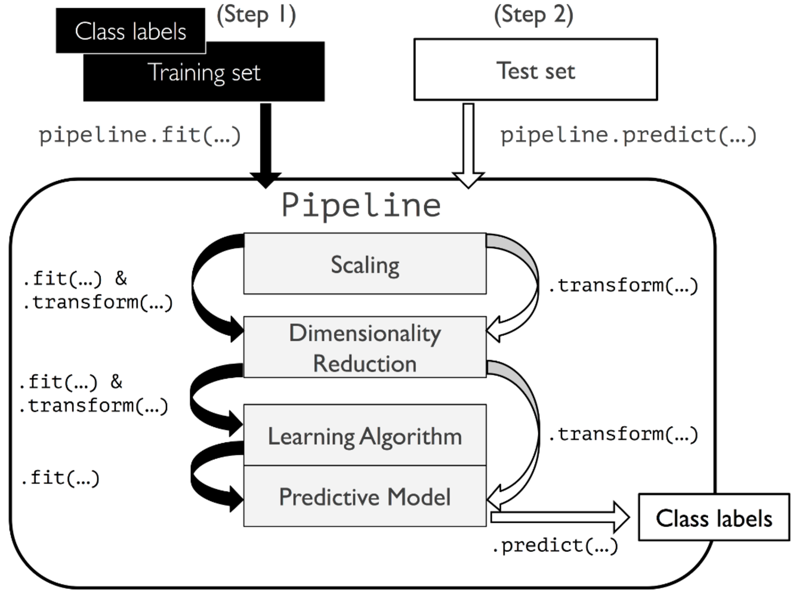
**Model Pipeline**

Pipeline is a tool that simplifies and automates the process of building a machine learning workflow.

* It allows for the **sequential execution of multiple steps**, combining **data preprocessing** and the **final model training or prediction** into a single and streamlined process.
* Sequential chain of **transformers** (e.g., preprocessors like scaling, encoding, or feature extraction) followed by an **estimator** (e.g., a classifier or regressor) 🡺 ensures that the entire workflow is applied consistently and efficiently.
* Each step in the pipeline performs its transformation and passes the output to the next step.
  + **Transformers**:
    - Apply ***.fit\_transform()*** during **training** and ***.transform()*** during **testing**.
  + **Estimator**:
    - Applies ***.fit()*** during **training** and ***.predict()*** during **testing**.
* **Workflow**



Binary classification with numeric features

A screen shot of a computer code

Description automatically generated

A black text on a white background

Description automatically generated

* The goal is to predict the binary classification with building the pipeline.

**Components of a pipeline**

1. **Steps**:

* Each step is a tuple of a name and an object (transformer or estimator) 🡺 ('scaler', StandardScaler()).

A black text on a white background

Description automatically generated

1. **Preprocessing**:

* Steps like scaling, encoding, or imputing missing values.
* Must implement ***.fit()*** and ***.transform()*** methods.

1. **Estimator**:

* The final step of the pipeline is the machine learning model.
* Must implement ***.fit()*** and ***.predict()*** methods.

A close-up of words

Description automatically generated

1. Hyperparameter tuning with pipeline

* Use “\_ \_” for each specific model for the hyperparameter tuning.
* Apply pipeline with dictionary of the hyperparameters using ***.fit(X\_train, y\_train)***

A screenshot of a computer code

Description automatically generated

**Workflow of pipeline in binary task using *Pipeline([ ])***

A screen shot of a computer code

Description automatically generated

* **'scale'**: Applies StandardScaler to standardize the features by removing the mean and scaling to unit variance.
* **'pca'**: Uses PCAto reduce the dimensionality of the dataset to 15 components.
* **'lr'**: Fits a LogisticRegression model as the final estimator.
* Using ***.fit(train, test)*** to get the sense of the model prediction by log reg.

A screenshot of a number

Description automatically generated

* Use log reg model performance from pipeline (pipe1) to extract the PCs coefficients

A screenshot of a number

Description automatically generated

* Use PCA model from pipeline (pipe1) to get each PCs

**Hyperparameter tuning via Pipeline**

A screenshot of a computer code

Description automatically generated

* We need to specify the hyperparameter’s name with its corresponding model names with **“\_ \_”**
* Fit the pipeline (pipe1) with the specified hyperparameters to fit the X\_train and y\_train.
* Apply ***.score(X\_train, y\_train)*** to get the result.

**Pipeline display**

* By fitting the training data from the pipeline construction, we can visualize how each parameter inside the pipeline’s content and the workflow
* Order of each of the parameter inside the pipeline matters.

A screenshot of a computer

Description automatically generated

* Display the best estimator of each model inside the pipeline using ***.best\_estimator\_***

A screenshot of a computer program

Description automatically generated

**Column Transformer**

It allows you to apply different transformations to different subsets of columns, which is particularly useful when dealing with datasets that have mixed data types, such as categorical and numerical features.

A table with numbers and letters

Description automatically generated

Column Transformer components

* Selective Transformation:
  + Apply specific preprocessing steps to **specific columns**.
* Integration with Pipelines:
  + Can be **used as part of a scikit-learn pipeline**.
* Support for Mixed Data Types:
  + Handle **numeric and categorical** data differently.

Workflow of column transformer with pipeline

1. Apply data column with **numerical features** processing with specific columns

A screen shot of a computer code

Description automatically generated

* + Fill the missing value for columns age and fare with median with ‘imputer’
  + Standardize both columns with standard scaler with ‘scaler’

1. Apply another different column with **categorical features** with different processing method

A close-up of a computer code

Description automatically generated

* + Use simple imputing and one-hot encoding to process specified columns with ‘imputer’ and ‘onehot’

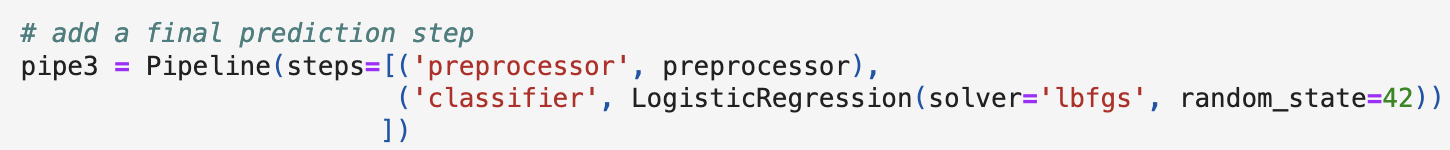
1. Combine both processing column transformer pipeline as preprocessor.

A computer code with black text

Description automatically generated

* + Combine the preprocessing with ColumTransformer

1. Set the final pipeline with the model training



1. Visualization of pipeline

A screenshot of a computer

Description automatically generated

1. Apply the data splitting for model performance

A computer screen shot of a computer code

Description automatically generated

* + We can use ‘survived’ as the binary target and all other features as the IVs
  + Apply train\_test\_split to split the data with X and y for model training

1. Connect with previous column transformation pipeline with split data using ***pipeline.fit (X, y)***

A screenshot of a computer code

Description automatically generated

* + With pipe3, we conducted all the column transformation with numerical and categorical feature transformation and apply to the splitted train and test dataset.

1. Hyperparameter tuning with pipeline

A screenshot of a computer code

Description automatically generated