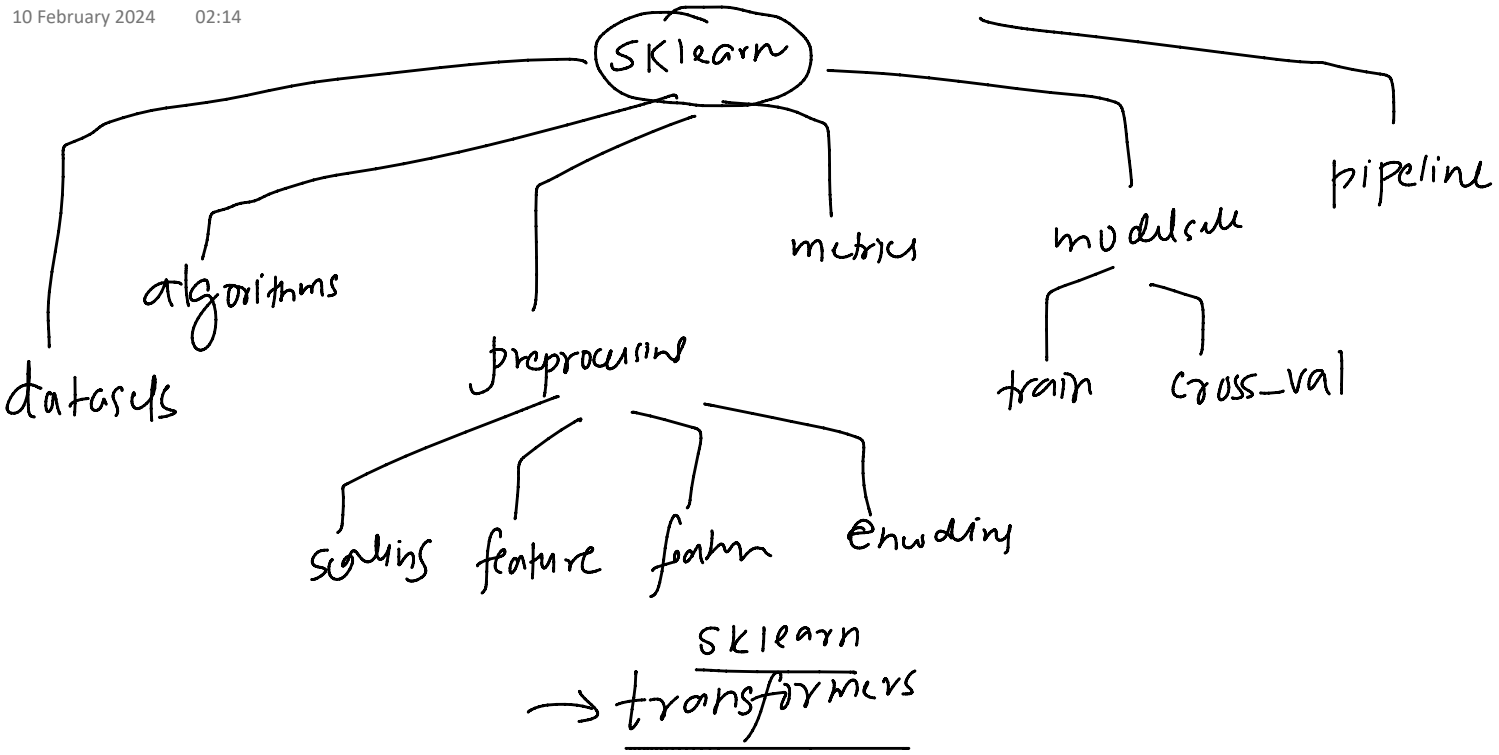
Plan of Attack

10 February 2024 02:14



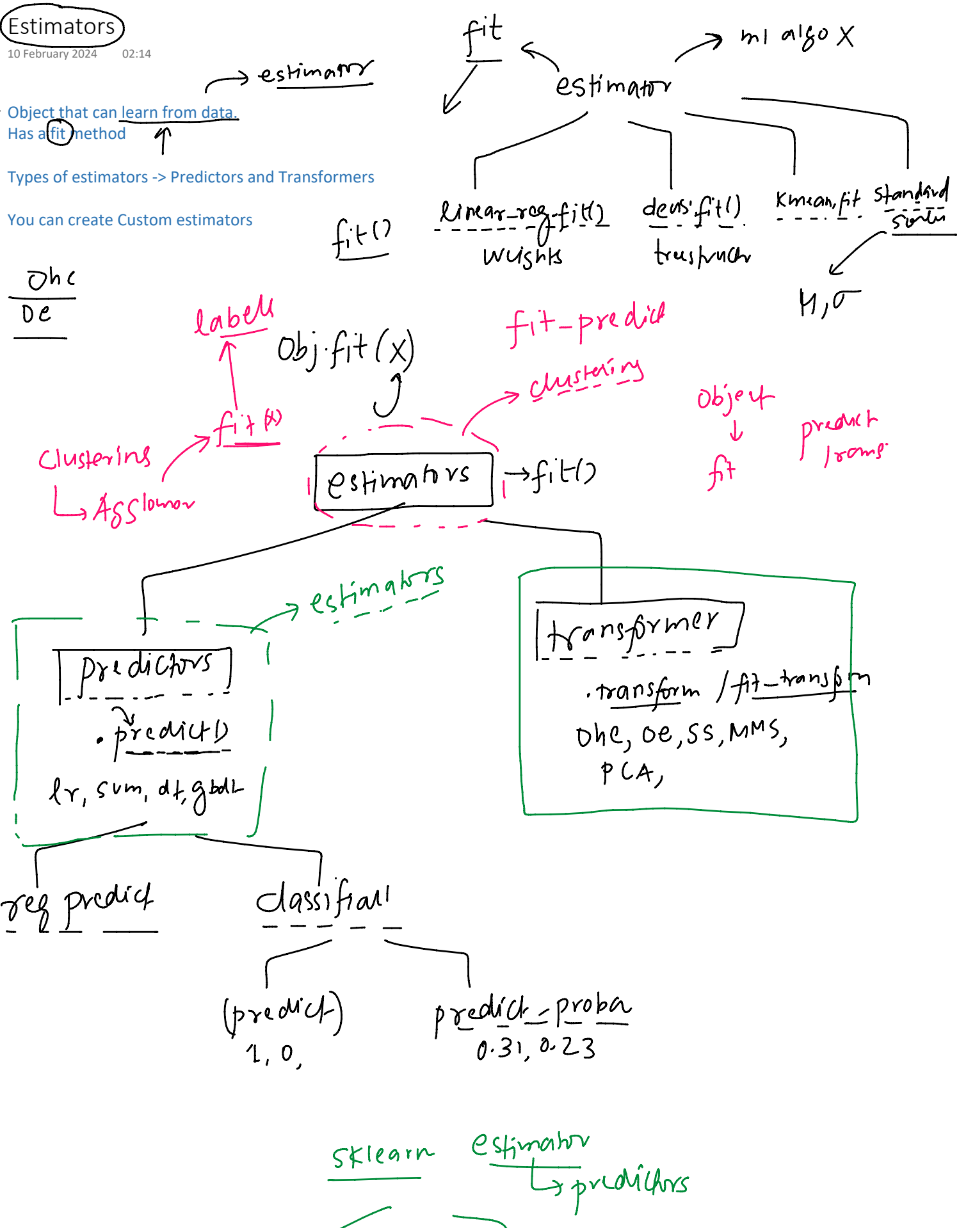
Estimators

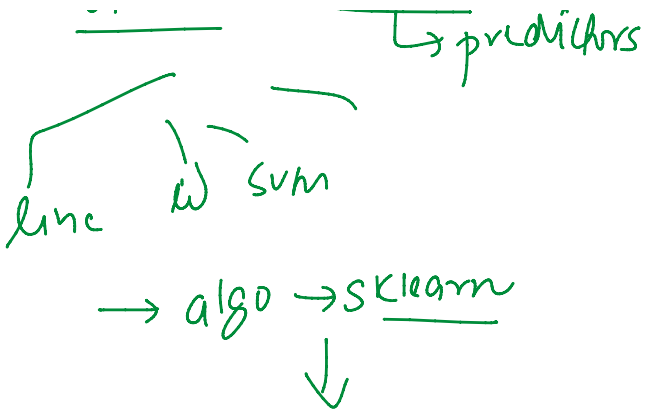
10 February 2024 02:14

→ Object that can learn from data.
Has a fit method

Types of estimators -> Predictors and Transformers

You can create Custom estimators




predictors
linc
svm
algo
sklearn

Custom Estimators

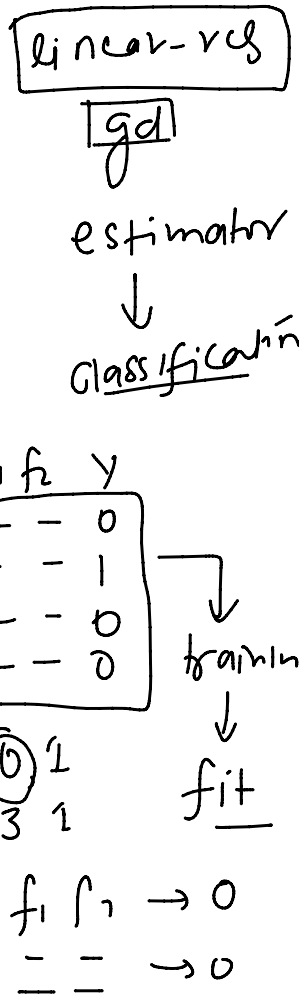
10 February 2024 12:47

Custom estimators in scikit-learn are user-defined classes that implement the scikit-learn estimator interface. These are crafted to fit specific modeling or pre-processing needs that are not directly met by the built-in estimators provided by scikit-learn. Custom estimators can range from simple modifications of existing algorithms to entirely new algorithms developed for unique tasks.

The creation of a custom estimator involves defining a Python class that adheres to certain conventions and implements required methods, ensuring it integrates seamlessly with scikit-learn's broader ecosystem, including pipelines, cross-validation tools, and hyperparameter tuning utilities.

Key Components of Custom Estimators:

1. Consistency with the Estimator Interface: At a minimum, a custom estimator must implement a fit method. If it's a transformer (for pre-processing), it should also implement a transform method, and if it's a predictor (for modeling), it should implement a predict method.
2. Inheritance from Base Classes: Custom estimators often inherit from base classes like BaseEstimator and optionally ClassifierMixin, RegressorMixin, or TransformerMixin, depending on their purpose. This inheritance brings in useful methods and ensures compatibility with scikit-learn's utilities.
3. Parameters and Initialization: Custom estimators should allow parameters to be set via the constructor (i.e., __init__ method), following the pattern of scikit-learn's estimators. Parameters should be explicitly declared in the __init__ method and stored as instance variables.
4. Parameter Validation: It's good practice to validate parameters and inputs, using scikit-learn's utilities like check_X_y for input validation and check_array for validating features.

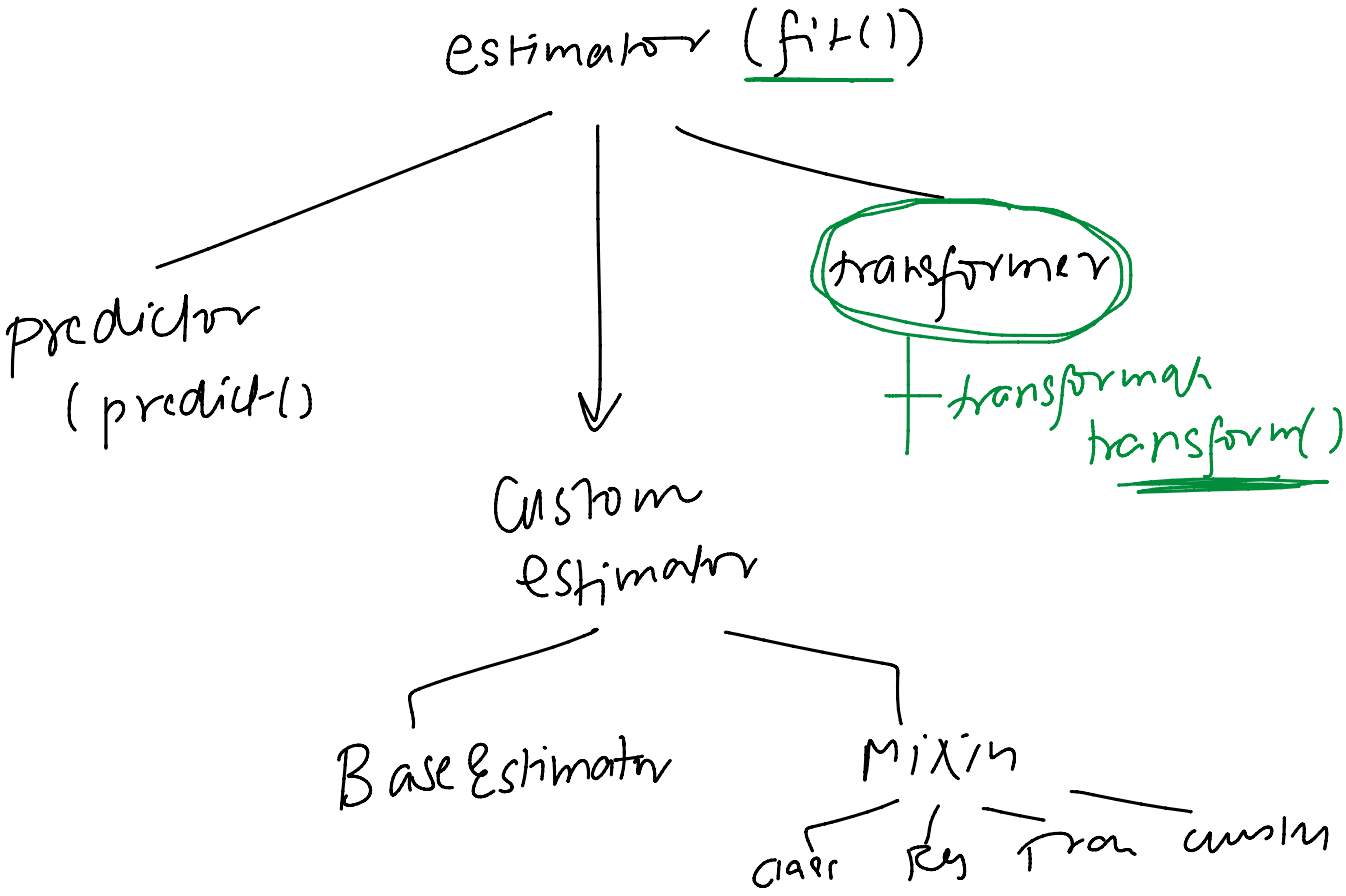


Mixins

10 February 2024 12:24

Mixin classes in scikit-learn are auxiliary classes that provide additional methods and functionality to custom estimators. They are designed to be used through multiple inheritance, allowing custom estimators to gain standardized capabilities, such as scoring, fitting, predicting, or transforming, without needing to reimplement these methods from scratch.

- Scoring: The ClassifierMixin and RegressorMixin provide a default implementation of the score method for classification and regression estimators, respectively.
- Transformation: The TransformerMixin provides a fit_transform method, allowing transformation estimators to both learn and apply a transformation in a single step.
- Fitting and Predicting: The ClusterMixin adds the fit_predict method to clustering estimators, enabling them to fit to the data and then return cluster labels.



Transformers

10 February 2024 02:14

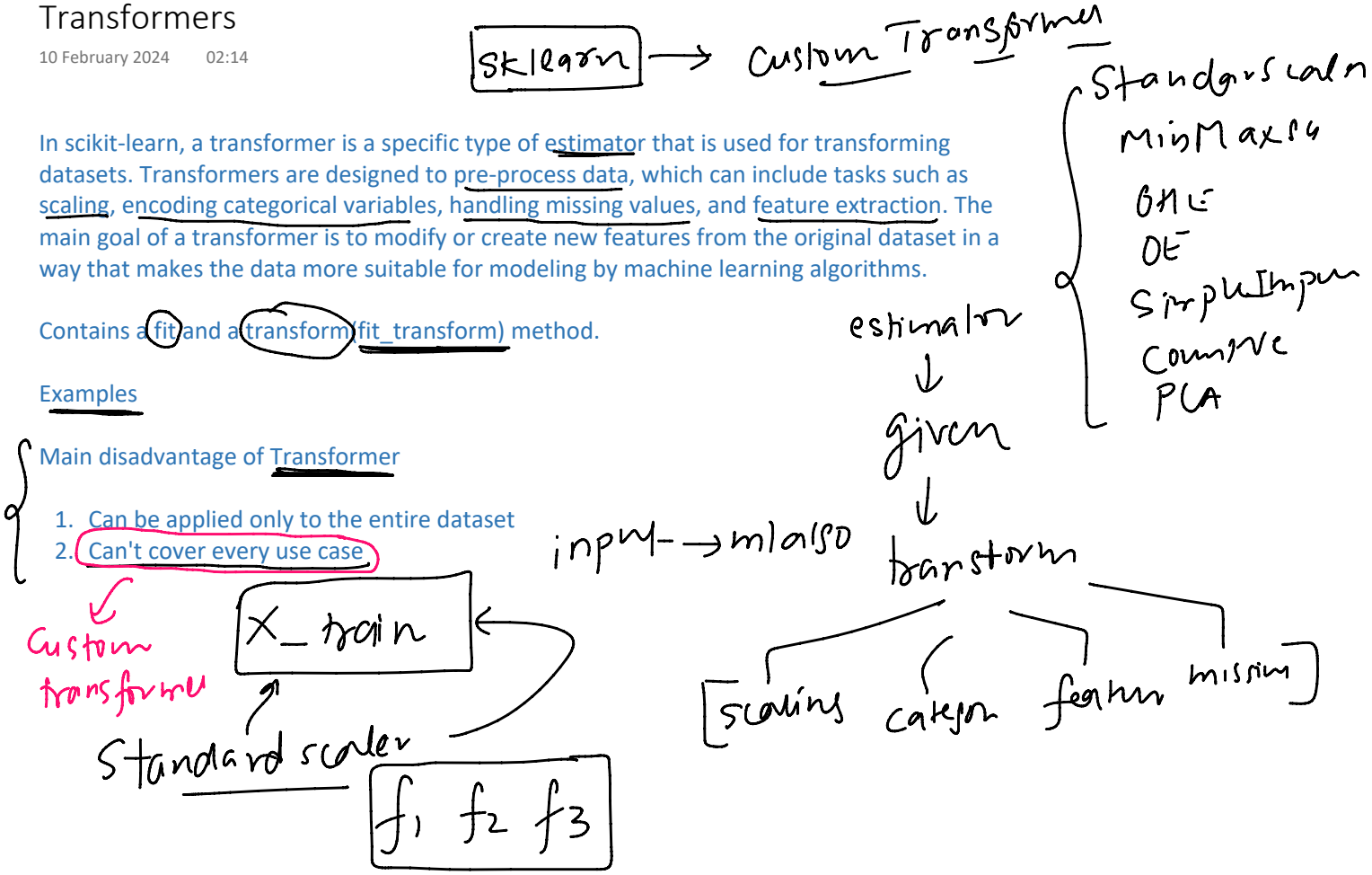
In scikit-learn, a transformer is a specific type of estimator that is used for transforming datasets. Transformers are designed to pre-process data, which can include tasks such as scaling, encoding categorical variables, handling missing values, and feature extraction. The main goal of a transformer is to modify or create new features from the original dataset in a way that makes the data more suitable for modeling by machine learning algorithms.

Contains a fit and a transform (fit_transform) method.

Examples

Main disadvantage of Transformer

1. Can be applied only to the entire dataset
2. Can't cover every use case



Custom Transformer

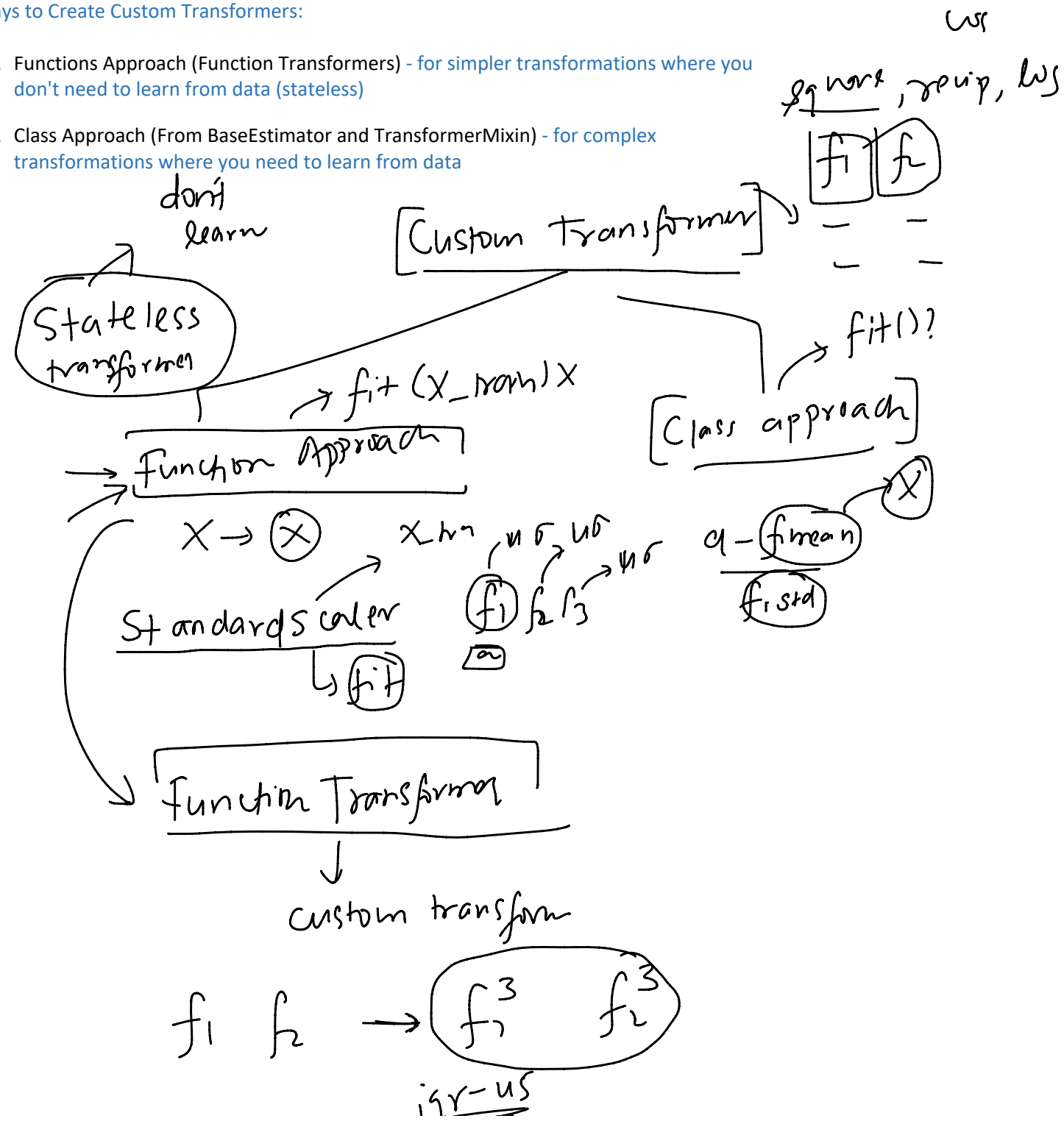
10 February 2024 02:15

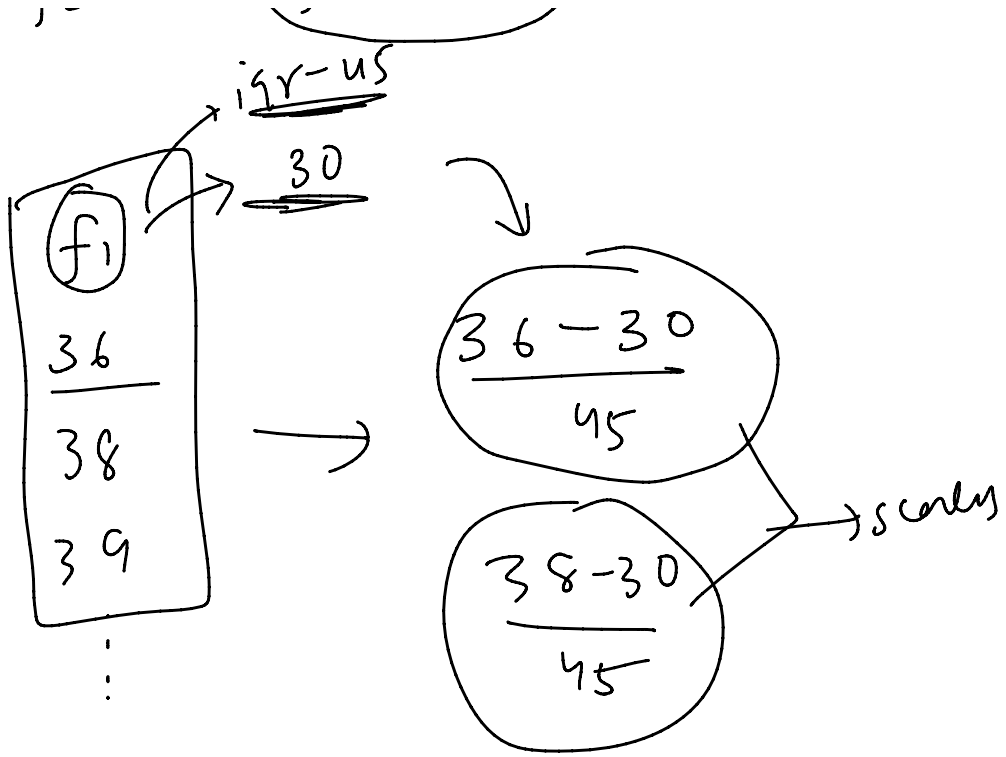
Custom transformers in scikit-learn are user-defined transformers that are designed to carry out specific data transformations or pre-processing steps that are not covered by the built-in transformers in scikit-learn.

Creating custom transformers is a powerful way to incorporate domain-specific knowledge or custom data processing logic into your machine learning pipelines.

Ways to Create Custom Transformers:

1. Functions Approach (Function Transformers) - for simpler transformations where you don't need to learn from data (stateless)
2. Class Approach (From BaseEstimator and TransformerMixin) - for complex transformations where you need to learn from data





estimator

→ custom estimator

transformer

→ custom transformer

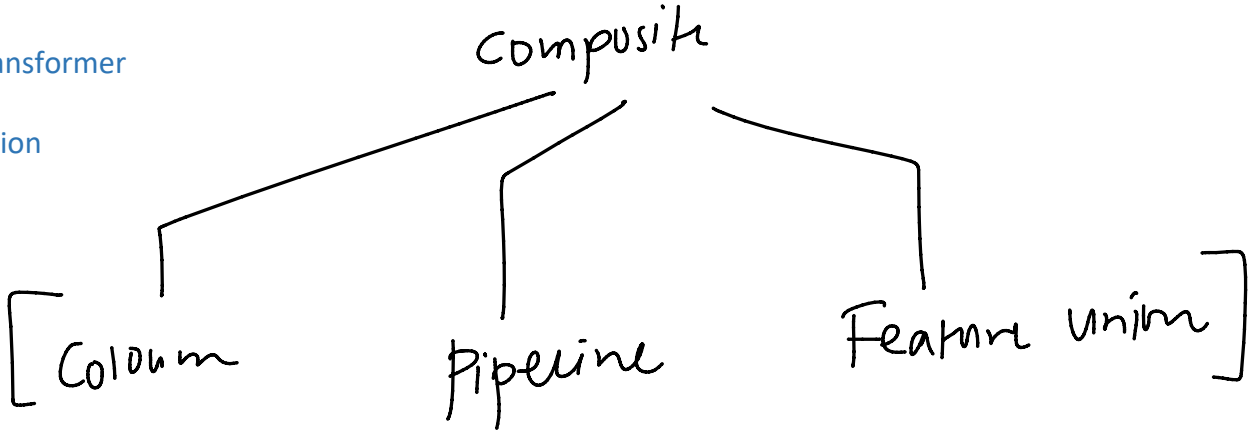
Composite Transformers

10 February 2024 03:16

A composite transformer refers to a transformer that is built from multiple other transformers or estimators, combining their functionalities to apply a series of transformations or processing steps in a specific way.

Types

1. Column Transformer
2. Pipeline
3. Feature Union



Column Transformer

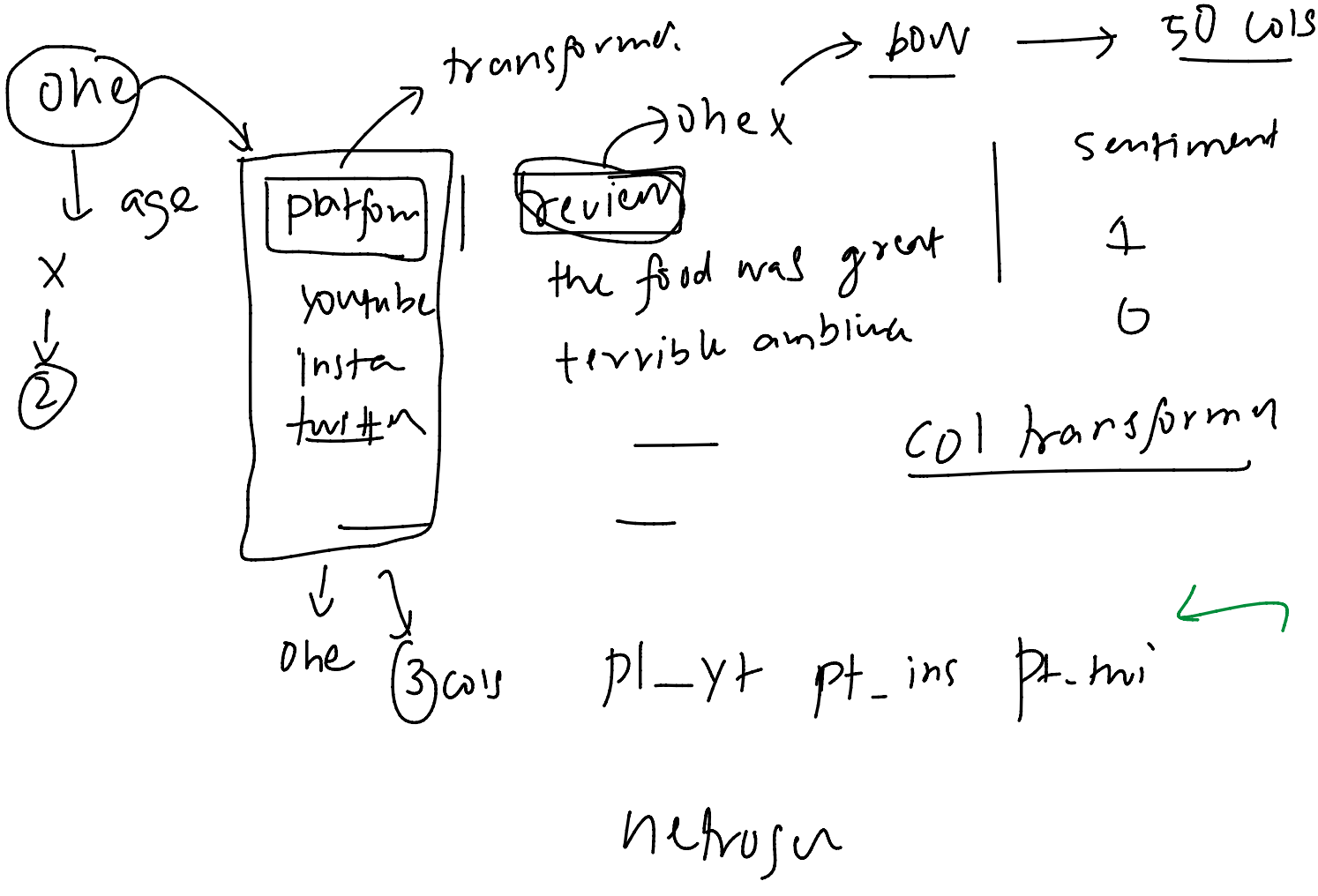
10 February 2024 02:15

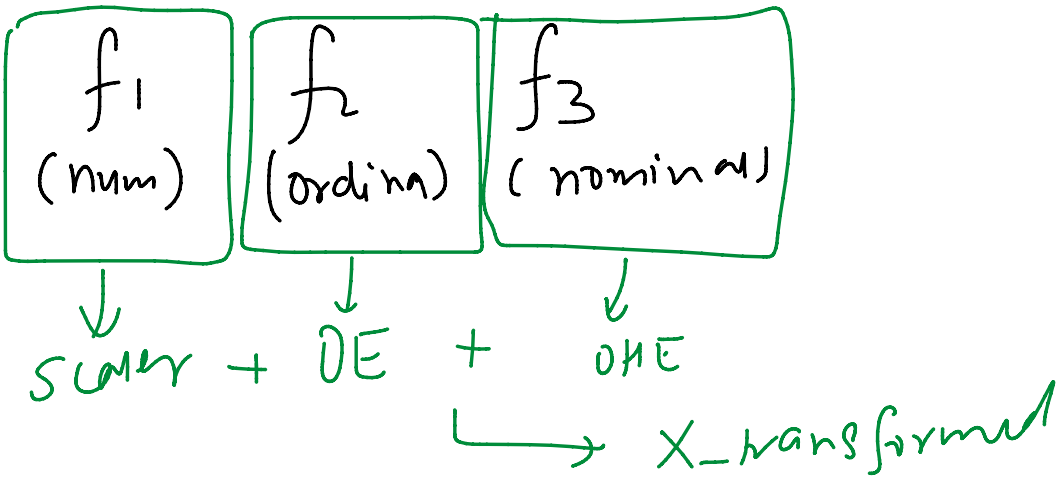
The ColumnTransformer is a feature transformer from scikit-learn that allows different columns or column subsets of the input dataset to be transformed separately and the features generated by each transformer to be concatenated into a single feature space. This is particularly useful for datasets that contain various types of data requiring different preprocessing steps, such as numerical data that needs scaling and categorical data that needs to be encoded.

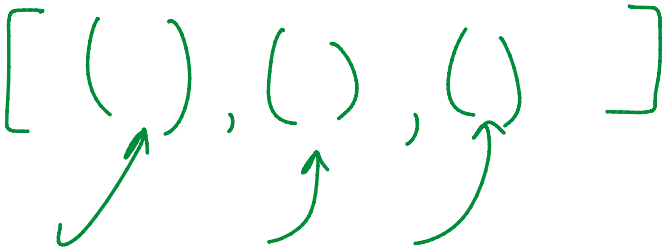
Code Example

What is allowed in Column Transformer?

Transformers, Function Transformers, Custom Transformers, Pipelines, Feature Unions



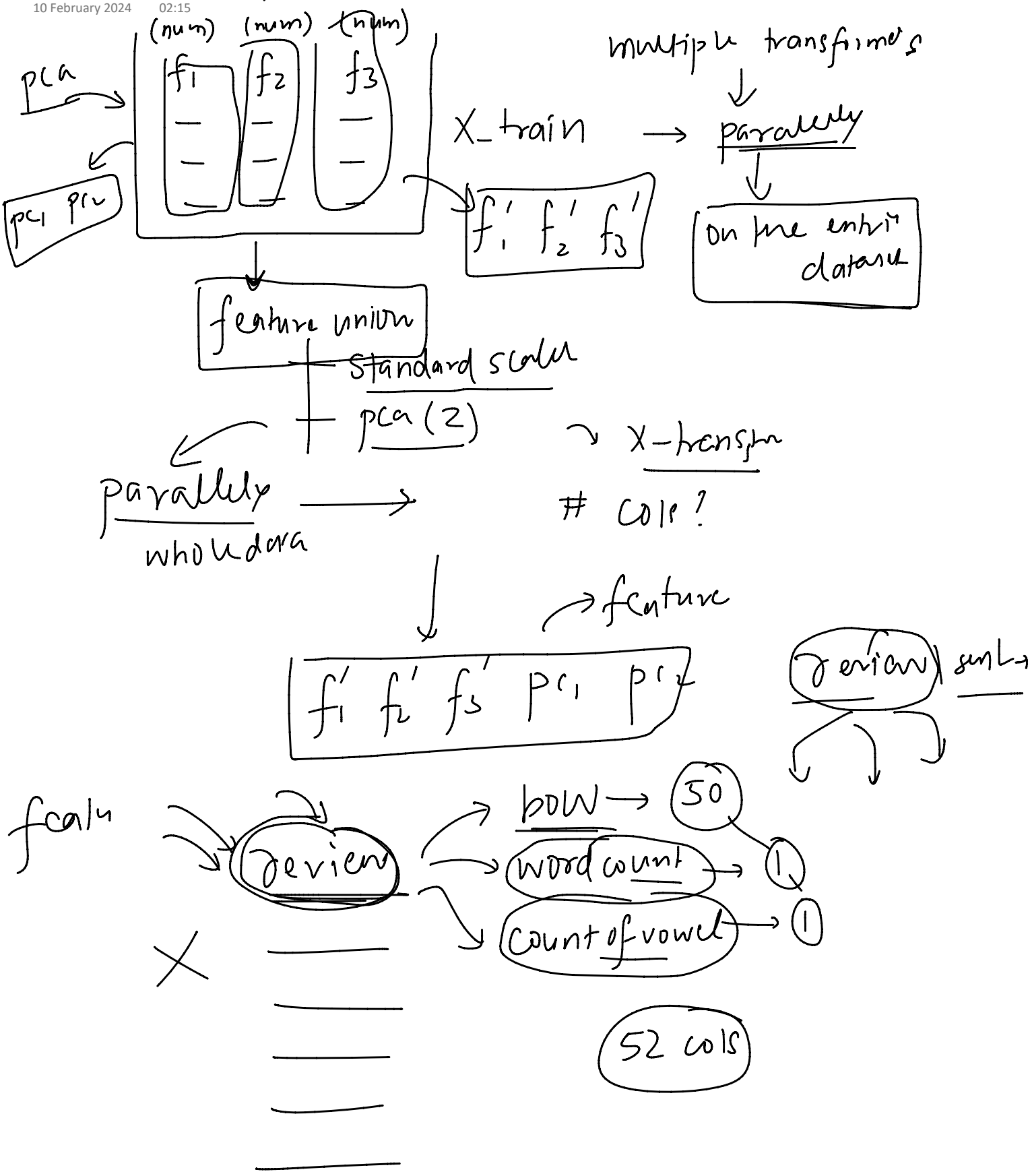


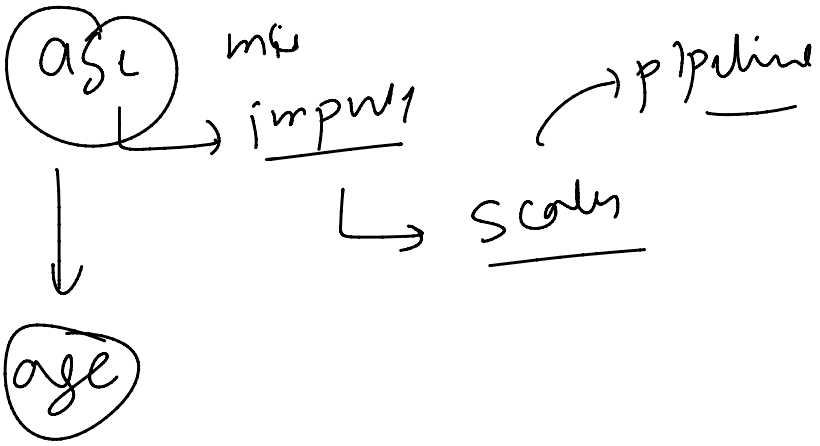


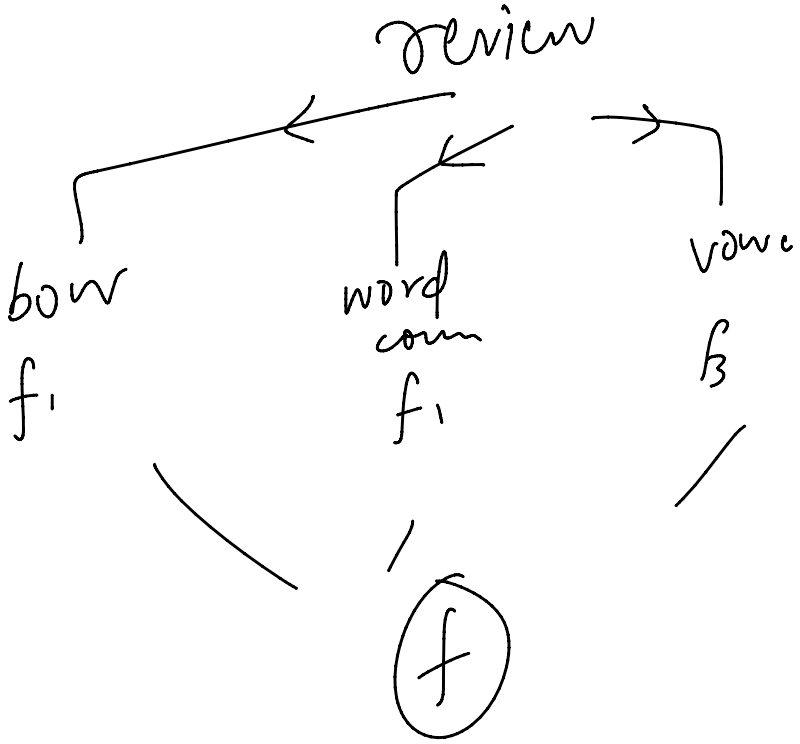
Salman, OHE(), ——

Feature Union ⟷ pipeline

10 February 2024 02:15

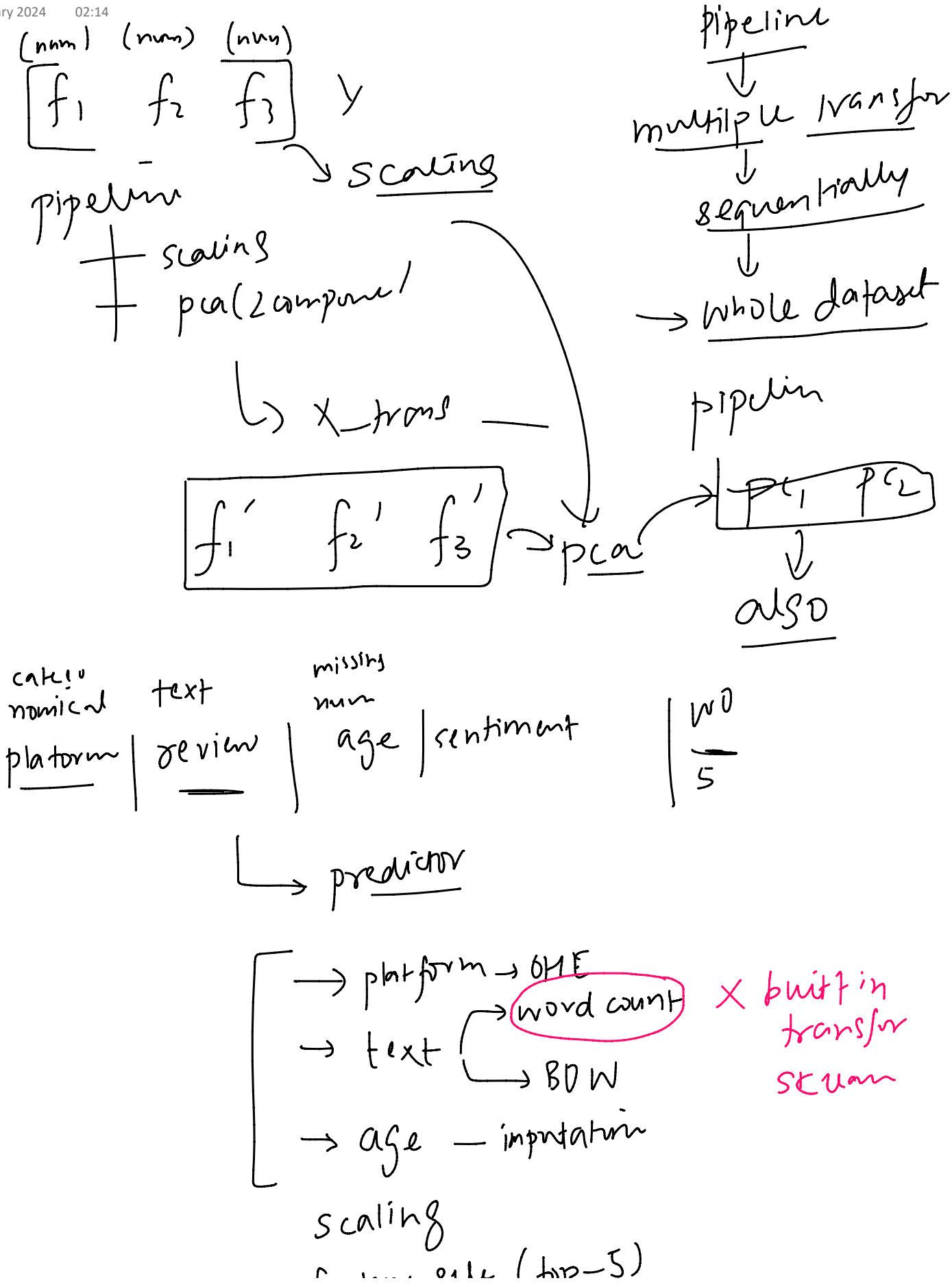



asl
mg
impwl
pipeline
scala
age


review
bow
f1
word
com
f1
vow
f3
f

Pipeline

10 February 2024 02:14



scaling

feature selection (top-5)

↳ logistic regression