Data Camp Cheat Sheet

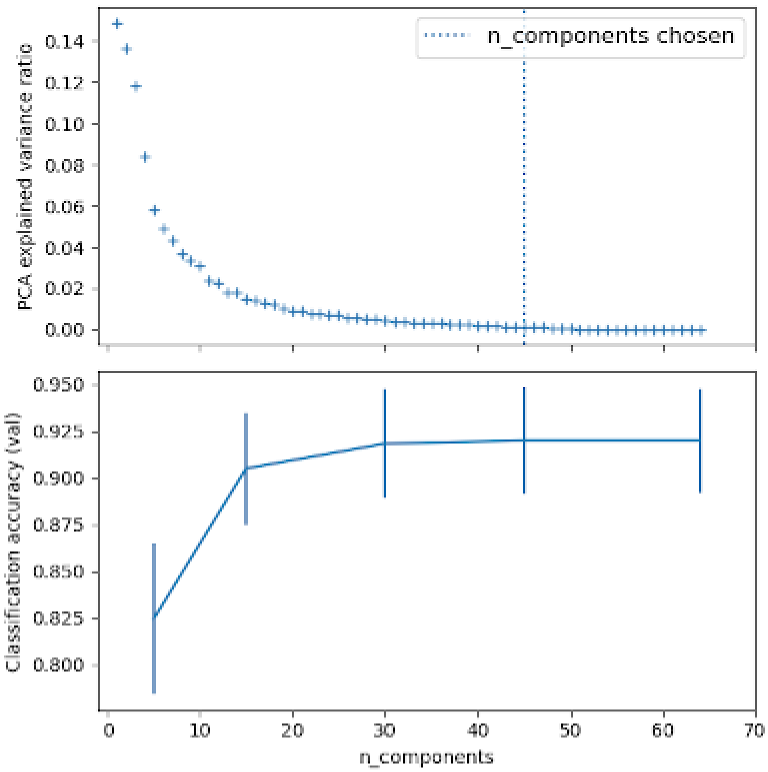
Techniques

Here you can find various ideas of methods to try to solve the problem. Most of them can be combined and can increase - or decrease - your final score. Think of this document as a kind of general machine learning (ML) cheat sheet.

To evaluate the benefit of those methods it is recommended to:

- Use a hold-out set or a cross-validation method and report the variance
- Display the scores with Visualization (plots)
- Compare scores with only one component varying (e.g., different models but same pre-processing)

Here is an example of the kind of experiments you can put in place to improve your model.



In this plot, you can see the accuracy obtained by an algorithm on the validation set for different numbers of dimensions (dimensionality reduction method used: PCA).

1. Processing

/!\ Don't forget to apply the same pre-processing to your train and test sets!

1.1 Missing values

If you have missing values (NaN) in your data, you must remove them or replace them. The main methods are:

- Remove rows or remove columns that contain NaN
- Replace NaN with a value. It can be a specific value such as 0 or a computed value from the other values in the same row or column (mean, median, most occurring value, etc.)
- Using a ML model to infer the value given neighbor values

Here is a simple code to replace the missing values from your `pd.DataFrame` object:

```
xs = xs.fillna(method="pad")
xs_test = xs_test.fillna(method="pad")
```

1.2 Encoding

Encodings are useful to be able to give categorical variables as an input of your ML algorithm. Their main purpose is to transform **strings into integers**.

Some ideas of encoding:

- Ordinal encoding (or label encoding)
- Frequency encoding
- Likelihood encoding
- Target encoding
- One-hot encoding (*WARNING: it creates a new variable for each category and therefore can be very memory consuming!*)
- Deep learning embedding
- ...

Here is a piece of code useful to apply **ordinal encoding** to your data. As you can see, it is important to store the `str -> int` mapping with a dictionary to apply it to another subset of the dataset.

```
class OrdinalEncoder():
    """ Ordinal encoding.
        ["fish", "cat", "fish", "dog"] -> [0, 1, 0, 2]
    """
    def __init__(self):
        # Dictionnary containing the saved mapping from String to Integers
        self.mapping = {}

    def fit(self, variable):
        """ Fit the encoding from a categorical variable.
            :param variable: 1D vector (list, np.array, pd.Series)
        """
        i = 0
        for e in variable:
            if e not in self.mapping:  # This category is not encoded
                self.mapping[e] = i
                i += 1

    def transform(self, variable):
```

```
        """ Apply the encoding to a categorical variable.
            :param variable: 1D vector (list, np.array, pd.Series)
        """
        i = len(self.mapping)
        encoded = variable.copy()
        for j, e in enumerate(variable):
            if e not in self.mapping:  # Unseen category
                self.mapping[e] = i
                i += 1
            encoded[j] = self.mapping[e]
        return encoded
```

To use it on your dataset:

```
X_encoded = X.copy()
X_test_encoded = X_test.copy()

str_columns = X.select_dtypes(include="object")

for column in str_columns:
    encoder = OrdinalEncoder()
    encoder.fit(X[column])
    X_encoded[column] = encoder.transform(X[column])
    X_test_encoded[column] = encoder.transform(X_test[column])
```

HINT: You can treat separately the variables representing timestamps (e.g., convert them into seconds).

1.3 Normalization

You can try to normalize your features.

Some ideas of normalization:

- Min-max normalization
- Standard normalization

Here is more information on mix-max normalization

1.4 Feature engineering

You can transform your features, reduce their numbers, or even create new ones.

A) Feature selection

You can reduce the number of features of your data by removing some features (feature selection). Some ideas of feature selection methods: Remove by hand, remove the variables with the worst Pearson's correlation with target, Lasso, tree-based method, recursive feature elimination.

B) Dimensionality reduction

You can reduce the number of features of your data by combining them (dimensionality reduction). Some ideas of dimensionality reduction methods: PCA, LDA, autoencoder, T-SNE, feature hashing.

Here is the PCA from the scikit-learn package.

/!\ Don't forget to vary the number of dimensions to see what works best!

C) Hand-made features

You can combine features or create new ones using your expert knowledge of the problem.

Examples:

- (Image classification) The sum of blue pixels in the image
- (Image classification) The mean color of all pixels
- (Medical record) The difference between "departure time" and "admission time"

D) More advanced transformation

Use deep learning models to process data.

Example: train a convolutional neural network (CNN) to solve your supervised learning problem, then keep the first layers as an embedding. Some similar ideas can also be performed in an unsupervised learning setting (e.g., using autoencoder).

1.5 Up-sampling and down-sampling

If the label we try to predict is imbalanced.

To reduce or suppress the imbalance and **help your model in its training**, you can do:

- Up-sampling: re-sample (with replacement) new points from the least represented class
- Down-sampling: keep only a subset of most represented classes

This simple preprocessing can be very effective.

2. Model

2.1 Model selection and hyperparameter selection

Last but not least, you need to choose a ML model. Each model has a set of hyperparameters that influence its behavior and therefore its performance.

Ideas of models:

- Logistic Regression
- KNN
- Random Forest
- SVM

- Multi-layer perceptron
- CNN
- ... *(there are tons of models)*
- Combine several with an
- ensemble of models

/!\ Don't forget to read about the models' hyperparameters and to make their values vary!

It is always good to use Cross Validation techniques to compute error bars and validate your models.

2.2 Automated machine learning

We've recently put in place an AutoML "Self-Service".

You can try to use it to solve your problem.

- ***Warning 1**: nothing guarantees that the AutoML model will perform well on your task*
- ***Warning 2**: converting your data in the format needed by this service may be difficult*

3. Visualization

Let's divide the visualizations into two groups:

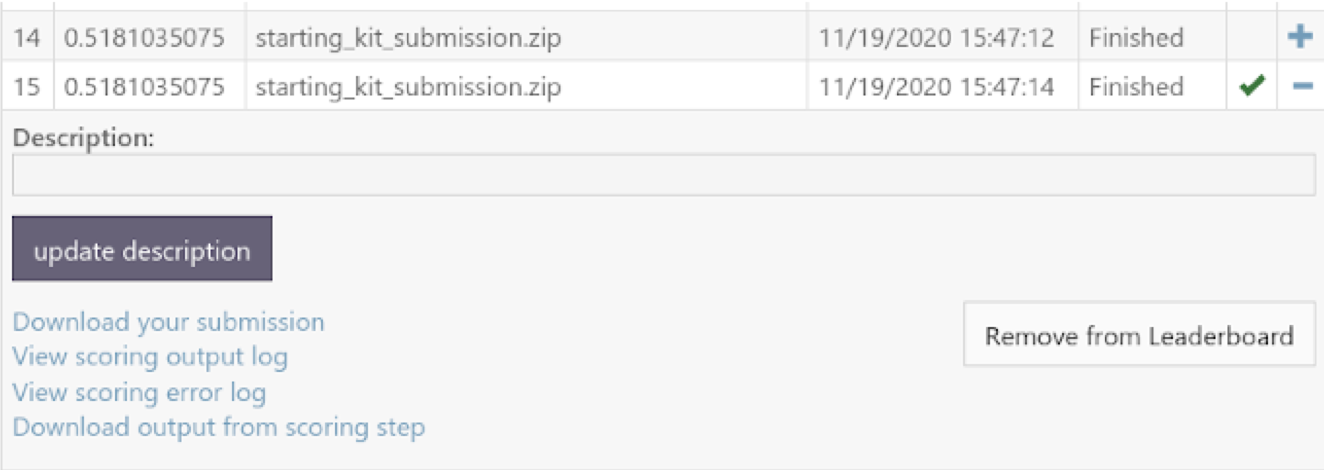
- **Exploratory analysis:** to have a better understanding of the data. Some examples:
  - Show a specific variable with a simple graph or bar plot
  - Show the relationship between two variables with a scatter plot
  - Show your data in 2D or 3D using a dimensionality reduction algorithm
  - Show the correlation between your features and the target variable
  - Show the distribution of the target variable
  - A heatmap of the dataset
  - Try to think of your own visualizations that could be interesting for your specific dataset
- **Results analysis:** to understand your models' behaviors. Some examples:
  - Confusion matrix
  - Compare several models with a box plot
  - Show the variation of the score according to a hyperparameter
  - Visualize your models' output
  - Try to think of your own interesting visualizations

The most common Python library for visualizations is Matplotlib. You can also try Seaborn which is based on Matplotlib. Here are examples at the Seaborn's gallery.

Submit to Codalab

- **You must submit to Codalab**
- Create your submission file using the Jupyter notebook. You need to have a file named "mimic_synthetic_test.csv" inside a ZIP archive.
- Go to `Participate` > `Submit / View Results` and click on `Submit` button to upload your submission.

- If your submission fails, have a look at the scoring output and scoring error logs! Click on the cross "+" on the right of the submission interface (see the image below).



- If the error states `Bad prediction shape: (20002, 1) != (20001, 1)` it means that your `.csv` file is one line too long. Each line has to be one prediction (but the notebook's code should do that for you). Maybe it's saving it with a header, **check the first line** of your file, and add `header=False` to the `to_csv` method.

Computing power and Jupyter Hub

You can try out IJCLab's Jupyter Cloud if you need more computing power.