Group 33 - Chapter 7

Member 1: Anissa Hidouk
Member 2: Akrame Assoufi
Member 3: Jean Baptiste Tartarin
Member 4: N'chonon Cho Carelle
Member 5: Malek barbirou

Introduction

In chapter 7, the book basically walks us through the process of evaluating a machine learning model and make sure that it is flexible and can be autonomously accurate.

In machine learning, we teach the computer how to make predictions or decisions based on data. But, we don't just want the computer to make guesses randomly. We want it to make accurate predictions based on what it has learned. So, we need to evaluate the model's performance to make sure it's doing the best job.

Evaluating the model's performance means checking how good the predictions are, and we can do that using different measures. We want to make sure that the model can make accurate predictions on data it hasn't seen before. That's why we need to test it on a different set of data than what we used to train it. It's like teaching a robot how to do something, and then checking if it's doing it well, and if not, figuring out how to make it better.

The chapter looks at evaluation metrics for two types of machine learning models:

- Regression
- Classification

Regression Tasks Evaluation Metrics:

- **Mean Squared Error (MSE):** This is the average of the squared differences between the predicted and actual values. It measures the average squared deviation of the predicted values from the true values.
- **Mean Absolute Error (MAE):** This is the average of the absolute differences between the predicted and actual values. It measures the average absolute deviation of the predicted values from the true values.
- **R-squared (R2):** This is a statistical measure that represents the proportion of variance in the target variable that is explained by the model. It ranges from 0 to 1, with 1 indicating a perfect fit.

Classification Tasks Evaluation Metrics:

Numerical Metrics:

- **Accuracy:** It is the ratio of the number of correctly classified instances to the total number of instances. While this is a useful metric, it can be misleading in cases where the class distribution is imbalanced.
- **Precision:** It is the ratio of true positives (TP) to the sum of true positives and false positives (FP). It measures the fraction of positive predictions that are correct.
- **Recall:** It is the ratio of true positives (TP) to the sum of true positives and false negatives (FN). It measures the fraction of actual positives that are correctly predicted as positive.
- **F1-score:** It is the harmonic mean of precision and recall. It combines both precision and recall into a single metric, which is useful when the classes are imbalanced.
- **Confusion Matrix:** A matrix of actual vs predicted labels of the model. It gives an overview of all the predictions made by the model and helps understand the types of errors made by the model.

Curve Plots:

- **Receiver Operating Characteristic (ROC) curve:** It plots the true positive rate against the false positive rate for various decision thresholds.
- **Area Under the Curve AUC:** is a useful metric because it measures the classifier's performance across all possible decision thresholds. It can also be used to compare the performance of different classifiers, even when the class distributions are imbalanced.

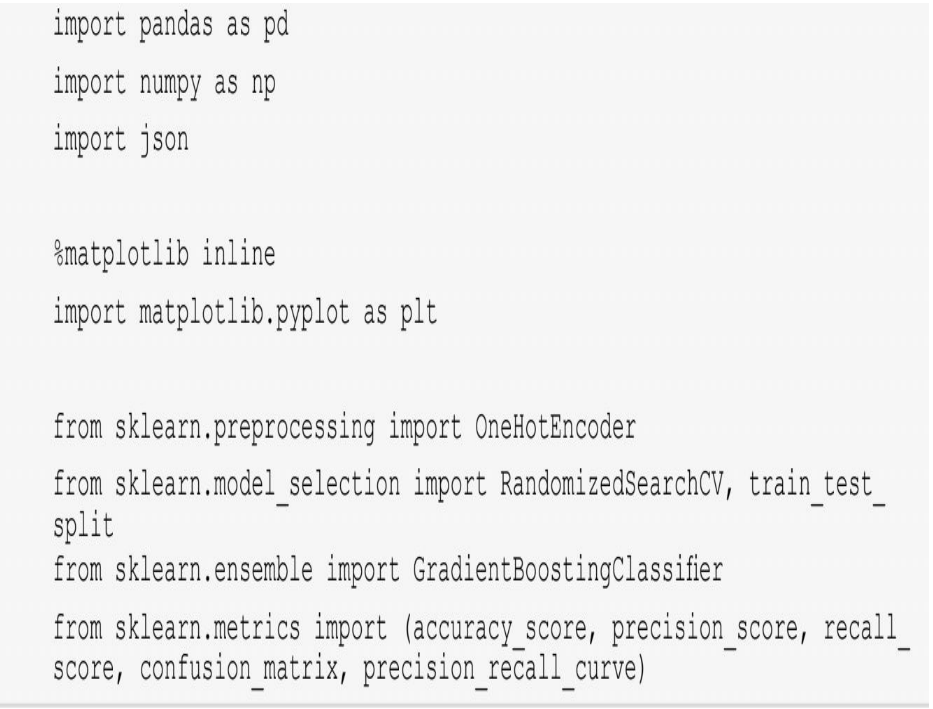
Use Case Presentation and Problem Definition

The purpose of exercise 7.01 is to evaluate a binary classification machine learning model to see if it can predict whether a person has breast cancer or not and basically diagnose breast cancer based on a number of medical features (the size of the tumor, the shape of the cells…etc)

The model is trained on a dataset with known outcomes (whether or not the person had breast cancer), and uses a technique called gradient boosting to make predictions. The process involves tuning the hyperparameters of the model to improve its accuracy, and test the model on a validation set to ensure it reacts well to new data it hasn't seen before and that is different than what it was trained on.

This exercise's goal is to maximize recall, which is the proportion of true positives (malignant cases correctly identified as malignant) over the total number of actual positives (all malignant cases). This is because in a medical diagnosis, it is more important to minimize false negatives (malignant cases identified as benign) and catch all instances of cancer, even if it means accepting some false positives (benign cases identified as malignant).

1. Import the relevant libraries



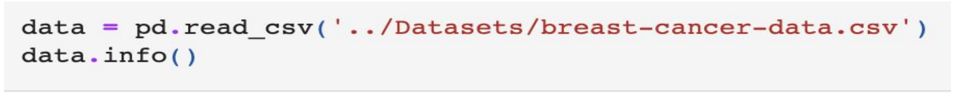
```
import pandas as pd
import numpy as np
import json

%matplotlib inline
import matplotlib.pyplot as plt

from sklearn.preprocessing import OneHotEncoder
from sklearn.model_selection import RandomizedSearchCV, train_test_
split
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.metrics import (accuracy_score, precision_score, recall_
score, confusion_matrix, precision_recall_curve)
```

‘Pandas' is a popular Python library for data manipulation, offering flexible and powerful data structures for analyzing data. Allows you to refer to it with the shortcut pd in the code
‘NumPy' This line imports the NumPy library and assigns the alias "np" to it.. He is a Python library for scientific computing, providing functions for performing mathematical and statistical operations on data arrays.
‘JSON’ is a Python library for working with JSON data, a commonly used data format for exchanging data on the web. »

"%matplotlib inline" will display plots directly in the notebook.
"matplotlib.pyplot" will allow you to create various types of plots and visualize data.

‘OneHotEncoder’ can be used to perform one-hot encoding on categorical variables.
‘RandomizedSearchCV’ will perform a random search to find the best hyperparameters for a machine learning model.
‘train_test_split’ will divide data into a training set and a test set to evaluate the performance of a model.
‘GradientBoostingClassifier’ will build a gradient boosting classification model.
‘accuracy_score, precision_score, recall_score, confusion_matrix, precision_recall_curve’ will evaluate the performance of a classification model by calculating metrics such as accuracy, precision, recall, confusion matrix, precision-recall curve, etc.

2. Read the breast-cancer-data.csv dataset

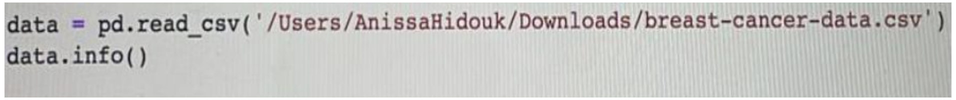


```
data = pd.read_csv('../Datasets/breast-cancer-data.csv')
data.info()
```

data = pd.read_csv('breast-cancer-data.csv') reads a CSV file named "breast-cancer-data.csv" using the pandas library and stores it in a DataFrame object named "data".

data.info() calls the **info()** method on the DataFrame object data. This method provides a concise summary of the DataFrame, including the data type of each column, the number of non-null values in each column, and the memory usage of the DataFrame.

Overall, this code snippet reads the data from a CSV file and provides a summary of the data using the .info() method.



```
data = pd.read_csv('/Users/AnissaHidouk/Downloads/breast-cancer-data.csv')
data.info()
```

I had to download the breast cancer data file from the GitHub datasets . Then, in my code line, I specified the user and session name. After that, we add 'Downloads' to tell the software which category of the computer it should look for the file. Then we add the file name for it to understand which one to use. And the software provide us an output.

2. Read the breast-cancer-data.csv dataset

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 31 columns):
mean radius                569 non-null float64
mean texture               569 non-null float64
mean perimeter             569 non-null float64
mean area                  569 non-null float64
mean smoothness            569 non-null float64
mean compactness           569 non-null float64
mean concavity             569 non-null float64
mean concave points        569 non-null float64
mean symmetry              569 non-null float64
mean fractal dimension     569 non-null float64
radius error               569 non-null float64
texture error              569 non-null float64
perimeter error            569 non-null float64
area error                 569 non-null float64
smoothness error           569 non-null float64
compactness error          569 non-null float64
concavity error            569 non-null float64
concave points error       569 non-null float64
symmetry error             569 non-null float64
fractal dimension error    569 non-null float64
worst radius               569 non-null float64
worst texture              569 non-null float64
worst perimeter            569 non-null float64
worst area                 569 non-null float64
worst smoothness           569 non-null float64
worst compactness          569 non-null float64
worst concavity            569 non-null float64
worst concave points       569 non-null float64
worst symmetry             569 non-null float64
worst fractal dimension    569 non-null float64
diagnosis                  569 non-null object
dtypes: float64(30), object(1)
memory usage: 137.9+ KB
```

The output:

- The first line of output shows that the DataFrame data has 569 rows and 33 columns. The RangeIndex indicates that the index of the DataFrame is a range of integers from 0 to 568.

- The second line shows the column names and their corresponding non-null count and data types. For example, the first column is named ID, has 569 non-null values, and is of type int64. The second column is named Diagnosis, has 569 non-null values, and is of type object.

- The last line shows the memory usage of the DataFrame.

3- Preparing Data for Machine Learning: Separating Input Features and Target Variables

This code prepares data for machine learning model training. It separates input features and target labels from a DataFrame called 'data'. The 'diagnosis' column is used as the target label, while the remaining columns are used as input features. The code creates a new DataFrame called 'X' by dropping the 'diagnosis' column, and maps the target labels to binary values. Finally, it returns the values of the target label as a numpy array, making the data compatible with many machine learning libraries. The resulting X and y can be used in a machine learning model by splitting them into training and testing sets, fitting a model on the training data, and evaluating its performance on the testing data.

4. Divide the dataset into train and validation DataFrames:

The dataset is split into training and test sets using the train_test_split function from the Scikit-learn library. The training set is used to train the machine learning model, while the test set is used to evaluate its performance.

5. Initialize dictionaries in which to store the train and validation MAE values:

A base model is chosen as a starting point, and a range of hyperparameter values is defined for each hyperparameter that will be tuned. In our exercise, a gradient-boosted classifier is chosen as the base model. A dictionary called param_dist is created, which specifies the ranges of values for each hyperparameter that will be searched during hyperparameter tuning.

Hyperparameter tuning is a crucial step in machine learning since it helps in finding the best set of hyperparameters that will result in the highest performance of the model on the test set.

6-Finding the Best Hyperparameters for a Gradient Boosting Classifier Model using Randomized Search Cross-Validation

This task aims to find the best set of hyperparameters for a Gradient Boosting Classifier model. To do so, the Randomized Search Cross-Validation technique will be used, which involves randomly selecting hyperparameters from a defined range and testing them using cross-validation. The steps include defining a dictionary of hyperparameters to test, creating a dictionary of parameters for the RandomizedSearchCV object, creating an instance of the RandomizedSearchCV object, and fitting it with training data. This process can help find the best set of hyperparameters for the GBC model to improve its performance on test data.

7 - Data Preparation Technique for Machine Learning: Mapping Target Labels and Extracting Input Features

"When creating a machine learning model, splitting the original dataset into training and validation sets is crucial for evaluating the model's performance and ensuring its ability to generalize well to new data. The scikit-learn train_test_split() method splits the data into random training and validation sets based on a specified test_size parameter.

In this example, the validation set comprises 15% of the original dataset, and a random_state parameter is set to ensure reproducibility. After splitting the data, a new model is trained using the final hyperparameters determined during hyperparameter tuning, using only the training set. The model's performance is then evaluated on the validation set to verify its generalizability to new data."

8.Training a classification model by gradient reinforcement and the probability regarding validation set

Training a gradient reinforcement classification model involves, adjusting the parameters of the model so that it can learn to associate features of the data with their corresponding class labels. The "hyperparameters" are the parameters of the model that are not learned from the data, but that must be specified before training,

in this exercise,it means that we need to fit a gradient boosting classification model using the final hyperparameters chosen, and then use this model to make predictions on the training and validation sets created. The final objective is to use a model to make predictions on the validation set, and then calculate the probability of each prediction to be correct.

9.Model Performance Metrics and Evaluation in Classification

This exercise is used to see if our model is effective through accuracy, pressure and recall. So, we evaluate the performance of a classification model to determine its accuracy and effectiveness. There are several common measures used to evaluate a classification model, including accuracy, precision, and recall.

Accuracy is the measure of the proportion of correctly classified samples compared to the total number of samples. Precision is the proportion of true positives among all samples classified as positive.
Recall is the proportion of true positives among all samples that are actually positive.
The confusion matrix is a table that visualizes misclassification by showing the number of true positives, false positives, true negatives and false negatives for each predicted class.

These measures and the confusion matrix are useful for understanding the strengths and weaknesses of the model and for improving it if necessary because it helps to understand how the model classifies the samples and where it goes wrong.

10. Experiment with varying thresholds to find the optimal point having a high recall

The goal here is to explore the precision-recall trade-off in binary classification tasks and to find the optimal threshold for a given model that maximizes recall while maintaining an acceptable precision level.

We are expected to experiment with different thresholds and plot the precision-recall curve to visualize the trade-off between precision and recall. Then, we have to identify the threshold that gives the highest recall while maintaining a minimum precision level, and report the corresponding recall, precision, and F1 score values. This helps to evaluate the performance of a binary classification model and make informed decisions about the threshold setting for optimal results.

11.Finalize a threshold that will be used for predictions in relation to the test dataset:

It is about choosing the threshold value that determines how the model classifies predicted probabilities into binary classes. The threshold value should be chosen based on the performance of the model on the validation dataset, taking into account the precision-recall trade-off.

Now, The chosen threshold will be used to make predictions on the test dataset. It should maximize recall while maintaining an acceptable precision level, or be chosen based on a specific business objective or application requirement. Once the threshold is chosen, it can be applied to the predicted probabilities of the test dataset to obtain the final binary predictions.

12. Predicting the final value based on the test dataset to determine classes using our threshold value:

The objective here is to demonstrate the process of finalizing a threshold value for a binary classification model and applying it to make predictions on a test dataset. The process involves training and evaluating the model, exploring different threshold values, and selecting the optimal threshold based on the precision-recall trade-off and business objective. Once the threshold value is chosen, it is applied to the predicted probabilities of the test dataset to obtain the final binary predictions, which should be saved to a file in the CSV. (*final_predictions.csv)*

The ultimate goal is to use the model to predict the binary class of new, unseen data, and assess its ability to generalize using standard evaluation metrics.

Conclusion

In conclusion, this chapter explains that training a model and assessing its accuracy is essential, but it also teaches us that it is aslo important to ensure that the model's performance is representative and that it can accurately predict outcomes on unseen data. So, evaluating and improving a model's performance is crucial to make sure it is the best version of itself. This chapter also provides essential techniques for evaluating and improving a model's performance to make sure that it can make accurate predictions based on the data it has learned.

Software and Datasets

The softwares we used were:

- Anaconda Navigator
- Jupiter Notebook

Datasets can be found here: https://packt.live/2T1fCWM.