Lab 07: Model Training & Evaluation
CS3300 Data Science

Goal

- Apply cross-validation to perform model selection and feature selection.
- Practice the use of Pipeline from Scikit-learn.
- Choose the appropriate metric score and baseline model.
- [Optional] - Experiment with some of the techniques that address imbalanced learning.

Overview

For lab 7, you are going to analyze the Stroke prediction dataset that can be found here. Each row in the data provides information related to a patient (gender, age, various diseases, smoking status) and specifies whether the patient had a stroke. The data set has 5110 observations (rows) and 12 variables.

The columns are as follows:

- id: unique identifier
- gender: "Male", "Female" or "Other"
- age: age of the patient
- hypertension: 0 if the patient doesn't have hypertension, 1 if the patient has hypertension
- heart_disease: 0 if the patient doesn't have any heart diseases, 1 if the patient has a heart disease
- ever_married: "No" or "Yes"
- work_type: "children", "Govt_jov", "Never_worked", "Private" or "Self-employed"
- Residence_type: "Rural" or "Urban"
- avg_glucose_level: average glucose level in blood
- bmi: body mass index
- smoking_status: "formerly smoked", "never smoked", "smokes" or "Unknown"*
- stroke: 1 if the patient had a stroke or 0 if not

You will perform an exploratory data analysis (EDA) and then build a classification model to predict if a patient is likely to get a stroke.

Instructions

Your work will consist of the following steps:

1. Exploratory Data Analysis

- Guess which columns will be good predictors of presence of stroke before looking at the data.
- Load and explore the DataFrame using `head()`, `info()` and describe() (there's one column that has some missing entries. For this lab, we want you to use SimpleImputer of sklearn. You can ignore these missing entries in your EDA),
- Convert the columns to the right types.
- Characterize each variable. Plot the distributions of each variable and describe the range of values.
- Explore the relationships between each variable and the target variable. Choose the appropriate plots (*and statistical tests [optional*]) based on the variable types (e.g., categorical, numerical). Do your plots support your initial guesses or rule them out?

2. Train-Test Split of the data

- Separate the feature matrix X and the target variable y.
- How many rows in your dataset belongs to class 1. Comment on the severity of imbalanced classes in the data.
- Divide the data into training and testing sets. The training set will be used for model and feature selection. The testing set will be used to evaluate the final model.
  Useful link: sklearn.model_selection.train_test_split.

3. Choice of evaluation metric and baseline model

- What is the appropriate evaluation metric to use if false positives and false negatives are both costly?
- When we don't know ahead of time, which model or approach to choose, we need first to evaluate a suite of different types of algorithms with minimal hyperparameter tuning. In order to understand how skillful these algorithms are, we need to compare their performance to a naïve model or baseline model. When the minority class 1 is of interest for us and we choose F1-score as the evaluation metric, an appropriate choice for a baseline/naïve model would be a model that predicts the minority class in all cases. To create a baseline model, use the class DummyClassifier and create an instance of it. Make sure to specify the strategy as "constant" and to specify the constant parameter as 1.

4. Define the pipeline or the pre-processing steps of your model

Before we start to evaluate different machine learning algorithms, we need to encode the categorical columns, scale the features in the data set and impute the missing values in the column "bmi" (Note: imputing missing values with the mean or median might not be always the best approach. For this lab, we want to experiment with SimpleImputer provided by Scikit-learn). These pre-processing steps should be always applied on the data we're using for training (whether we're doing cross-validation or evaluating the final model). In scikit-learn, creating a Pipeline object that consists of the pre-processing and modeling steps can help ensure we're following the best practices.

- First create a Pipeline that consists of two steps: SimpleImputer followed by a StandardScalar. This pipeline defines the pre-processing steps that we want to apply to the numerical features.
- Since we need to apply different pre-processing steps to the numerical and categorical columns, we need to clarify what these pre-processing steps are and to which columns they should be applied. This can be done in scikit-learn using ColumnTransformer. Create a ColumnTransformer that consists of two transformations: 1. OneHotEncoder for the categorical columns and 2. the pipeline you created in the previous part for the numerical columns.
- Finally, create the main Pipeline which consists of the ColumnTransformer that you created in the previous part and of the specific model that you want to try. Let's try first logistic regression.

5. Perform Model Selection using cross-validation on the training set

Using cross_validate, evaluate the main Pipeline you created in the previous step. Make sure to specify the cv parameter as 5, the scoring parameter as 'f1' and to use the training set. Compute the average F1-score obtained on the validation sets. Repeat the steps but now to evaluate different machine learning models:

- Linear model: logistic regression
- Non-linear model: naïve bayes
- Tree-based model: random forest.
- Class-weighted logistic regression: LogisticRegression (class_weight="balanced")

What is the average performance in terms of F1 for each model? How does it compare to the average performance of the baseline model? What if we set the parameter class_weight to "balanced" for logistic regression?

6. Address Imbalanced Learning – Optional

Setting the class_weight parameter of logistic regression to "balanced" is one way to draw the attention of the training algorithm to the minority class. There exist other techniques that address imbalanced dataset: ***oversampling*** (generate new samples in the classes which are under-represented) and ***undersampling*** (balance the data by randomly selecting a subset of data for the targeted classes). The package **imblearn** supports many techniques that address imbalanced learning. Feel free to try some of these techniques:

- First you need to install the package (pip install imblearn).
- Second you need to import the Pipeline from imblearn (it is the same exact syntax)
- Create a RandomOverSampler instance and add it to the main pipeline between the column transformer and the machine learning model (make sure to remove the parameter class_weight if you're working with logistic regression). What is the average F1 performance?
- Create a RandomOverSampler instance and add it to the main pipeline between the column transformer and the machine learning model (make sure to remove the parameter class_weight if you're working with logistic regression). What is the average F1 performance?
- Create a RandomUnderSampler and repeat the same steps.
- You can also try to evaluate the BalancedRandomForestClassifier.html

7. Feature Selection – greedy approach

If you did not do part 6, then only focus on the use of class-weighted logistic regression. Otherwise, choose the model that looked the most promising. Now select the most predictive features using a "greedy" approach. Build a model for each variable individually. Sort the features from highest F1 to lowest F1. Then start again by adding each feature one at a time to the model. If feature improves the model over the last model, keep that feature in the model. If the feature does not improve F1, skip that variable. At the end, you should have a single model with multiple features.

a. Which features were the most predictive? Can you give explanations for why you think these features were the most predictive based on your exploratory analysis?

b. Which approach (exploratory analysis, greedy, or all variables) produced the best model?

8. Evaluation of the final model

Train the best model on the entire training set. And evaluate it on the testing set,

Submission Instructions

To submit, please upload a pdf or HTML version of your notebook.