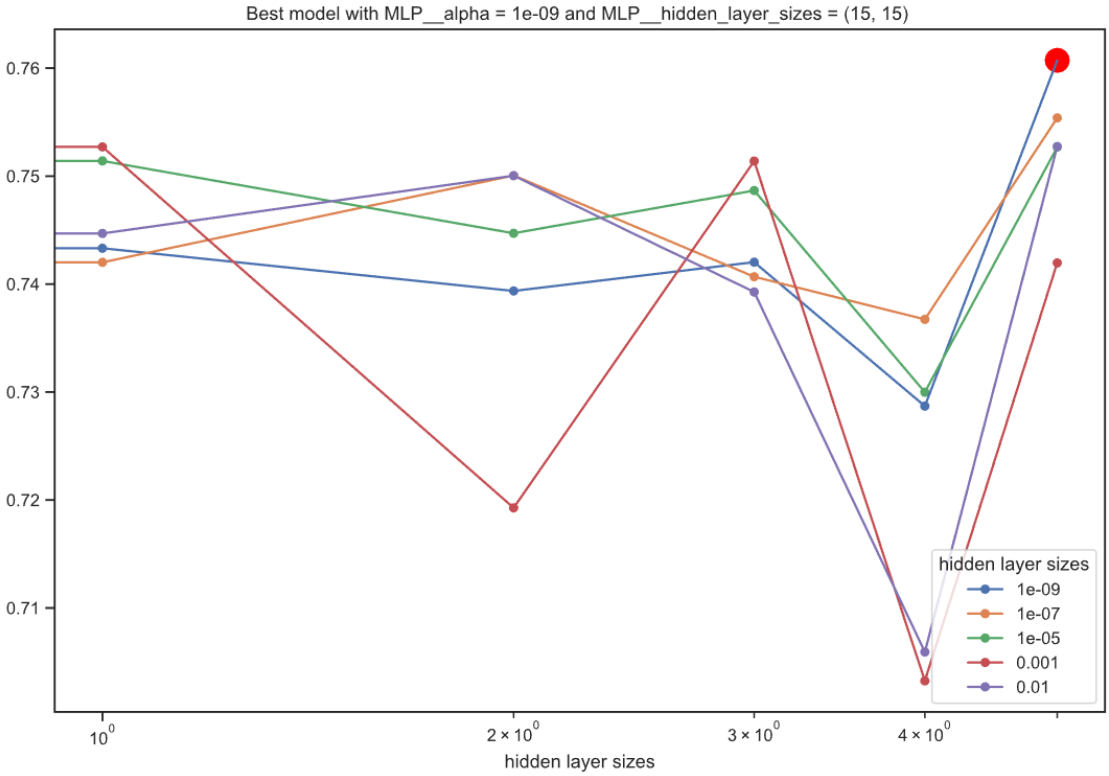
We can see that the area under the roc curve is 0.87, which is quite a good result for this model.

- *TEST WITH MODIFIED DATASET*

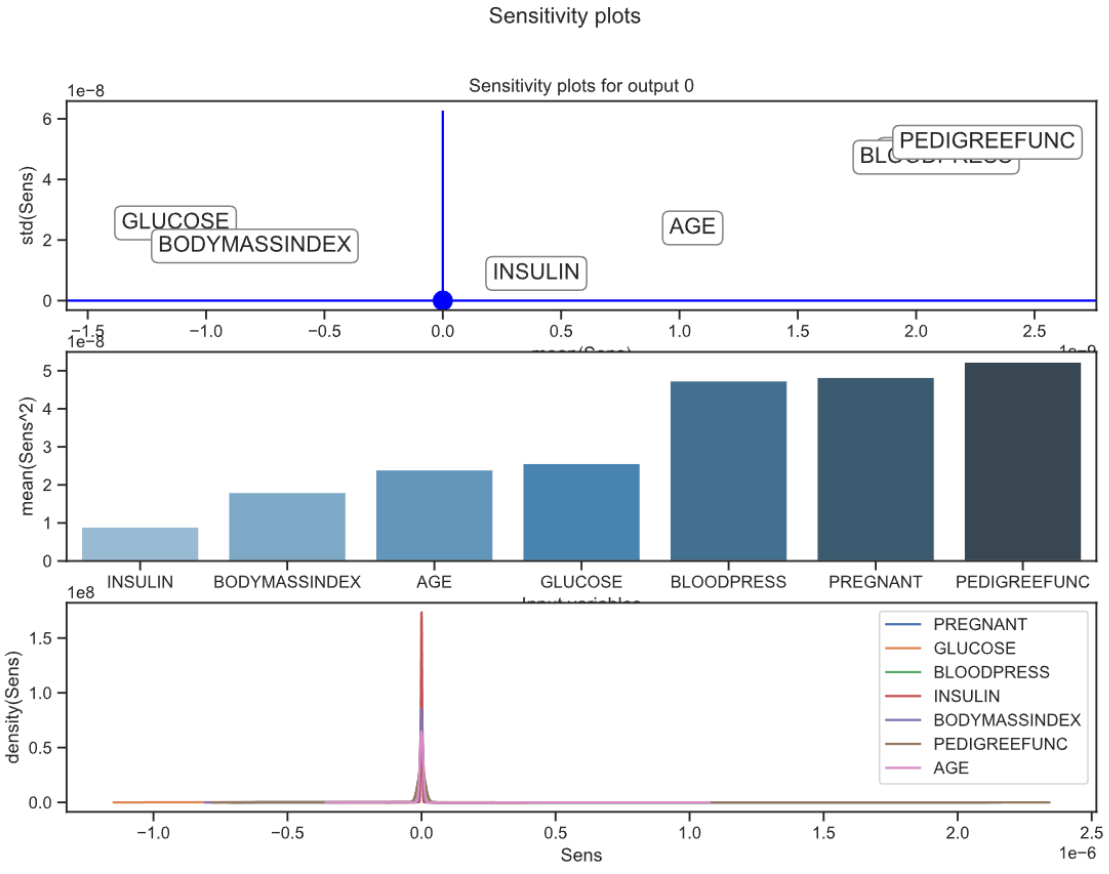
For the same model configuration, both the hidden layers and the regularization terms, we will apply it to the clean dataset, with balanced classes and without outliers.

In this case, the grid search with cross validation has chosen as the best model the neural network with two hidden layers of 15 neurons each, and the same alpha as in the previous model, i.e. alpha = . $1e^{-09}$.





The results of the sensitivity of the model are shown below, indicating which of the variables has been more significant in the prediction.



We can see that PEDIGREEFUNC, PREGNANT and BLOODPRESSURE are the variables that have helped the model the most.

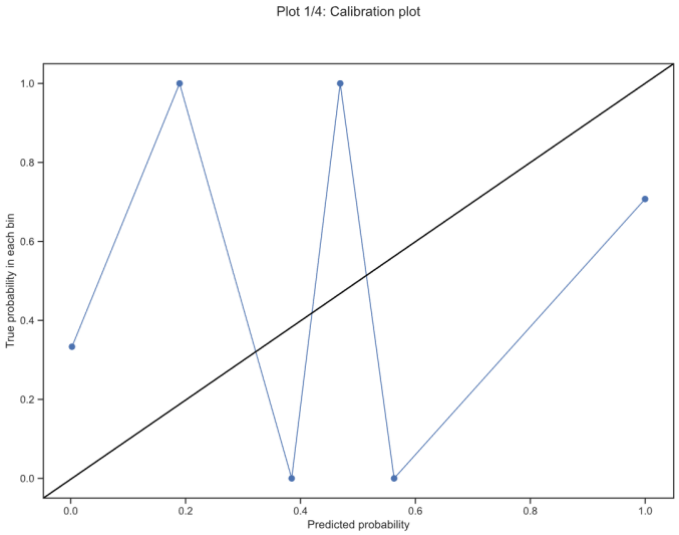
The confusion matrices resulting from this model are as follows:

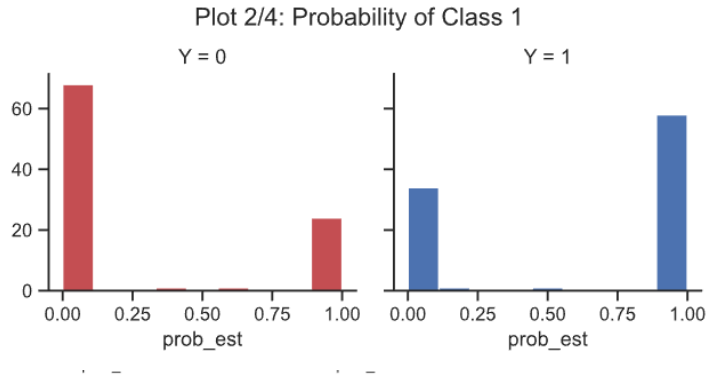
```
Confusion Matrix and Statistics
          Prediction
 Reference   0   1
         0 374   0
         1   0 374

Accuracy: 1.0
No Information Rate: 0.5
P-Value [Acc > NIR]: 0.0
Kappa: 1.0
Mcnemar's Test P-Value: 1.0
Sensitivity: 1.0
Specificity: 1.0
Precision: 1.0
Recall: 1.0
Prevalence: 0.5
Detection Rate: 0.5
Detection prevalence: 0.5
Balanced accuracy: 1.0
F1 Score: 1.0
Positive label: 0
```

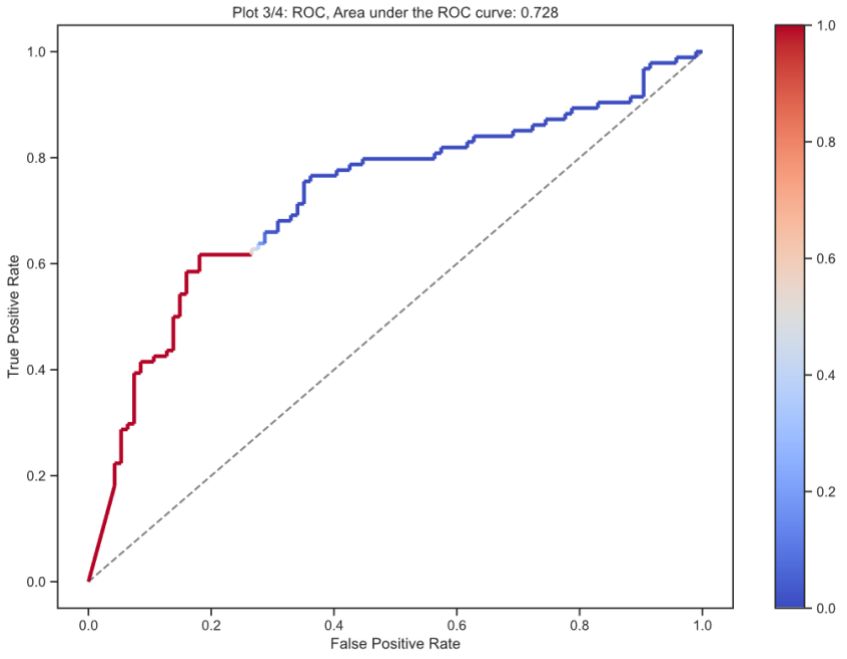
```
Confusion Matrix and Statistics
          Prediction
 Reference  0  1
         0 69 25
         1 36 58

Accuracy: 0.68
No Information Rate: 0.5
P-Value [Acc > NIR]: 0.0
Kappa: 0.35
Mcnemar's Test P-Value: 0.2
Sensitivity: 0.62
Specificity: 0.73
Precision: 0.66
Recall: 0.73
Prevalence: 0.5
Detection Rate: 0.31
Detection prevalence: 0.44
Balanced accuracy: 0.68
F1 Score: 0.69
Positive label: 0
```

We can see that in training we have an accuracy of 1, this may be indicating that it is an overtrained model, although the cross validation has chosen it as the best model. In test we can see that we do not have a very high accuracy, only 0.68, which makes us suspect that it is an overtrained model.

Looking at the calibration curves and the ROC, we can deduce that, although this model had the same options as the other one, the cross validation has given worse results.







Results and conclussions

As conclusions of this model, we have seen that when applying a neural network on a clean and balanced dataset we do not always obtain good results. On the other hand, we have seen that with a single layer of neurons for the original dataset we can obtain quite good results that help us to predict our classes.

On the other hand, we have observed that, although the cross validation chooses some variables as optimal for the model, they are not always optimal, since these variables can cause the model to be overtrained and, therefore, not to make good predictions, as has been the case of our second model.

3. Comparative analysis of the fitted models

In this table, keep in mind that when we refer to diabetes, with all the inputs, we are talking about these variables: PREGNANT, GLUCOSE, BLOODPRESS, SKINTHICKNESS, INSULIN, BODYMASSINDEX, PEDIGREEFUNC and AGE.

But if we are talking about the diabetes_new dataset, and we specify that we use all the variables, they are the same variables as diabetes, but without the SKINTHICKNESS variable, since we have previously removed it from the data frame.

Model	Structure	Inputs	Accuracy CrossVal	Accuracy TRAIN	Accuracy TEST
kNN	diabetes	ALL	0.725	0.73	0.77
kNN	diabetes_new	ALL	0.743	0.81	0.76
kNN	diabetes_new	PREGNANT, GLUCOSE, BLOODPREDD, INSULIN, BODYMASSINDEX, PEDIGREEFUNC AND AGE	0.771	0.8	0.74
DECISION TREE	diabetes	ALL	-	0.76	0.81
DECISION TREE	diabetes_new	ALL	-	0.8	0.78
LOGISTIC REGRESION	diabetes	ALL	0.767	0.78	0.78
LOGISTIC REGRESION	diabetes	'PREGNANT','GLUCOSE','BODYMASSINDEX','PEDIGREEFUNC'	0.764	0.77	0.77
LOGISTIC REGRESION	diabetes_new	ALL	0.7392	0.75	0.71
LOGISTIC REGRESION	diabetes_new	'GLUCOSE','BODYMASSINDEX','PEDIGREEFUNC','AGE'	0.7486	0.76	0.73
SVM Linear	diabetes	ALL	0.767	0.78	0.78
SVM Polynomic	diabetes	ALL	0.6922	0.83	0.73
SVM Radial	diabetes	ALL	0.7622	0.78	0.79
SVM Linear	diabetes_new	ALL	0.7581	0.74	0.72
SVM Polynomic	diabetes_new	ALL	0.7768	0.91	0.81
SVM Radial	**diabetes_new**	**ALL**	**0.8062**	**0.89**	**0.82**
MLP	diabetes	ALL	-	0.83	0.81
MLP	diabetes_new	ALL	-	1	0.68

4. Conclusions

The best model is the Radial SVM model (marked in bold), as we can see this model is made with the dataset we have modified without outliers and with all the data balanced. We see that it has a CV accuracy of 0.80 which is a quite high average compared to other models, both the train and test acuracy are high values, 0.89 and 0.82 respectively. With this model we obtained the best ROC being 0.89, this makes it very good as it is very tight to the corner.

This model is the best fit to the data since, by configuring the radial kernel, it allows us to predict classes that are not linearly separable. In addition, with the hyperparameters selected by the cross validation, cost of 10 and gamma of 0.1, which are quite large, we obtain a fairly wide band, with which we use many of the points in the dataset to build the hyperplane. This makes the model more generic and therefore more robust, which makes it more accurate in predicting the test data.

We think it has been a very rewarding work since we have been able to learn what we have developed in class facing a real dataset, we have had to do an exploratory analysis making several decisions about the data always trying to get a better model.