Facultat d'Informàtica de Barcelona

Diagnosing diabetes in female patients

Machine Learning I

Martí Farré Farrús, 47934800V
Miquel López Escoriza, 54480127V
Pere Cornellà Franch, 41669066N

June 2022

Contents

1 Objectives and data

1.1 Contents of the data set

Our data set consists of observations from 768 patients with 9 variables each, 8 continuous and 1 categorical (the target variable). Our goal is to classify patients that will potentially test positive and negative. The data was collected in the USA in 1990 and all patients were from Pima Indian heritage females with at least 21 years of age. More information on [1].

It is necessary to deeply explain each of the features, as it is from a sector we aren't really familiar with, and it can be hard to get the concept and relevance of each one. We also have made some research about diabetes hoping more understanding will enhance our predicting algorithms. Roughly, diabetes is a disease that affects the way the human body gets glucose into cells. This may occur due to insulin insufficiency (diabetes type I) or because cells in the tissues don't respond well to insulin, known as insulin resistance (diabetes type II).

- *preg:* Number of times pregnant.
- *plas:* Plasma glucose concentration in an oral glucose tolerance test in (mg/dL). The test is commonly performed on patients suspicious of diabetes, if result is over 140 there are some signs of intolerance to glucose or resistance to insulin (prediabetes), if it is over 200 there are clear signs of diabetes.
- *pres:* Diastolic blood pressure (mmHg)
- *skin:* Triceps skin fold thickness (mm)
- *insu:* 2-Hour serum insulin (μU/ml : 10^{-6} units per milliliter). This parameter is typically obtained by giving the patient a dose of glucose to then measure the amount of insulin produced, a hormone that allows the glucose in the blood to enter cells. A diabetic patient should present an insufficient amount (<) and require external administration.
- *mass:* Body mass index (kg/m^2).
- *pedi:* Diabetes pedigree function. This value provides information about the likelihood of having diabetes by looking at the patients relatives taking into account their ages and the degree of kinship. [2]

[1] https://www.openml.org/search?type=data&status=active&id=37&sort=runs

[2] https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2245318/pdf/procascamc00018-0276.pdf

– *age:* Age (years).

– *class*: Class variable (tested positive or tested negative).

We conclude high *plas* (hyperglycemia) and low *insu* are typically symptoms of type I diabetes and mid *plas* combined with and mid *insu* together with low *skin* are typically symptoms of type II diabetes.

1.2 Preprocessing and Descriptive statistics

Here are the first five instances of our data set.

id	preg	plas	pres	skin	insu	mass	pedi	age	class
1	6	148	72	35	0	33.6	0.627	50	tested_positive
2	1	85	66	29	0	26.6	0.351	31	tested_negative
3	8	183	64	0	0	23.3	0.672	32	tested_positive
4	1	89	66	23	94	28.1	0.167	21	tested_negative
5	0	137	40	35	168	43.1	2.288	33	tested_positive

Consider this small sub sample of our data set, we can see that the variable 'id' is not an important feature in any aspect as only identifies the patient so we can erase it. It would be also relevant to know the data type of every variable.

Data columns (total 10 columns):

#	Column	Non-Null Count	Dtype
0	id	768 non-null	int64
1	preg	768 non-null	int64
2	plas	768 non-null	int64
3	pres	768 non-null	int64
4	skin	768 non-null	int64
5	insu	768 non-null	int64
6	mass	768 non-null	float64
7	pedi	768 non-null	float64
8	age	768 non-null	int64
9	class	768 non-null	object

We can see that all the explanatory variables are numerical. Nevertheless, the target variable is a string, so it is a good idea to change this feature to a binary one (from now on 1 = *tested_positive*).

Now we would like to know how the range of values of each variable. The reason for this is to find errors or abnormal values (outliers) and to identify categorical and continuous variables.

	preg	plas	pres	skin	insu	mass	pedi	age
count	768	768	768	768	768	768.00	768	768
mean	3.85	120.89	69.11	20.54	79.80	31.99	0.47	33.24
std	3.37	31.97	19.36	15.95	115.24	7.88	0.33	11.76
min	0.00	0.00	0.00	0.00	0.00	0.00	0.08	21.00
Q1	1.00	99.00	62.00	0.00	0.00	27.30	0.24	24.00
Q2	3.00	117.00	72.00	23.00	30.50	32.00	0.37	29.00
Q3	6.00	140.25	80.00	32.00	127.25	36.60	0.63	41.00
max	17.00	199.00	122.00	99.00	846.00	67.10	2.42	81.00

So we can see that all explanatory variables are continuous. We can also notice some alarming values. For instance, the minimum values for the variables *plas*, *pres*, *skin*, *insu* and *mass* is zero, which can't be impossible. So we will recode these values as `NANs`. On the other hand, there are not unrealistically large values present that could also represent missing values.

We would also like to know if our data is balanced, meaning that we have the same number of diabetic patients than patients who are not.

	percentage
tested_postive	0.65
tested_negative	0.35

We can see that portions are not totally equal but it is not an extreme situation. Now, let's see how many `NAN` values exist in each variable:

	preg	plas	pres	skin	insu	mass	pedi	age	class
`NANs`	0	5	35	227	374	11	0	0	0

We can see that there are way too many missing values to simply erase the rows that contain them. So we are going to apply a method to fill these missing values. We have selected the k-nearest neighbours method (KNN with $k = 10$) which consists of taking the k most similar individuals to the one that has a missing value and fills the missing value with the an average of the other values found in the new patients. See the missing values are gone after performing the algorithm; we print the same header shown in the beginning of the section colouring in red the previous `NANs`.

preg	plas	pres	skin	insu	mass	pedi	age	class
6.0	148.0	72.0	35.0	203.1	33.6	0.627	50.0	1.0
1.0	85.0	66.0	29.0	67.3	26.6	0.351	31.0	0.0
8.0	183.0	64.0	26.8	192.6	23.3	0.672	32.0	1.0
1.0	89.0	66.0	23.0	94.0	28.1	0.167	21.0	0.0
0.0	137.0	40.0	35.0	168.0	43.1	2.288	33.0	1.0

See that in the table shown before of the ranges of every variable there is not any missing value now so we can proceed.

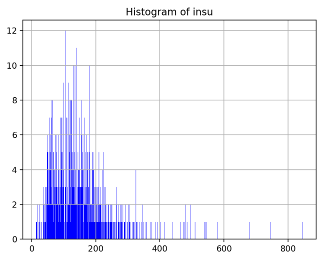
	preg	plas	pres	skin	insu	mass	pedi	age	class
count	768	768	768	768	768	768	768	768	768
mean	3.85	121.63	72.33	29.11	152.77	32.43	0.47	33.24	0.35
std	3.37	30.47	12.17	9.23	95.91	6.88	0.33	11.76	0.48
min	0.00	44.00	24.00	7.00	14.00	18.20	0.08	21.00	0.00
Q1	1.00	99.00	64.00	23.00	89.08	27.50	0.24	24.00	0.00
Q2	3.00	117.00	72.00	29.00	135.00	32.05	0.37	29.00	0.00
Q3	6.00	140.25	80.00	34.18	188.95	36.60	0.63	41.00	1.00
max	17.00	199.00	122.00	99.00	846.00	67.10	2.42	81.00	1.00

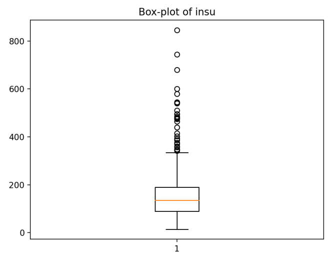
We are now going to take a deep look into a variable of great importance: the insuline.

These are the first 10 values of this variable:

```
203.1, 67.3, 192.6, 94.0, 168.0
93.5, 88.0, 117.5, 543.0, 220.0
```

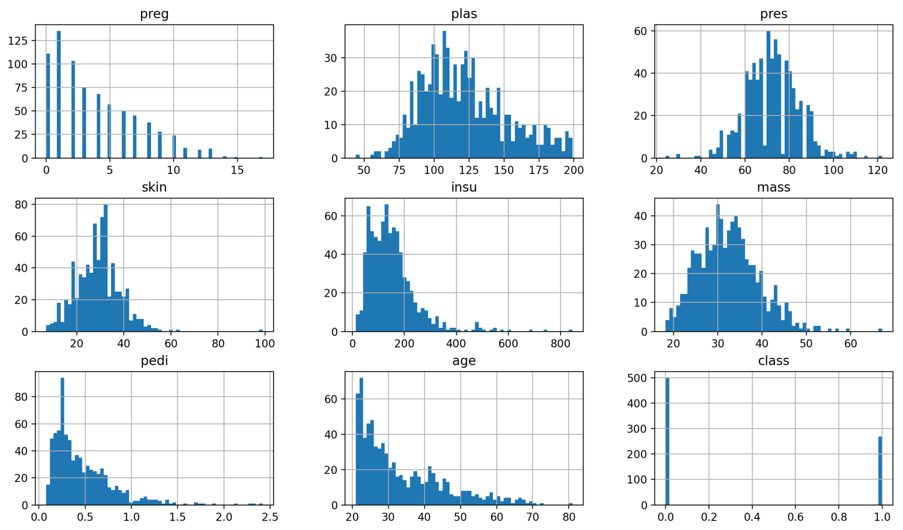
So we can see this feature presents very different values. So in order to know how these values actually vary we are going to take a look into its box-plot and histogram.



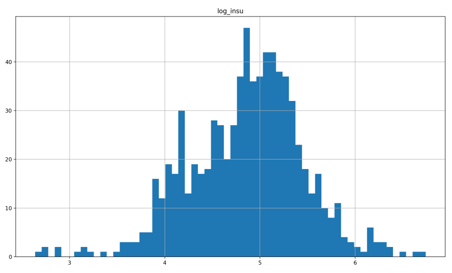


The normal levels of insuline in blood ar below 174 pmol/L. A higher level of insuline means that the pacient has hyperinsulinemia. Alone, it isn't diabetes. But hyperinsulinemia is often associated with type 2 diabetes. We can see that there are a lot of pacients over this insuline value.

Now we are going to focus on the gaussianity of each variable. These are the histograms of all variables.

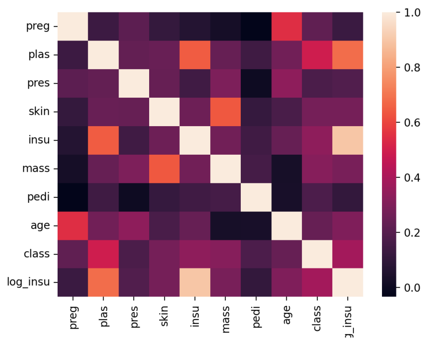


We can see how values of each variable are distributed. If we look closely we can see that a log transformation on insulin might be a good idea as it would convert its histogram into a Gaussian like. See the histogram after the log transformation for insulin



We can actually see how its form changes into a Gaussian-like one.

Now we are going to see graphically the correlation amongst variables. This should give us a sense of what data are we working and how its features are related with each other. We also include in the following table the top 3 correlations.



pair	pearson coef.
mass-skin	0.639
insu-plas	0.628
age-preg	0.544

We propose the following explanations to the Pearson coefficients.

* It is obvious that as a patient's mass increases, the thickness of her skin is also going to augment.

* It has been already explained that when insulin is produced glucose can be assimilated by the cells, so there must be an inverse proportional relation between this variables.

* If a woman is older than another woman the odds that these one has been pregnant more time are pretty high.

Finally we would like to standardise all the variables that we have considered. This is a useful technique as every variable from now on will be using the same scale as all the other. After this process the table shown before presents the following values:

	preg	plas	pres	skin	insu	mass	pedi	age	class	loginsu
count	768	768	768	768	768	768	768	768	768	768
mean	0.77	0.50	0.51	0.76	0.83	0.71	0.83	0.80	0.65	0.46
std	0.20	0.20	0.12	0.10	0.12	0.14	0.14	0.20	0.48	0.15
min	0	0	0	0	0	0	0	0	0	0
Q1	0.65	0.38	0.43	0.70	0.79	0.62	0.77	0.67	0.00	0.36
Q2	0.82	0.53	0.51	0.76	0.86	0.72	0.87	0.87	1.00	0.45
Q3	0.94	0.65	0.59	0.83	0.91	0.81	0.93	0.95	1.00	0.55
max	1	1	1	1	1	1	1	1	1	1

We can see that now all variables vary in the same interval [0,1].

2 Modeling

2.1 Re-sampling

Before submitting the data to make predictions we have partitioned the data set into three subsets, train data to build the models chosen, test data to choose between the final models obtained for each algorithm and prediction data to make the predictions with the final model chosen. The train data represents 75% of the total data. The test data 22% and the prediction data is a 3%.

The train data has been also separated. We have defined X train, which is composed of all the explanatory variables and the Y train, which only contains the class of each patient.

Here is a description of the train data.

	preg	plas	pres	skin	mass	pedi	age	log_insu	class
count	576	576	576	576	576	576	576	576	510
mean	0.78	0.50	0.51	0.76	0.71	0.83	0.80	0.46	0.66
std	0.20	0.20	0.12	0.10	0.14	0.14	0.20	0.15	0.47
min	0.00	0.01	0.00	0.39	0.00	0.00	0.00	0.00	0
25%	0.65	0.38	0.43	0.70	0.63	0.76	0.68	0.36	0
50%	0.82	0.54	0.51	0.76	0.72	0.87	0.87	0.45	1
75%	0.94	0.65	0.59	0.83	0.81	0.93	0.95	0.57	1
max	1.00	1.00	0.94	1.00	1.00	1.00	1.00	1.00	1

We can consider that the train data is balanced.

2.2 Algorithms selection

After an exhaustive research, we have selected a total of five algorithms based on the lectures and previous works [2] [1]. The algorithms chosen are Logistic Regression, SVC, Random Forest Classifier and K-Nearest-Neighbours Classifier.

We briefly explain the algorithms for the sake of understanding the following sections:

Logistic regression: This model classifies each sample by predicting the probability of belonging to each cluster to then assign it to the one that maximizes the probability.

Support Vector Machine: This algorithm aims to find a hyperplane that can perfectly divide the data in (almost) disjoint classes while maximizing the distance between them. It is said that it almost divides since it also considers margins with points from any class.

Random Forest Classifier: It is a bagging technique that classifies using a group of several Decisions Trees that give a cluster for each sample: after each one makes its decision the algorithm assigns the sample to the most frequent result.

K-Nearest-Neighbours: To predict the class of a sample it performs a search to find the k nearest neighbours and assigns the class that maximizes the likelihood.

With the partitions done and the algorithms selected we have tried to find the best model for each algorithm(by hyper-paramameter tuning). For each combination of hyper-parameters we have performed a 10X10-Fold cross validation of the train data. Each one of these partitions will be the validation set once, while the other part of the data is used to construct the model, calculating its coefficients. So we will obtain a total of ten scores for each model which represent the accuracy accomplished, as we have a balanced data this metric is very representative of the capacity of prediction of each model. Our final score is going to be an average of these ones.

For each model we provide also the standard error. This s.e is related to the variance of the model, if the coefficients vary greatly through the ten partitions, then the model has great variance.

2.3 Hyper-parameter tuning

In order to select which algorithm works the best for our problem, we must perform a tuning over their hyper-parameters to find the *optimal* model for each algorithm. For each model, we will proceed considering a set of combinations of parameters, take each one of these and do 10X10-Fold Cross-Validation of the train data to obtain an accuracy score and an error associated to that particular choice. In the end it will allow us to choose for each algorithm the *optimal* model. The names of the parameters correspond with the code. [3]

* For **Logistic Regression** model we tune the following:

  `C`**:** Recall that in logistic regression we aim to minimize

  $$\sum_i (y_i - \hat{y}_i)^2 + C \sum_j \|\beta_j\|_2$$

  So `C` will penalize the size of the vector of parameters used to calculate each prediction $\hat{y}_i$, calculated with the L2 norm. The parameter will be taking the values [0.01,0.1,1,10,100]. We consider this wide range since we don't have any knowledge of the real distribution.

`solvers:` This parameter refers to the type of method(algorithm) that is used to find the minimum of the loss function. We are going to consider the several algorithms to diversify: Newton-CG (conjugate gradient), BFGS and liblinear.

Here is a table the top 5 combination of parameters based on their accuracy.

Accuracy	S.e.	C	Solver
0.762843	0.052172	10	'liblinear'
0.762486	0.052857	10	'newton-cg'
0.762486	0.052857	10	'lbfgs'
0.759719	0.050486	1.0	'newton-cg'
0.759719	0.050486	1.0	'lbfgs'

We note that the solver had few impact on the accuracy. We have tried to perform a second search with C in [5, 7, 10, 12, 15] and no improves were found.

* For **SVC** model the hyper-parameters that we tune vary depending on which kernel we are using. We have chosen the Gaussian Kernel and the linear kernel:

  `C:` Again it is regularization parameter that regulates the penalty associated with the penalization of the complexity of the model. Again, this parameter will be taking the values [0.01,0.1,1,10,100].

  `gamma:` is relevant when considering a Gaussian RBF kernel, in other words, it has no effect on the accuracy for the lineal kernel. We recall the RBFs expression

$$k(x, x') = \exp\left(-\gamma\|x - x'\|_2\right)$$

  so $\gamma = 1/\sigma^2$ and we can interpret it as how much curvature we want in the decision boundary. It will take values [1, 0.1, 0.01, 0.001, 0.0001].

Here is the same table as before for the SVC:

Accuracy	S.e.	C	Gamma	Kernel
0.763533	0.046713	1	1	'linear'
0.763533	0.046713	1	0.1	'linear'
0.763533	0.046713	1	0.01	'linear'
0.763533	0.046713	1	0.001	'linear'
0.763533	0.046713	1	0.0001	'linear'

We remark that all the above are equivalent since gamma has no influence on the linear kernel. We add the best five results for the Gaussian RBF kernel for mere curiosity

Accuracy	S.e.	C	Gamma	Kernel
0.762489	0.048603	10	0.1	'rbf'
0.761639	0.048653	1000	0.001	'rbf'
0.761636	0.049095	100	0.01	'rbf'
0.757855	0.046128	1	0.1	'rbf'
0.757807	0.046455	1	1	'rbf'

We have performed a second tuning revolving around $C = 1$ aiming for an increase of accuracy and we have obtanied a new optimal result

Accuracy	S.e.	C	Kernel
0.766146	0.046282	0.7	'linear'
0.765792	0.046234	0.6	'linear'
0.765103	0.046145	0.8	'linear'
0.764752	0.046212	0.9	'linear'
0.764062	0.047466	0.5	'linear'

* For **Random Forest Classifier** taking into the big amount of time it takes to run this algorithm we consider a wide initial set of hyper-parameters which will be narrowed afterwards:

`n_estimators`: Are the number of trees in the forest. It will take values [200, 500].

`max_features`: The maximum number of features to consider each time to make the split decision on a tree. It will take values ['auto', 'sqrt', 'log2'] which stand for all features N, $\sqrt{N}$ and $\log_2(N)$, respectively. It will allow us to control how much the trees will be unrelated to each other.

`max_depth`: The maximum depth of the trees. It will take values [4,8].

`criterion`: The metric to decide how to build the trees. For clear understanding see [3]. It will take values ['gini', 'entropy'] refering to gini impurity and the Shannon information gain.

Accuracy	S.e.	Criterion	Depth	Max feat.	N estim.
0.749356	0.057036	'gini'	4	'auto'	500
0.749356	0.057036	'gini'	4	'sqrt'	500
0.749356	0.057036	'gini'	4	'log2'	500
0.748990	0.057820	'entropy'	4	'auto'	200
0.748990	0.057820	'entropy'	4	'sqrt'	200

On regard of the results we launch a second search with `criterion` taking values in ['gini'] since it appears that the Gini Impurity criterion

[3] *StatQuest*, Decision and Classification Trees: `https://www.youtube.com/watch?v=_L39rN6gz7Y`

gives the best accuracy, `max_features` in ['log2'] since the number of features has had no noticeable influence and we select the one that might give the fastest execution, `n_estimators` in [300,400,600,700] and `max_depth` in [3,4,5].

Accuracy	S.e.	Criterion	Depth	Max feat.	N estim.
0.749858	0.057167	'gini'	3	'log2'	300
0.749516	0.055840	'gini'	5	'log2'	300
0.748996	0.054096	'gini'	4	'log2'	600
0.748488	0.054714	'gini'	4	'log2'	700
0.747943	0.055022	'gini'	3	'log2'	700

* For **KNeighborsClassifier** we tune the following:

  `leaf_size`: Size of the KDTree or BallTree, both structures to speed up the execution of the algorithm. The larger, the faster the structure is made but slower the classification of a target point. It will take values [1, 20, 40].

  `n_neighbors`: Number of neighbors to use for k-neighbors queries. It will take values [30, 40, 50, 60, 70].

  `p`: Power parameter for the Minkowski metric $\|\cdot\|_p$. It will take values [1, 2, 3, 4].

  `weights`: How the neighbours are weighted when the predictions are made, it will take values ['uniform', 'distance'] which stand for all weighted equally and weighted inversely to it's distance to the point (the closest the most important).

Accuracy	S.e.	Leaf size	N Neigh.	p	Weights
0.755254	0.050878	20	60	1	'uniform'
0.754995	0.050665	1	60	1	'uniform'
0.754994	0.051028	40	60	1	'uniform'
0.753855	0.049263	40	70	1	'distance'
0.753519	0.048964	40	70	1	'uniform'

It is clear that the optimal number of neighbours is winding around 60, and that leaf size hasn't had very much influence and that the best power is $p = 1$. We perform a second search to refine the result (`p=1`, `weights='uniform'` and `leaf_size='20'`)

Accuracy	S.e.	N Neigh.
0.754014	0.051116	62
0.753506	0.053324	64
0.752798	0.055146	68
0.752786	0.053116	60
0.751939	0.053488	70

Now that we have the best set of hyper-parameters for every algorithm, each one of these models will be trained using both train and validation data and will be subject to certain metrics that will evaluate their performance. Our goal will be to end up selecting one model which will be the one used to make predictions with the test data.

2.4 Model selection

To select our final model we need to compare the best combination of parameters we get for each algorithm. We have computed some metrics for each one.

The metrics we are going to analyze to decide which model is going to be used to make predictictions are the one's we have seen in the lectures. These are accuracy, precision, recall and F1-score. This metrics have been calculated using our validation subset.

Accuracy: This metric is the percentage of true positives in our predictions. It is sensitive to imbalanced data, but as we have checked, our data is in fact balanced. We use this formula, where tp are the true positives.

$$\text{Accuracy} = \frac{\sum tp}{n}$$

Precision: This metric is used to check how good the model measures a class with respect of all the predictions we have assigned to this class. This metric is especially good when having a false positive is harmful, but it is not our case. Diagnosing a healthy person with diabetes it only implies a contradiction with future medical tests. It is calculated with the following formula, where fp are the false positives.

$$\text{Precision} = \frac{tp}{tp + fp}$$

Recall: Recall is the metric that measures how good is the prediction of a class our model does with respect of all the values that actually belong to this class. It is very useful when in our problem it is dangerous to have a false negative. This is one of the main metrics we are going to focus on when deciding which model to choose since it is very dangerous to miss-classify a person with diabetes in the sane class, as it could even kill them. It is calculated with the following formula, where fn are the false negatives.

$$\text{Recall} = \frac{tp}{tp + fn}$$

F1-score: This metric is harmonic mean of precision and recall. It is useful in those cases where we want a good balance between both metrics. In our case is not as useful as recall is, but it is still a good metric to have in mind. It is calculated with the following formula.

$$F_1\text{-score} = \frac{2 \cdot \text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

Having this in mind, let's compare the metrics of our methods and choose the best method according to them. In our table we have the general accuracy, and then each metric is calculated for both classes. We also compute its average and weighted average to have a full picture of our metrics.

Logistic Regression metrics
Accuracy = 77.51%

	Precision	Recall	F1-score
0-class	0.74	0.54	0.63
1-class	0.79	0.90	0.84
Average	0.76	0.72	0.73
W. Avg	0.77	0.78	0.77

SVM metrics
Accuracy = 77.51%

	Precision	Recall	F1-score
0-class	0.74	0.54	0.63
1-class	0.79	0.90	0.84
Average	0.76	0.72	0.73
W. Avg	0.77	0.78	0.77

Random Forest Classifier metrics
Accuracy = 76.33%

	Precision	Recall	F1-score
0-class	0.70	0.56	0.62
1-class	0.79	0.87	0.83
Average	0.74	0.72	0.73
W. Avg	0.76	0.76	0.76

K-Nearest Neighbours metrics
Accuracy = 76.92%

	Precision	Recall	F1-score
0-class	0.72	0.56	0.63
1-class	0.79	0.88	0.83
Average	0.75	0.72	0.73
W. Avg	0.76	0.77	0.76

Observe that the results we get with each algorithm are not far from each other. Also notice that the accuracy has improved for every one of them when comparing it with the one we got with the Cross Validation.

When we take a closer look, we see that the worst algorithm is the Random Forest Classifier. Its accuracy is the lowest, and taking a look at the other metrics we can see that it has the lowest values amongst all. It only ties on Precision and F1-score with K-Nearest Neighbours. Although it is a little better than the Random Forest Classifier, K-Nearest Neighbours has lower metrics than the other two algorithms.

The best performing algorithms are Logistic Regression and SVC, with identical values in every metric. They have an accuracy of 77.51%, and the best metric is the recall, with a value of 0.78. As we have mentioned, recall is the most important metric when analyzing our problem and choosing which method is going to model it. But because both Logistic Regression and SVC outperform the other algorithms in every metric, we don't have to prioritize it in exchange to a worse precision, for example.

To help us decide between Logistic Regression and SVC we are going to calculate each algorithm standard error. This standard error is related to the variance of the model. If the coefficients of the model change greatly through the cross validation partitions, we are going to have a large standard error. In our case we get $\sigma_{LR} = 0.2248$ and $\sigma_{SVC} = 0.2130$. We can see that SVC has the lowest standard error between both methods.

This means that our method to predict future data is the **SVC** with $C = 10$ and the linear kernel.

3 Predictions and conclusions

After deciding that SVC is the best algorithm to model our problem, it is finally time to predict with our test data and check the generalization error we get. Before doing so, we refit our model with the train and validation data.

After predicting our test data we get these results. We are comparing the actual class of the instance with our predicted class and we mention if our model has predicted it correctly. The results are displayed in the next table

Real	Predicted	Match
1.0	1.0	YES
0.0	0.0	YES
1.0	1.0	YES
0.0	0.0	YES
0.0	0.0	YES
0.0	0.0	YES
1.0	1.0	YES
1.0	1.0	YES
0.0	0.0	YES
0.0	0.0	YES
0.0	1.0	NO
1.0	0.0	NO
0.0	1.0	NO
1.0	1.0	YES
0.0	0.0	YES
1.0	1.0	YES
0.0	0.0	YES
1.0	1.0	YES
1.0	1.0	YES
1.0	1.0	YES
1.0	1.0	YES
0.0	1.0	NO
1.0	1.0	YES

Our model only miss-classifies 4 observations in our test sub set. If we analyze it, we can see that 3 of these miss-classifications are a non ill person classified as a diabetic person and only one case where a woman with diabetes is classified as sane. This is what we were aiming to, minimizing the recall and the classification of ill woman as non diabetic.

To end our model we estimate its generalization error. We know that in classification is calculated as $GE = 100 - \text{accuracy}$. Knowing that accuracy = 82.61, so we know that $GE = 17.39$. If we take a look at this metrics we can say that our model has a good accuracy, and therefore low generalization error.

To sum up, we have seen that a very important part of building a prediction model is to pre-process the data we are working with. First of all, knowing the variables that compose your data is essential, not only to always keep in mind what kind of data you are working with, but to take advantage of it and use it in order to help build your future model. An example of this would be to derive a new variable that would be of huge use to predict. In our case this was not possible as we couldn't find any combination of variables that made any medical sense. We also found very interesting and important the method we applied in order to replace the Nan values. The KNN regression is a very thoughtful way to fill a row using data information without having to remove it from the sample.

Thanks to the lecturers advise, we avoided committing the mistake of selecting the best algorithm without defining any hyper-parameters in each of them (we were using heuristic values). This led us, on a first instance, to conclude a whole different results heavy conditioned on the bad approach we were taking to the problem. Afterwards, the change of methodology made us realise the importance of the modeling selection. When we tackled this problem with our previous procedure the scores found were not nearly as high as we have found now. In conclusion, we find very astonishing how from a part of the initial data we have constructed a model and we have made predictions of the other part of the data (without cheating) and have obtained an 82 percent accuracy. It has been a factual proof of the use of machine learning in medicine.

Nonetheless, we have encountered a few limitation in this project. Firstly, the population we have been working with is not representative. This meaning that it is only composed by women of a certain characteristics. In our research we have came across with other different projects where data sample was non-discriminant, and predictions were made for all types of population. However much more variables were taken into account and more complex algorithms were studied. Secondly, due to our lack of knowledge we have only considered a handful of algorithms and we assume there may be other ones that may have a better modeling for this type of problem such as Ada Boost Classifier or Gradient Boosting Classifier. We would have also liked to enter in collaboration with some medically trained people to help us derive more sophisticated techniques and to understand what issues are yet to be resolved in this field.

4 Related previous works and sources

We have found many projects revolving around this data-set since is very well known by many data scientist. We must acknowledge how helpful is has been to have so much previous work from more experienced scientists. We have learnt many different approaches to the same problem (and different conclusions!) that we have been able to handle our task with ease.

From our humble experience, we have been able to try to solve, or improve, some points the authors thought could have been done better in their work.

The main projects that have inspired and helped us are cited on the bibliography. As for the few medical explanations, our sources have been so diverse (from articles, text books and even medical professionals...) that we have not been able to reference them as we recognise we have should.

References

[1] **Gopal Joshi**. Diabetes Prediction using Machine Learning. https://medium.com/geekculture/diabetes-prediction-using-machine-learning-python-23fc98125d8, 2021. [Online; accessed 4-June-2022].

[2] **Keshav Dhandhania**. End-to-End Data Science Example: Predicting Diabetes with Logistic Regression. https://towardsdatascience.com/end-to-end-data-science-example-predicting-diabetes-with\-logistic-regression-db9bc88b4d16, 2018. [Online; accessed 4-June-2022].

[3] **Scikit-learn**. Documentation for (Logistic Regression, SVC, Random Forest Classifier, KNeighborsClassifier. https://scikit-learn.org/stable/modules/classes.html, 2022. [Online; accessed 6-June-2022].