Universitat de les Illes Balears

Institute for Cross-Disciplinary Physics and Complex Systems

Classical Machine Learning: Diabetes Classification

Applied Data Analysis and Machine Learning

Author:
Mateu Coll Comas

April 2025

Contents

1 Introduction

The actual amount of data has enhanced the use of Machine Learning (ML) algorithms that, even though their origin goes back to the 1950s as pattern recognition, have grown exponentially in the last years in a wide range of fields such as finance, healthcare, and natural language processing.

We are interested in applying one branch of ML, which is the supervised learning prediction methods, more precisely classification tasks, where we use some features (attributes) to classify the instances. For instance, we could consider the physical attributes of animals as features and classify them as either cats or dogs. The key point is that the given data is labelled. In other words, given the features, we know the classification. Then, based on the feature combination of the provided data, the goal of the ML algorithm is to be able to classify correctly new unlabelled data.

To study this, in this work we analyze a simple numerical/categorical subset of data from [1], which provides information about patients suffering from diabetes along with their respective lab test results. The main goal of this work is to predict whether the patient will have to be hospitalised in the future.

The project is divided as follows: First, we study, explore, and manipulate the data if necessary. Then, we optimise each model used in order to evaluate their advantages and disadvantages. Finally, we conclude which model performs best and provide some comments.

2 Data exploration

Initially, the provided dataset contained 37 columns: 35 of them are the patients' features, one corresponds to the class (whether the patient had to be hospitalised in the future), and another to the patient identification ID. The number of instances (individuals) is 5000. The features are labelled with their corresponding type and are shown in Table 1.

2.1 Missing and repeated values

Before starting to analyze the data and study classification models further, it is important to evaluate the quality of our data. Therefore, we found that there are 163 instances with the same ID, which means that there is duplicated data. To avoid biased results, and since this quantity is relatively small compared to the total dataset size, we keep only one of the repeated instances. This results in a reduction to 4837 instances. The ID feature is also removed.

We also observed that the features *race* and *tolbutamide* have missing values represented as '?'. However, the amount of missing data in each of these features is different. On the one hand, in the *tolbutamide* feature, some 79.8% of the values are missing. Since this percentage is significantly high, we decided to remove this feature from the data. On the other hand, the *race* feature presents only 2.27% of missing data. Thus, we should not proceed in the same way as before.

To deal with this issue, we could consider deleting these instances, but this would lead to a reduction in the data size. Therefore, we propose replacing the missing values by randomly selecting one of the possible values (Caucasian, African-American, Hispanic, Asian, and Other) according to their distribution in the available data.

2.2 Uninformative features

In the dataset, we also find some uninformative features, namely, features that do not provide additional information since all their values are the same. This is the case for: *acetohexamide*, *troglitazone*, *examide*, *glipizide-metformin*, and *metformin-rosiglitazone*. Therefore, we discard these features.

In the data exploration analysis, we also observe some categorical features in which one of the values dominates over the others, sometimes representing more than 90% of the data. For now, we decide not to modify these features and leave this decision to the feature selection procedure provided later in this work.

Feature	Data Type
race	Categorical
gender	Categorical
age	Categorical
admission_type_id	Numerical
discharge_disposition_id	Numerical
admission_source_id	Numerical
time_in_hospital	Numerical
num_lab_procedures	Numerical
num_procedures	Numerical
num_medications	Numerical
number_outpatient	Numerical
number_emergency	Numerical
number_inpatient	Numerical
diag_1	Numerical
diag_2	Numerical
diag_3	Numerical
diag_4	Numerical
number_diagnoses	Numerical
chlorpropamide	Categorical
glimepiride	Categorical
acetohexamide	Categorical
glyburide	Categorical
tolbutamide	Categorical
pioglitazone	Categorical
rosiglitazone	Categorical
troglitazone	Categorical
tolazamide	Categorical
examide	Categorical
citoglipton	Categorical
insulin	Categorical
glyburide-metformin	Categorical
glipizide-metformin	Categorical
metformin-rosiglitazone	Categorical
change	Categorical
diabetesMed	Categorical

Table 1: Features and their data types. The detailed description of each features is written in [1].

2.3 Data manipulation

Once we have identified the simplest quality problems in the data, we now aim to simplify the complexity of the problem by gaining a deeper understanding of the data values. Then, we apply the following modifications:

- We have previously observed that the *race* feature has multiple values. However, 'Hispanic' and 'Asian' represent less than 4% of the data. To simplify the number of possible values, we have decided to include these races under the 'Others' category.
- For *admission_type_id*, *discharge_disposition_id*, and *admission_source_id*, although the data type is numerical (as shown in Table 1), these numeric values are the result of a categorical mapping according to the *IDS_mapping.csv* file from [2]. By observing the different categories in each of these features, we can simplify them as follows:
  - *admission_type_id*: As shown in Table 2, we reduce the number of categories from 8 to 3: Urgent, Elective, Newborn.
  - *discharge_disposition_id*: As shown in Table 3, we reduce the feature dimensionality from 20 to 3: Home, Hospital, Hospice.
  - *admission_source_id*: As shown in Table 4, we reduce the feature dimensionality from 11 to 2: Emergency and Planned. This new label makes a distinction based only on the urgency of the patient.

  For the unknown parameters, we repeat the same procedure as done previously with the missing *race* values. We assign a random category according to the data distribution.
- In the dataset, we can identify 4 diagnosis columns: *diag_1*, *diag_2*, *diag_3*, and *diag_4*. The values of these features are the ICD-9 codes of the primary diagnoses [1], which are numerical. Depending on the interval they fall into, they can be grouped into different disease families, as shown in Table 5. Finally, we note that since the *diag_4* feature is not represented in [1] and differs significantly from the structure of the other diagnosis columns, we decided to remove this feature from the dataset.

The modified and simplified data with all these changes is exported in *preprocessed_data.csv* file from the repository.

Numeric value	Admission type	New Label
1	Emergency	Urgent
2	Urgent	Urgent
3	Elective	Elective
4	Newborn	Newborn
5	Not available	Unknown
6	Null	Unknown
7	Trauma Center	Urgent
8	Not Mapped	Unknown

Table 2: Simplified mapping of *admission_type_id* from [2].

Numeric value	Discharge disposition	New Label
1	Home	Home
2	Another short term Hospital	Hospital
3	SNF	Hospital
4	ICF	Hospital
5	another inpatient care institution	Hospital
6	Home with health service	Home
7	AMA	Hospital
8	Home under care	Home
9	Inpatient to this hospital	Hospital
10	Neonate to another hospital	Hospital
11	Expired	Unknown
13	Hospice/home	Hospice
14	Hospice/medical facility	Hospice
15	Medicare approved swing bed	Hospital
18	Null	Unknown
22	Another rehab fac.	Hospital
23	Long term care hospital	Hospital
24	Nursing facility	Hospital
25	Not Mapped	Unknown
28	Psichiatric hospital	Hospital

Table 3: Simplified mapping of *discharge_disposition_id* from [2].

Numeric Value	Admission source	New label
1	Physician	Planned
2	Clinic	Planned
3	HMO	Planned
4	Hospital	Planned
5	SNF	Planned
6	Anotehr health care	Planned
7	Emergency room	Emergency
9	Not available	Planned
11	Normal delivery	Planned
17	Null	Unknown
20	Not Mapped	Unknown

Table 4: Simplified mapping of *admission_source_id* from [2].

Diagnosis Category	ICD9 Codes
Circulatory	$[390-459]$, $[785]$
Respiratory	$[460-519]$, $[786]$
Digestive	$[520-579]$, $[787]$
Diabetes	$[250.xx]$
Injury	$[800-999]$
Musculoskeletal	$[710-739]$
Genitourinary	$[580-629]$, $[788]$
Neoplasms	$[140-239]$
Other	Other

Table 5: Diagnostic Mapping from [1]

2.4 Numerical data

Now, we are interested in transforming all the data into numerical values. We will then encode the categorical values and normalize the data, as this will be very useful for some classification models such as SVM, Logistic Regression, and KNN, which benefit from normalization.

We can generalize the encoding of the categorical data into the following cases:

- **Binomial values**: For categories with two outcomes, we assign 0 and 1 to each of them. Usually, if the binomial labels are positive/negative, for instance, 'No' and 'Yes', we set 0 to the negative value and 1 to the positive one.
- **Age**: For the *age* feature, the categorical values are age intervals $\{[0,10),[10,20),\ldots,[90,100)\}$. We assign 0 to the younger group, increasing in intervals of 0.1 up to 0.9 for the older range.
- **Medications**: The medication columns, such as *chlorpropamide*, *glimepiride*, *glyburide*, etc., present the values {No, Down, Steady, Up}. For this case, we assign the following mapping: {No: 0, Down: 0.33, Steady: 0.66, Up: 1}.
- **Diagnosis**: We arrange them according to how closely the diseases are related to diabetes and co-morbidity (the presence of two or more diseases in the same person). We go from 'Other': 0.0 up to 'Diabetes': 1.0. Since the order is not very clear for the intermediate values, we group 'Injury' and 'Respiratory' with a value of 0.33 (as they are not commonly related to diabetes in the literature) and assign the remaining diseases a value of 0.66.
- **Discharge disposition**: Finally, we define the following numeric values based on their similarity: {Home: 0, Hospice: 0.5, Hospital: 1}.

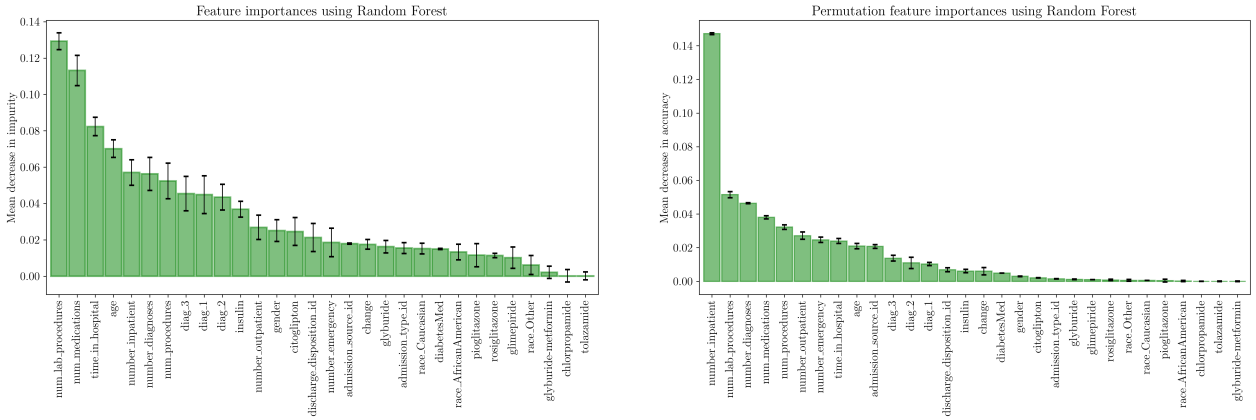
The remaining numerical values are normalized such that they lie in the interval $[0,1]$. The numerical and normalized data is stored in *normalized_data.csv*. From now, we use this dataset.

3 Feature Selection

Up to now, we have focused on exploring, cleaning, and manipulating the data, which now has a final structure of 30 features and 4837 instances. At this point, we may ask whether all the features are relevant for the artificial classification analysis. To study their importance, we apply two different approaches with the help of the scikit-learn documentation [3]: Feature importance based on mean decrease impurity (MDI) and feature permutation.

Feature importance based on MDI, also known as Gini importance, uses a random forest to compute the mean and standard deviation of the accumulation of the impurity decrease within each tree. Specifically, for each node and each tree in the forest, the method measures the impurity of the node before and after each split based on a feature [4]. In Fig. 1a, we show the MDI for each feature in a random forest using the predefined values established in scikit-learn. The higher the MDI, the more important the feature, as it reduces the impurity more. The main disadvantage of this method is its bias toward high-cardinality or continuous features. Therefore, we also analyze feature permutation.

Feature importance based on feature permutation addresses the high-cardinality problem. In this method, features are shuffled n times for a given model, and the mean decrease in accuracy is computed, as the relationship between the feature and the target is broken. With this method, we can observe the dependence of the model on each feature. This method can be applied to any model class, but for simplicity, we only show the random forest classifier. One limitation of applying this algorithm is that if there are high correlations between features, permuting one feature might not break the relationship, as the model can still obtain the same information from the unpermuted correlated features. This results in a lower importance score for the feature, even though it might still be important.



(a) (b)

Figure 1: Feature importance based on MDI (a) and feature permutation (b). We use random forest for simplicity.

In Fig. 1b, we compute the mean decrease in accuracy (MDA) for the random forest classifier with $n = 10$ random shufflings for each feature. As we can observe, the order of importance differs between the two methods, but the most and least relevant features are almost identical in both analyses.

As a result of this analysis, based on Fig. 1b, we discard the features from *glyburide-metformin* to *admission_type_id*. Note that most of these discarded features had nearly all their instances with the same values, so, as expected, their contribution to the model's performance is minimal.

It is important to recall that this analysis is based on a random forest classifier. We could expand our analysis to other classifier models, but the distinction between important and unimportant features is less clear than in the random forest due to high error bars and similar MDA values for all features, among other factors. You can check this in the repository.

Finally, we reduce the data dimensionality to 19 features, thereby simplifying the problem.

4 Performance evaluation

In this section, we study the performance of the different classifiers. In Table 6, the various classification models are presented along with their corresponding scikit-learn functions and parameters.

To avoid overfitting while training the model and ensure generalizability, we apply k-fold cross-validation (CV), in which the data is split into k folds. Of these, $k-1$ folds are used to train the model, and the remaining fold is used to test its performance. The final performance after applying CV is the average value from all possible fold combinations, such that each fold has been used for testing the model. After analyzing how accuracy varies depending on the number of folds in random forests (RF), we set $k = 10$ as the optimal choice for balancing accuracy and computational cost. Therefore, in the following sections, all results are obtained by averaging 10 different values.

Furthermore, since we are analyzing a healthcare classification problem, misclassifying an instance has different implications depending on the type of error. It is not the same if the model predicts that a patient will be hospitalized when it is not necessary (false positive) or if the model predicts that a patient will not

be hospitalized while they suffer from diabetes (false negative). To analyze this discrepancy in classification mistakes, Table 7 defines the cost matrix, where we also reward the detection of true positives.

Model	Pyhton implementation
RF	RandomForestClassifier(n_estimators=500, criterion='gini')
DT	DecisionTreeClassifier(criterion='gini', max_depth=5)
KNN	KNeighborsClassifier(n_neighbors=75, n_jobs=−1)
GNB	GaussianNB()
SVC	SVC(kernel='linear', $C = 2$, probability=True)
LR	LogisticRegression($C = 5$)

Table 6: Machine learning models and their hyperparameters. For parameters not explicitly mentioned, default values are used. RF: Random Forest, DT: Decision Tree, KNN: K-Nearest Neighbors, GNB: Gaussian Naive Bayes, SVC: Support Vector Classification, LR: Logistic Regression.

Actual \ Predicted	Positive	Negative
Positive	-1	5
Negative	1	0

Table 7: Cost matrix with classification outcomes

In the following sections, we analyze the different performance metrics to select the best classifier. The code can be found in this repository.

4.1 ROC curve

The Receiver Operating Characteristic (ROC) curve represents the performance of a model as a function of the threshold configuration. The perfect model is one in which the true positive (TP) rate is 1, and consequently, the false positive (FP) rate is zero.

In Fig. 2, we illustrate the ROC curves for the models discussed previously, along with their respective areas under the curve (AUC) with CV. The higher the AUC, the better the general performance of the model. Additionally, the closer a point is to the coordinate $(0, 1)$, the more optimal the model is. It is important to note that we aim to avoid high false negatives rates due to the high cost of misclassifying a person who suffers from diabetes and needs to be hospitalized. As we can observe, for RF, GNB, SVC, and LR, their behavior and AUC values are very similar. However, for KNN and DT, we see the poorest performance in the prediction task.

4.2 Quality metrics

From the previous section, we cannot draw strong conclusions since some of the prediction models behave similarly. Therefore, in this section, we study the value of different performance metrics: accuracy, precision, recall, and F-measure, which are defined as follows:

$$\text{Acc.} = \frac{TP + TN}{TP + TN + FP + FN}, \quad \text{Prec.} = \frac{TP}{TP + FP}, \quad \text{Rec.} = \frac{TP}{TP + FN}, \quad \text{F-meas.} = \frac{2TP}{2TP + FN + FP}.$$

These values for each classification model are plotted in Fig. 3a. Additionally, in Fig. 3b, we represent and compare the different cost scores obtained based on Table 7.

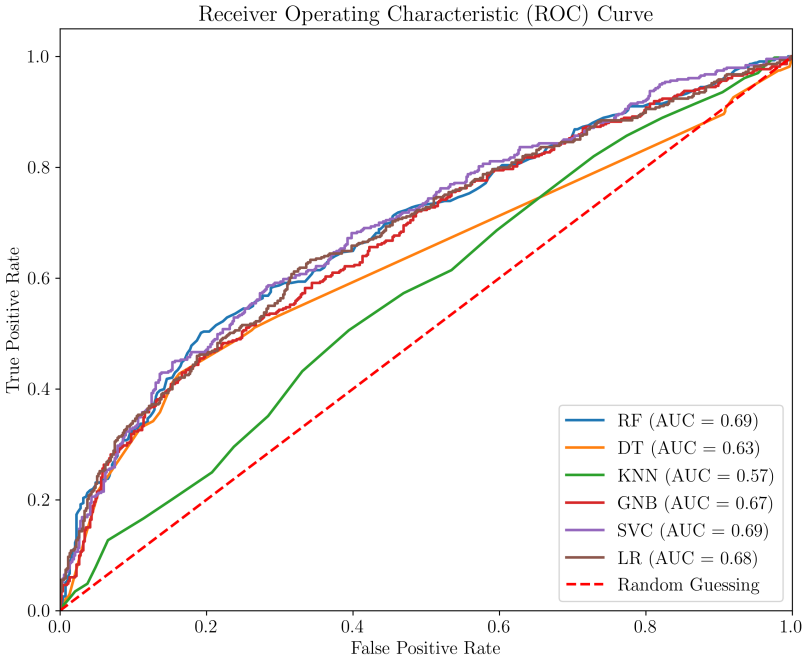
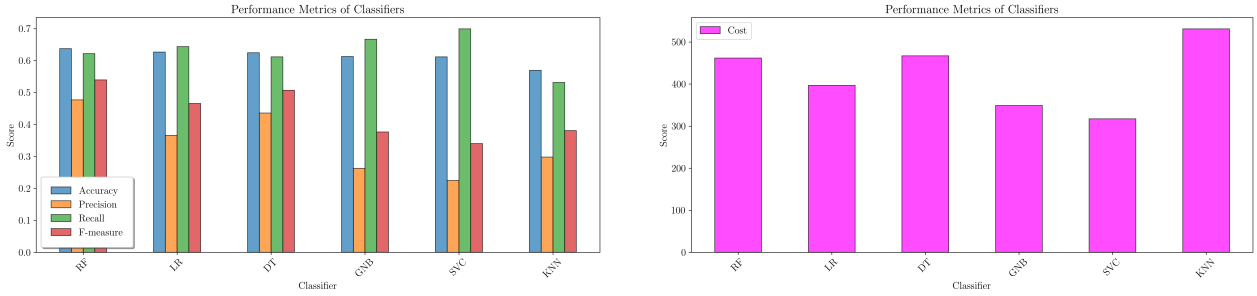


Figure 2: Receiver Operating Characteristic (ROC) curve for each model. The dashed line depicts the random guessing.

From these results, we can highlight that RF is the most balanced classifier, offering relatively good values for all performance metrics. However, the cost score is significantly high, which is not ideal for this classification task. A similar behavior is observed for DT, but with lower values for precision and F-measure. However, when we examine Table 8, we can see that the computational training time for the 10 folds is dramatically faster in DT than in RF. Therefore, for larger datasets, since their results are similar for this type of problem, one could consider using DT instead of RF. Nevertheless, both classifiers tend to produce false negatives more frequently than the other methods.

Then, for LR and GNB, we note that the recall increases, while precision and F-measure decrease in comparison with RF and DT. The high recall indicates that the number of true positives (TP) is relatively higher than the number of false negatives (FN), which is desirable since we aim to prioritize maximizing TP and minimizing FN. Additionally, the small values of precision and F-measure suggest that the number of false positives (FP) is relatively higher than the number of TP, but this is not a poor result given the low cost assigned to FP. This relationship is also reflected in a lower cost score, with GNB performing the best of these two options.

A similar behavior is observed with SVC, but with a greater discrepancy between recall and precision/F-measure. Consequently, SVC achieves the lowest cost score and appears to be the best classifier for this problem. However, it also presents the highest training time, which can increase dramatically for larger datasets. Therefore, choosing GNB seems like a good alternative to replicate SVC's results in less time.



(a) (b)

Figure 3: Metrics for performance evaluation. Each performance emtric has been computed as an average of the cross-validation.

Finally, the worst method appears to be KNN, as its performance metrics are not the best, and it has the highest cost score, which aligns with the information provided by the ROC curve.

Classifier	Time Taken (s)
RF	25.82
DT	0.11
KNN	0.22
GNB	0.06
SVC	23.73
LR	0.26

Table 8: Training time for each classifier to compute the average values.

5 Conclusions

To sum up, in this work we have studied how to handle incomplete datasets and modify them based on the problem at hand. It is important to emphasize that this manipulation of data is not unique and can be done in several ways, leading to different results and model selections. The same holds true for feature selection methods, where the number of features reduced can vary, affecting the outcome.

We then analyzed the significance of misclassification depending on the task. In this case, since we are addressing a healthcare problem, it is crucial to avoid false negatives, as their consequences can be highly damaging. For this reason, by leveraging the performance metrics, we have concluded that, for parameter selection, SVC is the best option for achieving high accuracy and minimizing false negatives. However, due to its high computational cost, we might consider using GNB or LR to obtain similar results in less time. As mentioned previously, some of these conclusions are based on the definition of the cost matrix. A different cost matrix may lead to a different choice of classifier.

References

[1] Brian Strack, Jonathan P. DeShazo, Chris Gennings, Juan L. Olmo, Sergio Ventura, Krzysztof J. Cios, and John N. Clore. "Impact of HbA1c Measurement on Hospital Readmission Rates: Analysis of 70,000 Clinical Database Patient Records". In: *BioMed Research International* 2014 (2014), p. 781670.

[2] John Clore, Krzysztof Cios, Jon DeShazo, and Beata Strack. *Diabetes 130-US Hospitals for Years 1999-2008*. UCI Machine Learning Repository. 2014.

[3] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[4] Cornellius Yudha Wijaya. "Random Forest Feature Importance Explained". In: *Non-Brand Data* (2023). URL: https://www.nb-data.com/p/random-forest-feature-importance.