Hospital Readmission Prediction for Diabetic Patients
Report

Candidate Number: HVWM7
Student Number: 23229722
COMP0189 - Applied Artificial Intelligence
Coursework 1

I. Task 1: Dataset description

The data set describes hospital patient encounters over ten years. It comprises 101 766 samples, and 50 features: identification (`int64`), categorical (`object`), and numerical (`int64`), as well as the target: `"READMISSIONS"`. Identification features allow us to uniquely identify the patient, the type and source of admission, and the type of discharge. The sample-to-feature ratio is as high as 2 000, but will vary once feature selection and one-hot encoding are applied. The challenges in the data set lie in the missing values, numerous categories of a feature, and data imbalance: Missing values percentage can be as high as 94%, and some of them are marked as `"Unknown/Invalid"` or `"?"`, which makes them complicated to identify. Some values are not missing at random, which requires extra care and a deeper feature understanding - see **Table I**. The diagnoses features, medical speciality and payer code show a large number of categories, which brings complexity through higher dimensionality - see **Table II**. The class imbalance can mostly be seen among the medications, where the categories range in `["No", "Down", "Steady", "Up"]`: 19 medications have more than 90% values equal to `"No"`, and 9 of them approximate to 100% of `"No"`. Imbalanced feature counts the ”race”, age, payer code: the `RACE` feature showed a high imbalance in favour of Caucasian patients (**Figure 1**), which represent 76% of the cohort, while the `AGE_GROUP`, which shows a higher number of older patients, as featured in **Figure 2**, which is also consistent with type-2 diabetes. The target feature first looks approximately balanced, with 54% of patients readmitted, as shown in **Figure 3**, but patients readmitted multiple times account as many times as they came: **Figure 4** shows, after the first encounter, how many patients were readmitted: 76% of them were not. Discharge codes needed further encoding through the ICD-9 library for a better understanding of features. Hence, feature interpretation requires high domain knowledge: lack of a deep understanding of the features may result in a lack of clarity in the preprocessing which compromises the interpretability of the machine learning pipeline.

II. Task 2: Data assembling and initial pre-processing

Firstly, an initial pre-processing was held. Patients out-of-scope, with a discharge relating to death, were set out of the cohort. Also, for each patient, only the first encounter was considered. The readmission was also recorded into binary, with `READMISSION` set to one whether the patient came back in more or less than 30 days. The `ENCOUNTER_ID` was dropped: while it gives insights in terms of the chronology of care, this feature brings too much importance. Secondly, a stratified subset of the patient was created, through the `groupby` df method, with `stratification_variables = [target, 'RACE',' GENDER']`. The dataset is not IID: when assembling it, it remains crucial to take into consideration the correlation and unbalance of the dataset. It was voluntarily chosen not to stratify by age, first because over-stratification may overfit the data, and second because the task is about type 2 diabetes, which is a condition that follows adults over 40. Thirdly, the dataset was split into `X` and `y`, and then into `X_train, X_test, y_train, y_test` with scit-kit learn's `train_test_split` - see **Table III**. Finally, the preprocessing pipeline was defined and prepared, but *not executed* on the dataset, to facilitate the cross-validation pipeline in the third task. The preprocessing consists of dropping columns with more than 95% of unique values, creating a filter for categorical data with more than 10 categories, and only keeping the most 10 frequent categories. This will be applied to the diagnoses, medical speciality and payer code features. encoding categorical data was the final step after feature engineering was held. The following features showed categories that are either ranked `'AGE', 'MAX_GLU_SERUM', 'A1CRESULT', 'GLIMEPIRIDE', 'GLIPIZIDE', 'GLYBURIDE', 'INSULIN', 'METFORMIN', 'PIOGLITAZONE', 'REPAGLINIDE', 'ROSIGLITAZONE'` or binary `'GENDER', 'CHANGE', 'DIABETESMED'`: an Ordinal Feature Encoding will be applied on them. The following features are then passed into One Hot Encoding: `"RACE", "PAYER_CODE", "MEDICAL_SPECIALTY", "DIAG_1", "DIAG_2", "DIAG_3"`. The preprocessing steps are detailed in **Table IV**.

III. Task 3: Design and build a machine learning pipeline

A. Metrics and model choices

As we're focusing on a binary classification of readmitted patients, the selected metrics are the **accuracy**, the **f1 score**,

the **precision** and **recall**. Accuracy offers an overall success rate, while precision and recall highlight the model's ability to correctly identify positive cases. The F1 score harmonizes precision and recall, offering a single metric for model comparison. On top of the linear Support Vector Machine (**LinearSVC**), the first evaluated selected models are **SVC**, **KNN**, and **RandomForestClassifier**. The model selection was based on scikit-learn's ML mapping studied in class, as we are in a classification task, with labeled data, a sample smaller than 100k, and non-textual data.

A first model evaluation without cross-validation was made to assess the best scaling technique. First, the preprocessing steps were applied to the dataset, followed by these scaling methods: `StandardScaler`, `MinMaxScaler`, `Normalizer`. This is done with in mind the idea that LinearSVM's and KNNs' results are highly dependant on distance calculations which are scaling-dependant, as well as non-linear SVM that seeks structure in data, hence is scaling-dependant. (**Figure 5**) show that all models show very low results for all metrics, with accuracy performing better than all of the other metrics. Also, the StandardScaler performs slightly better than the MinMaxScaler, which perform way better than Normalizer: we choose the **StandardScaling**. These results suggest that RandomForestClassifiers are more suited for this task than KNN. This can be explained by the non-linearity of the data, which RandomClassifiers handle well, while KNNs struggle with high dimensionality, imbalanced and noisy data: we **replace KNN with GradientBoosting**.

B. Cross-Validation Pipeline

The Cross-validation (CV) pipeline is designed to optimize model hyperparameters as described in **Table VI**. Given the unique patient identifiers in our dataset, we opted for `StratifiedKFold` cross-validation to maintain the proportion of target classes in each fold, ensuring the representativity of features and reducing bias. Pre-processing steps, handling missing values, filtering the categories to only select the most frequent, and rescaling, were uniformly applied to maintain data integrity and comparability across models, as shown **Table V**. To process with the cross-validation, a few parameters have to be defined: first, the model is defined with all its parameters. Then, the grid, meaning the set of hyper-parameters to evaluate, as defined in a dictionary: the selected grid is available **Table VI**, each key in the dictionary represents a hyperparameter to be tuned, and the value is a list of possible values for that hyperparameter. Then, a stratification method is defined, taking into consideration the number of splits, which was selected as `5` here, as well as the shuffling which is set to `True`, to not let chronology be a factor. Finally, the cross-validation pipeline is defined, which consists of the preprocessing steps defined beforehand, as well as the model implementation.

Now that the parameters are defined, we iterate on models: for each iteration, a `grid_search` is developed, with the cross-validation pipeline as an estimator, the grid as the `param_grid`, the StratifiedKFold as a `cv` parameter, and the selected metric as the `accuracy` parameter. Once defined, the grid_search is fitted into our dataset, with `X_train` and `y_train` as inputs. Then, y_hat is predicted through the `predict` method of the `grid_search`. Finally, the results are iterated on the metrics, and stores for each model and metric pair, saves the parameters, the **mean test score**, the **std test score**, as well as the **real test score** which is outside of the cross-validation, but uses our metric to compare the predicted valued with the true label. For each iteration, the model, metric, mean test score, standard deviation and test score are printed to assess the stage of our pipeline.

This pipeline was first made to be iterative on a list of models, but, due to the high dimension and complexity of a bigger result data frame, and specific performance and interpretation assessments, it was decided to hold the experiments separately for each model.

C. Performance Evaluation

To quantify the performance of the model, we assess the defined metrics: accuracy, precision, recall, and F1 score, for the mean test score, the standard deviation, as well as the test score. Moreover, we assess the confusion matrix and observe how the mean test score evolves as a function of the hyper-parameters.

First, the accuracy, precision, recall and F1-Score show low results on the Test Score: The accuracy lies around 0.61 for all models, with a slightly higher value for RandomForest and GradientBoosting (0.621). For f1-score, values are between 0.35 (LinearSVC) and 0.46 (SVC). Regarding the precision, all model values' are around 0.53, except for GradientBoost which is around 0.65. Finally, the recall is between 0.3 (LinearSVC) and 0.44 (SVC). These results show very close performances and don't allow us to draw any first conclusions. Accuracy results are available **Figure 6, 10, 13, 19**. In a nutshell, while Linear models have best accuracies aroung **0.60**, RandomForest and GradientBoosting show slightly better results, around **0.62**.

Secondly, the confusion matrix for LinearSVC and SVC are quite close, with the highest number of true positives (1015 and 1030 respectively), but, more worryingly, a high number of false negatives (596 and 597 respectively, which is also observed for tree-based models, which show a slightly highest number of true negative, but a higher number of false negatives as well. Confusion matrixes are available **Figure 7, 11, 14 and 20**.

Finally, the hyperparameter evaluation allows to see that, for Linear Models (LinearSVC and SVC), the C parameter, which is inversely proportional to regularisation, shows best results for lower C, hence for higher regularisation. For LinearSVC, the mean score starts to decrease for C over 1. Also, rbf and linear kernels behave differently: **Figure 12** shows that the rbf training set has growing scores for a growing C, which could show overfitting. On the validation set, the linear kernel shows more robust results while the rbf kernel has better results for C between 5.10e-1 and 5.10e+1, but not at extremes.

Moreover, let's note that for RandomForest and GradientBoosting, a higher difference of scores between the validation and training test was observed for deeper trees: This shows that the Trees can easily overfit and puts limits to our models.

IV. Task 4: Model Interpretation

To better understand how the model assessed the patients features with regards to readmission, the confusion matrix is plotted for all models, while the coefficient importance was plotted with Linear Model (**LinearSVC** and **SVC**) and feature importance were plotted for Tree-based and Ensemble Models (**RandomForestClassifier** and **GradientBoostingClassifier**). To go further with our interpretation assessment, **Permutation feature importance**, as well as Partial Dependence Plots, were conducted.

LinearSVC, RandomForest and GradientBoosting have all identified `AGE`, `INSULIN`, `DISCHARGE DISPOSITION` and `TIME IN HOSPITAL` as principal features, as well as the secondary and third diagnoses. There interpretation differs where Tree-Based Models were more sensitive to the `PATIENT NBR`, the `NUMBER INPATIENT` and the `ADMISSION TYPE`, while LinearSVC was more sensitive to `LAB PROCEDURES`, `DIABETES MED`, and `INSULIN`.

Feature importance slightly vary across models, but all of them show as top values, number of diagnosis, the first diagnosis, as well as the age, number of lab procedures, and time in hospital. The data is high dimension, and the model is not sparse, which makes it difficult to interpret the model, but feature importance provide a good interpetability measure.

V. Task 5: Alternative machine learning pipeline

Cross-validation is particularly efficient when the dataset is small, but shows limitations in terms of representativity and randomness: the stratification may not be optimal and the random splitting of the data can lead to very different, unreliable results. We implement Nested Cross-validation (NCV) to address data imbalance and enhance model performance, which are critical challenges in our framework. NCV not only facilitates model evaluation and selection but also ensures more dependable outcomes by preventing the test data from being non-representative. This method is particularly beneficial for models like Gradient Boosting or Random Forest, which have many hyperparameters, as it utilizes each inner loop to optimize a hyperparameter. Moreover, NCV maximizes data utilization by training the model multiple times—specifically, the number of inner loops multiplied by the number of outer loops. The inner loop focuses on model assessment through hyperparameter evaluation, while the outer loop is dedicated to model selection. Overall, NCV offers a comprehensive and more robust approach to model development.

The NCV implementation is very close to the CV, in the difference that inner and outer loops are defined for NCV. First, the inner and outer cross-validation methods are selected: we choose Stratified K-folds, with 5 outer splits, and 8 inner splits. An empty results dictionary is initialised as well. Just as with cross-validation, the models, and grids (hyperparameters) are initialised as well. Then, the pipeline iterates on the models and metrics, and starts with the outer cross-validation loop by iterating over training and test set indices generated by a cross-validation splitter. This step involves creating separate training and test datasets for each fold. This step involves initial pre-processing steps that don't involve data leakage. Moreover, the initial datasets are `X` and `y`, as the NCV takes care of creating a training set (outer loop). Afterwards, the training data from the outer fold is further split into smaller training and validation sets multiple times according to the inner cross-validation strategy we defined, StratifyKFold. Therefore, in each inner loop, the model is trained on the inner training set and evaluated on the inner validation set.

VI. Task 6: Identify limitations and propose potential solutions

This data set is rich and diverse, while it presents numerous limitations in terms of clinical evaluation, data encoding, and social changes.

First, the dataset has numerous limitations: following patients for a whole decade means facing inconsistencies in the dataset due to evolutions in technology and social considerations. The diagnoses are encoded in ICD-9, published in 1977, but were surpassed by ICD-10 (1994) and ICD-11 (2022). Notably, the dataset's handling of sensitive attributes such as race and gender may not reflect contemporary inclusivity standards, potentially affecting the applicability of the findings across diverse patient populations. Also, a significant limitation arises from the handling of missing data, particularly for crucial variables like patient weight, which is a key factor when following diabetic patients. A first optimisation of the dataset would have been to consider long-term complications, change of diagnoses, and correlate the readmission clinical results with the previous encounter, and identify patterns in admissions diagnoses and changes in clinical treatment. Furthermore, the study's analytical framework does not accommodate an in-depth examination of patients with more than two admissions within the same year, nor does it explore the frequency of such admissions.

Secondly, the machine learning and data science pipeline could benefit from a reassessment of the readmission setting, as well as providing more crucial clinical information. The data set initially separated patients readmitted within 30 days or after 30 days, according to clinical studies of diabetes. Also, a deeper scope definition is needed, especially when considering patients transferred to Skilled Nursing Facilities (SNFs).

Finally, the pipeline showed strong limitations, which shows that further data understanding and domain knowledge is needed to get the most out of it. Moreover, a change of paradigm, to assess the "chances" of readmission, by changing the task into a regression, and then using a softmax, could have been an adequate attempt, or to use deep learning methods to capture more feature patterns from the data.