LDSSA

SLU16 Workflow

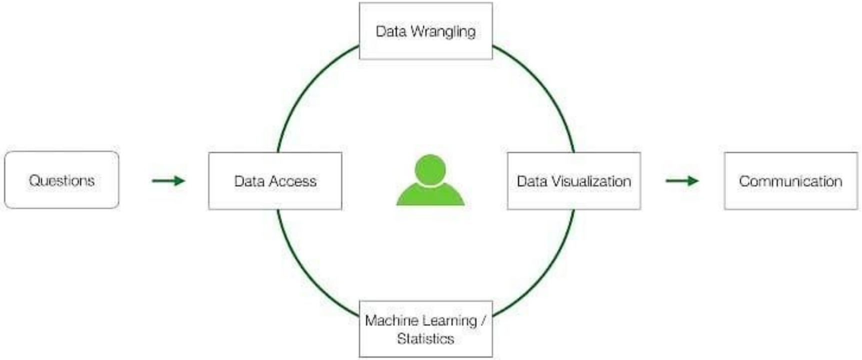
December 3rd, 2023

1. Introduction

Motivation

- **Data Science** is largely an **engineering discipline**. Writing code is an engineering practice and data science is mostly done with code.

- **A healthy respect for the engineering element of data science is key**. It will make your life an order of magnitude easier, and more importantly, it makes your **data science more responsible**.



Questions → Data Access

Data Wrangling

Data Visualization → Communication

Machine Learning / Statistics

Overview

In this **SLU** we will be covering the following:

- Workflow
- Pipelines and Custom Objects
- Workflow tips tricks
- Best practices

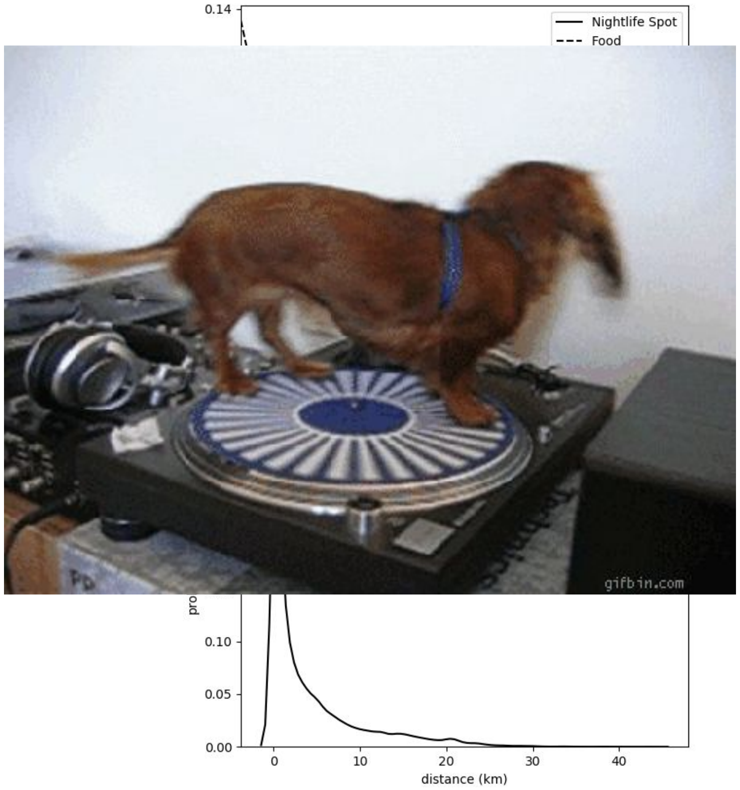
2. Topic Explanation

| Workflow |

Workflow

Specialization #1 - SLU04 & SLU05

1. Get the Data
2. Data Analysis and Preparation
    a. Data analysis
    b. Dealing with data problems
    c. Feature engineering
    d. Feature selection
3. Train Model
4. Evaluate Results
5. Iterate



| Pipelines and Custom Objects |

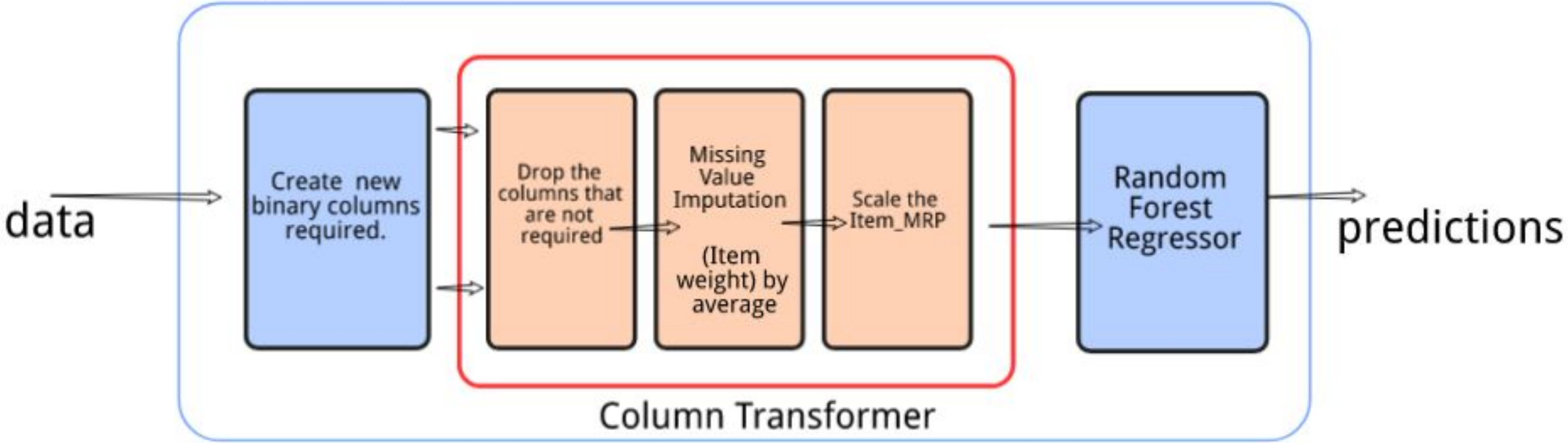
Sklearn Pipeline

- “Sequentially apply a list of transforms and a final estimator. “

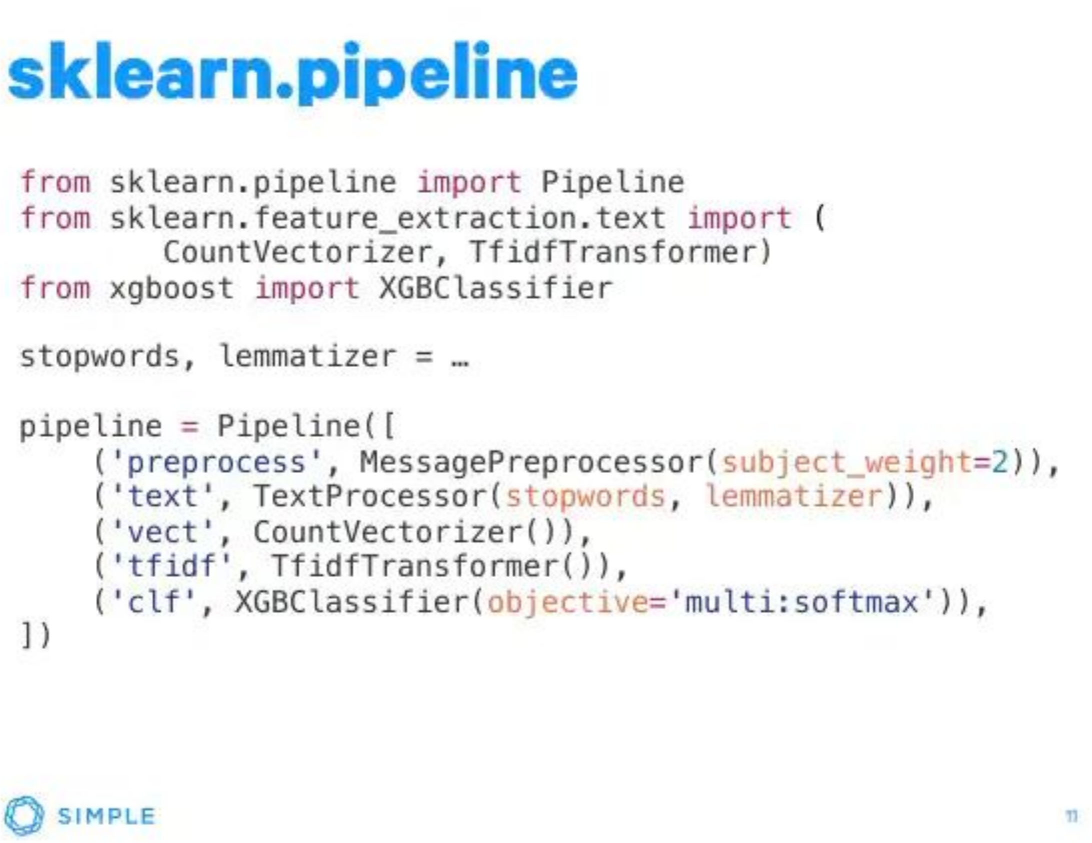
- It implements the same API as the models (has`predict` and/or`predict_proba`) but it applies each of the steps before calling the model with the input!

pipeline.fit() when using the training data

pipeline.predict() on test data



Pipeline



sklearn.pipeline

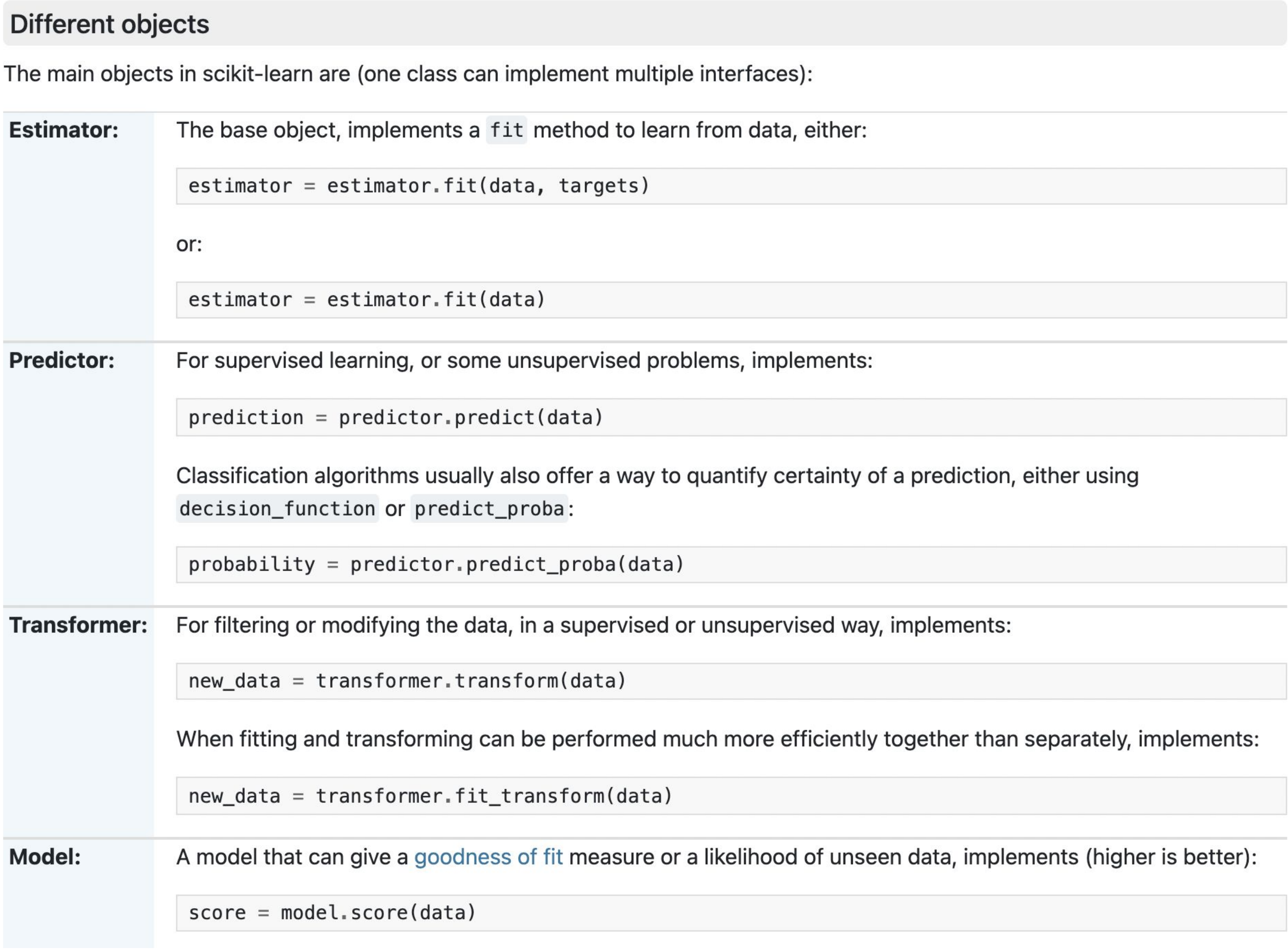
```python
from sklearn.pipeline import Pipeline
from sklearn.feature_extraction.text import (
        CountVectorizer, TfidfTransformer)
from xgboost import XGBClassifier

stopwords, lemmatizer = …

pipeline = Pipeline([
    ('preprocess', MessagePreprocessor(subject_weight=2)),
    ('text', TextProcessor(stopwords, lemmatizer)),
    ('vect', CountVectorizer()),
    ('tfidf', TfidfTransformer()),
    ('clf', XGBClassifier(objective='multi:softmax')),
])
```

SIMPLE

11

Data

Classification/ Prediction



Different objects

The main objects in scikit-learn are (one class can implement multiple interfaces):

Estimator: The base object, implements a `fit` method to learn from data, either:

```
estimator = estimator.fit(data, targets)
```

or:

```
estimator = estimator.fit(data)
```

Predictor: For supervised learning, or some unsupervised problems, implements:

```
prediction = predictor.predict(data)
```

Classification algorithms usually also offer a way to quantify certainty of a prediction, either using `decision_function` or `predict_proba`:

```
probability = predictor.predict_proba(data)
```

Transformer: For filtering or modifying the data, in a supervised or unsupervised way, implements:

```
new_data = transformer.transform(data)
```

When fitting and transforming can be performed much more efficiently together than separately, implements:

```
new_data = transformer.fit_transform(data)
```

Model: A model that can give a goodness of fit measure or a likelihood of unseen data, implements (higher is better):

```
score = model.score(data)
```

| Workflow tips tricks |

1. Establish a simple baseline FAST
2. Incrementally increase complexity
3. Use (and abuse) Scikit pipelines & Custom transformers

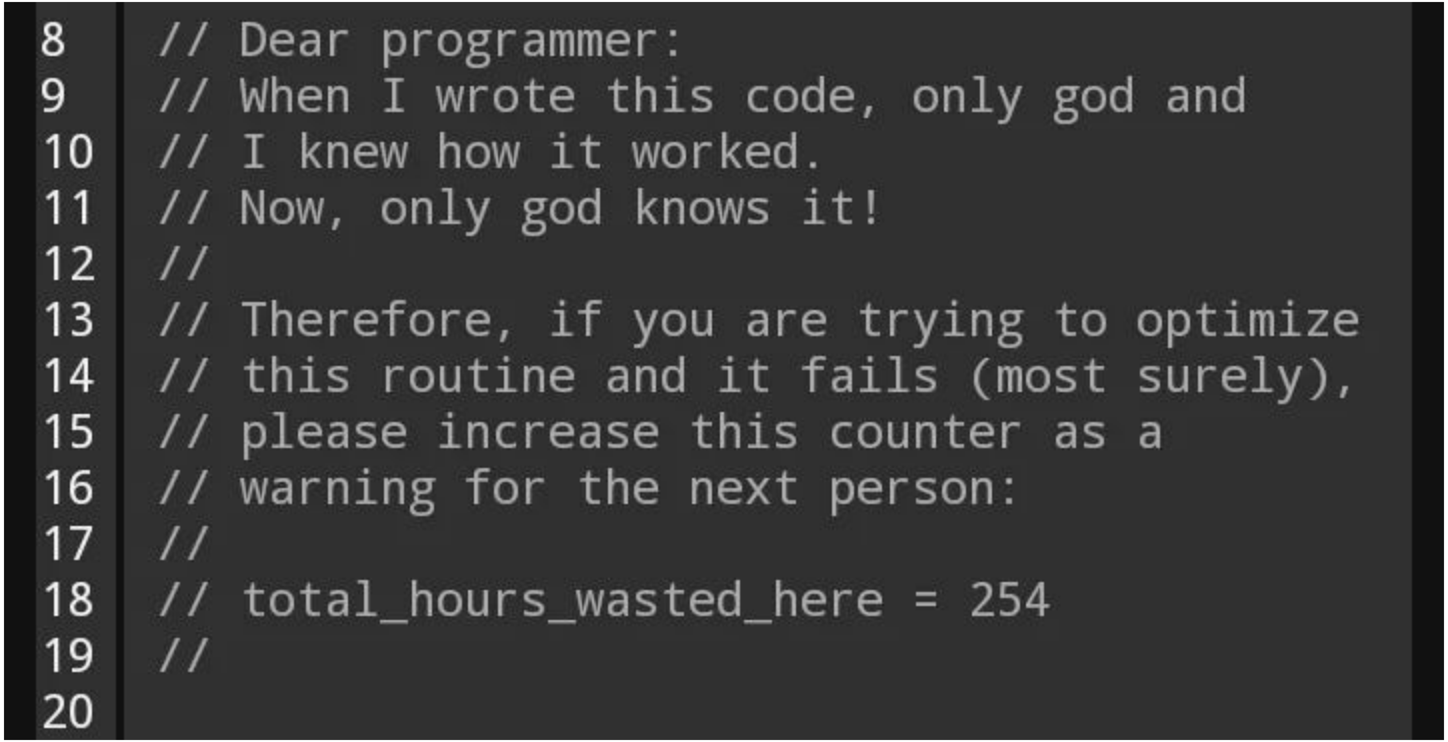
| Best Practises |

- **Use well-named variables**
- **Use functions!**
  - If you take a parameter that is a dataframe, always name if _df`
  - Outside of functions, never name a dataframe _df`
  - Inside of functions, never name a dataframe `df`
- **Immutability is key!**
  - Never use `inplace=True`
  - Always copy a dataframe at the beginning of the function and make your changes to that copy.

- **Imports up top!**
- **Organize your directory**
- **When writing functions**
    - Keep them close to the cell where you are testing it at first
    - When they are stabilizing, move them to the top of the notebook
    - When you are starting to use them a lot, move them into a utils.py file and import from there
- **“Restart and Run All”**
- **Each experiment should be runnable in a single cell**

3. Recap

- “First do it, then do it well, then do it better”
- Be organized - keep track of the improvements you need/want to do
- Try to use Scikit Pipelines whenever you can
- Make sure your code is readable not just to you, but to others!



```
// Dear programmer:
// When I wrote this code, only god and
// I knew how it worked.
// Now, only god knows it!
//
// Therefore, if you are trying to optimize
// this routine and it fails (most surely),
// please increase this counter as a
// warning for the next person:
//
// total_hours_wasted_here = 254
//

```

4. Q&A