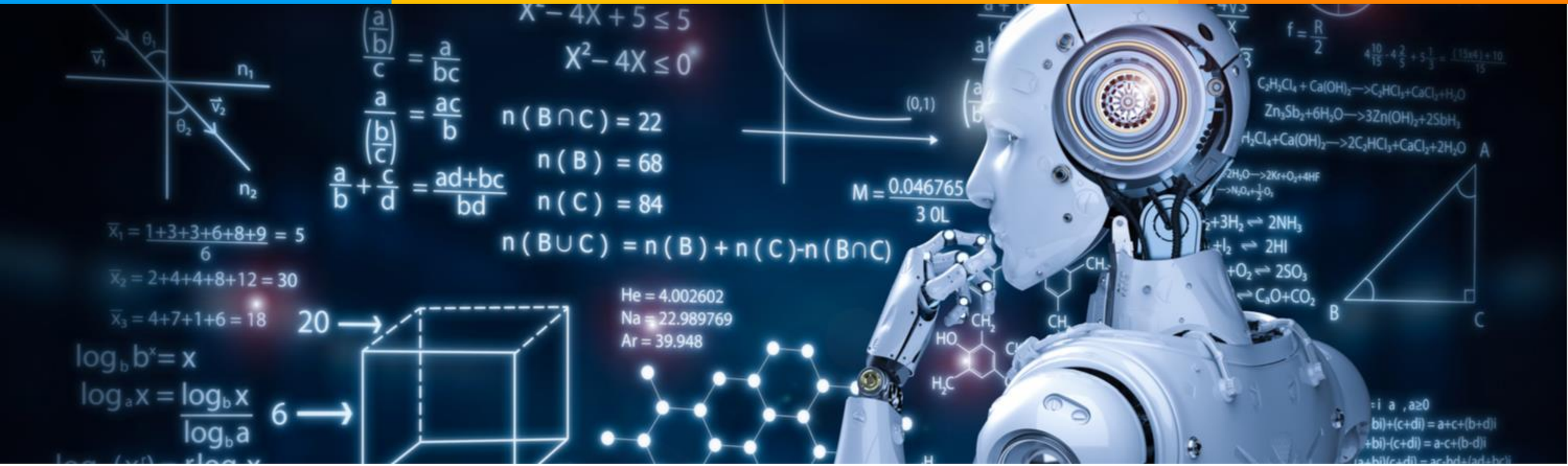
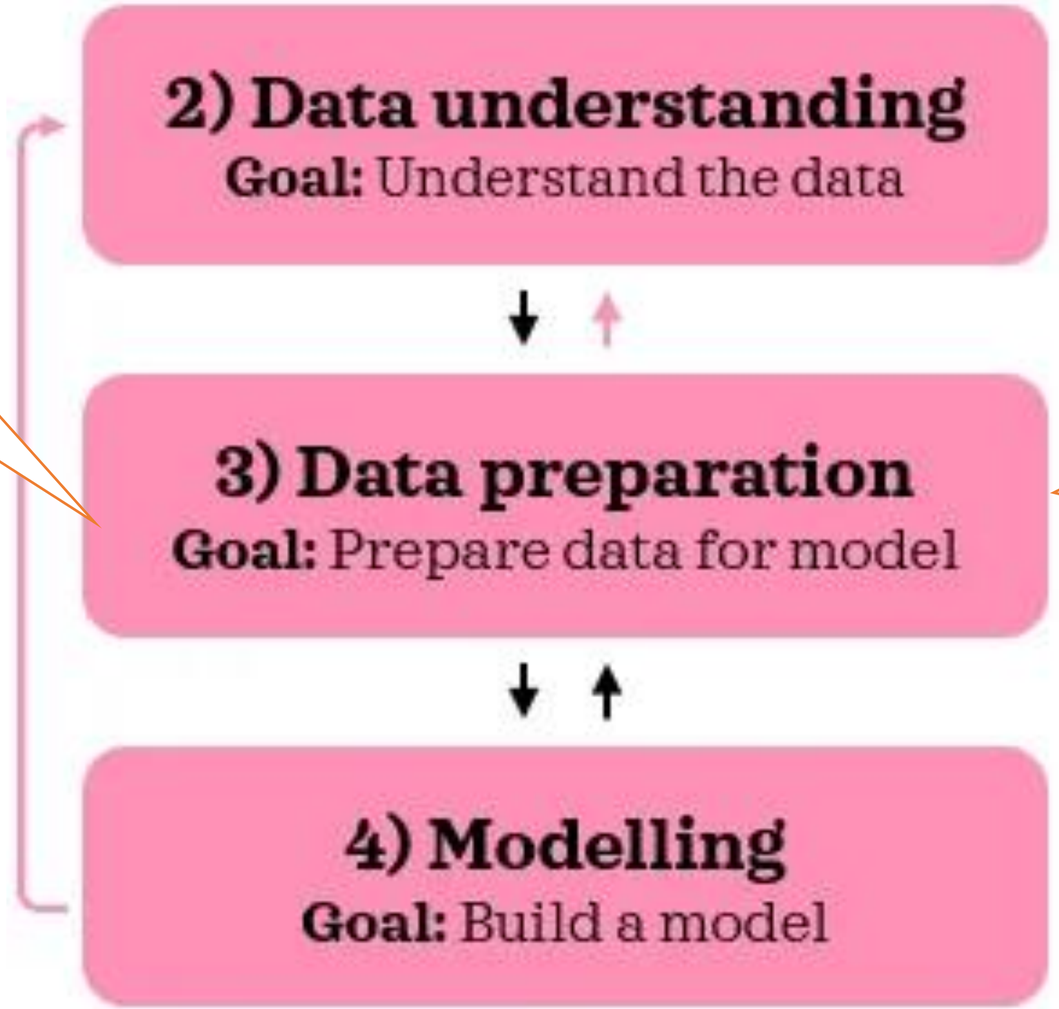


Lecture 14: Scikit Learn Pipelines

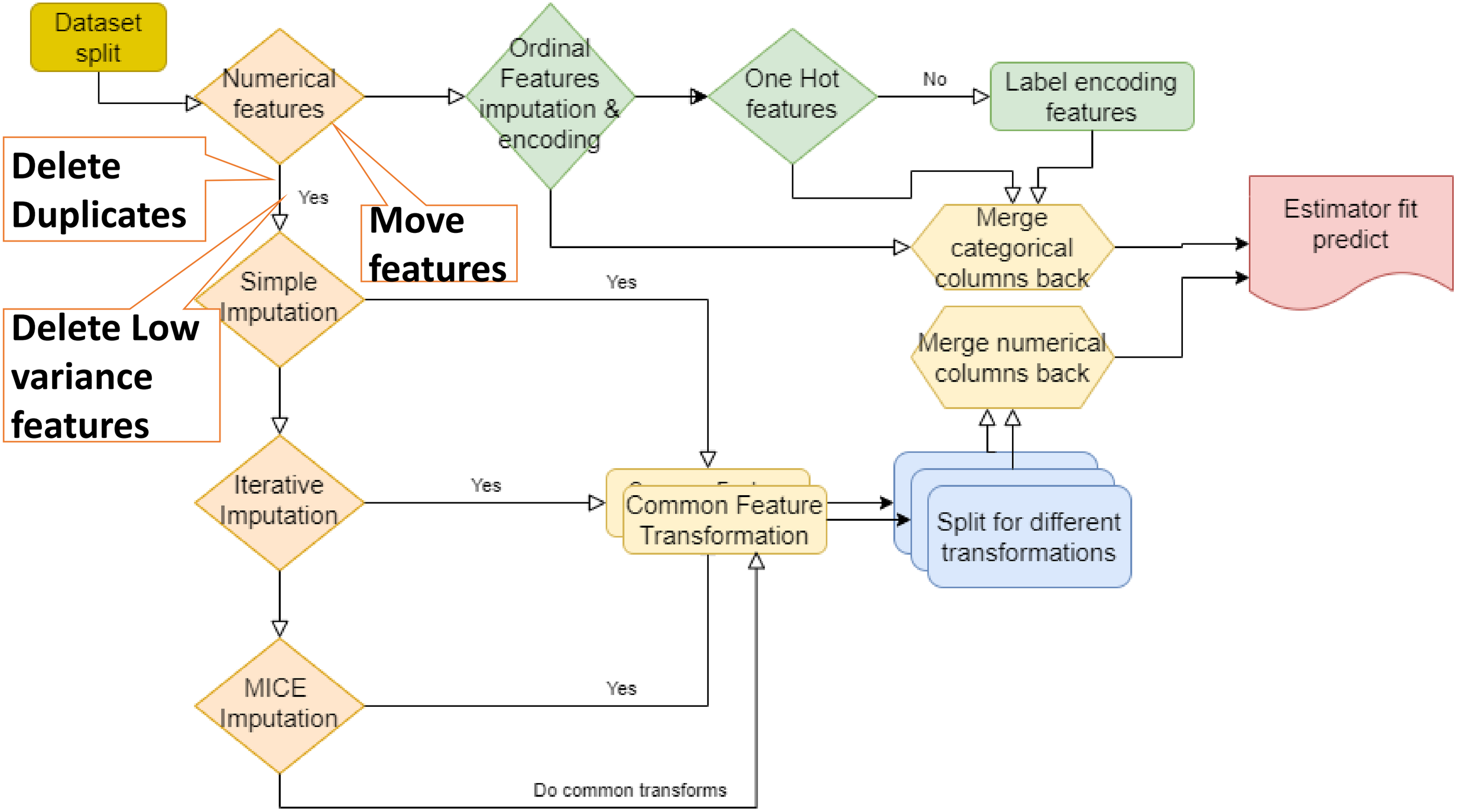
Recap

- Kmeans algorithm
- Expectation Maximization
- Centroid initialization with kmeans++
- Kmeans decision boundary
- Kmeans limitations with oblong data clusters
- Kmeans limitations with outliers
- Using Kmeans and silhouette analysis for outlier detection

Data Preprocessing (duplicate removal, imputation, encoding)



1. Feature Selection
2. Feature Transformation/ Engineering
3. Feature Extraction


Dataset split
Numerical features
Ordinal Features imputation & encoding
One Hot features
No
Label encoding features
Delete Duplicates
Yes
Move features
Merge categorical columns back
Estimator fit predict
Delete Low variance features
Simple Imputation
Yes
Merge numerical columns back
Iterative Imputation
Yes
Common Feature Transformation
Split for different transformations
MICE Imputation
Yes
Do common transforms

Without pipeline

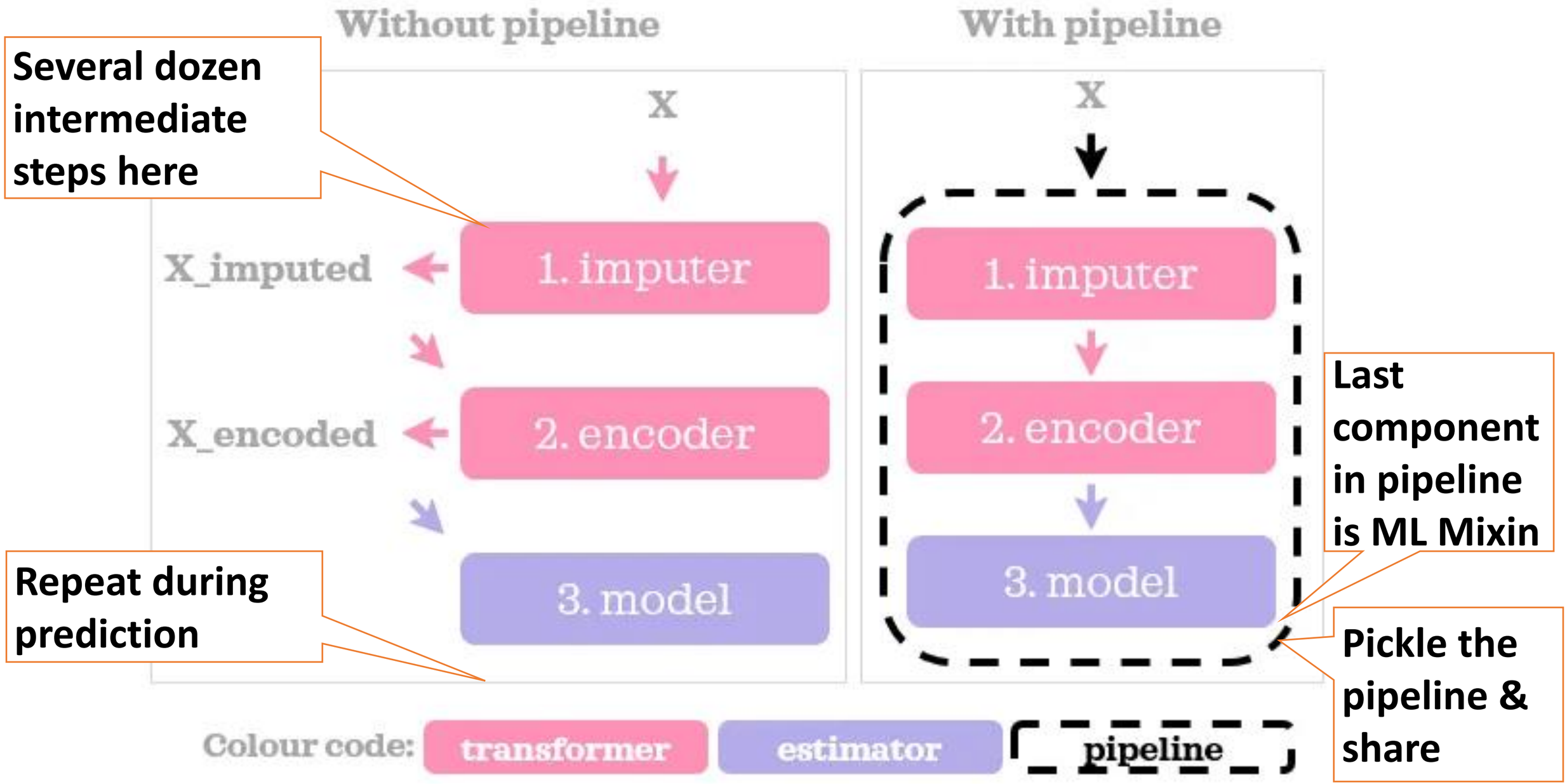
```
#imputer object for missing value
imputer = SimpleImputer(strategy="median")

# scales object the numerical feature
scaler = StandardScaler()

# one-hot object for the categorical features
one_hot=OneHotEncoder(handle_unknown='ignore',sparse=False)

# Applying the fit and transform methods to the Training data
imputer.fit(X_train[numeric_features])
imputed=imputer.transform(X_train[numeric_features])

scaler.fit(imputed)
scaled=scaler.transform(imputed)

one_hot.fit(X_train[categorical_features])
one_hotted_cat=one_hot.transform(X_train[categorical_features])

# Concatenating the scaled and one hot matrixes
Final_train=pd.DataFrame(np.concatenate((scaled,one_hotted_cat), axis=1))
lr.fit(Final_train, y_train)
```

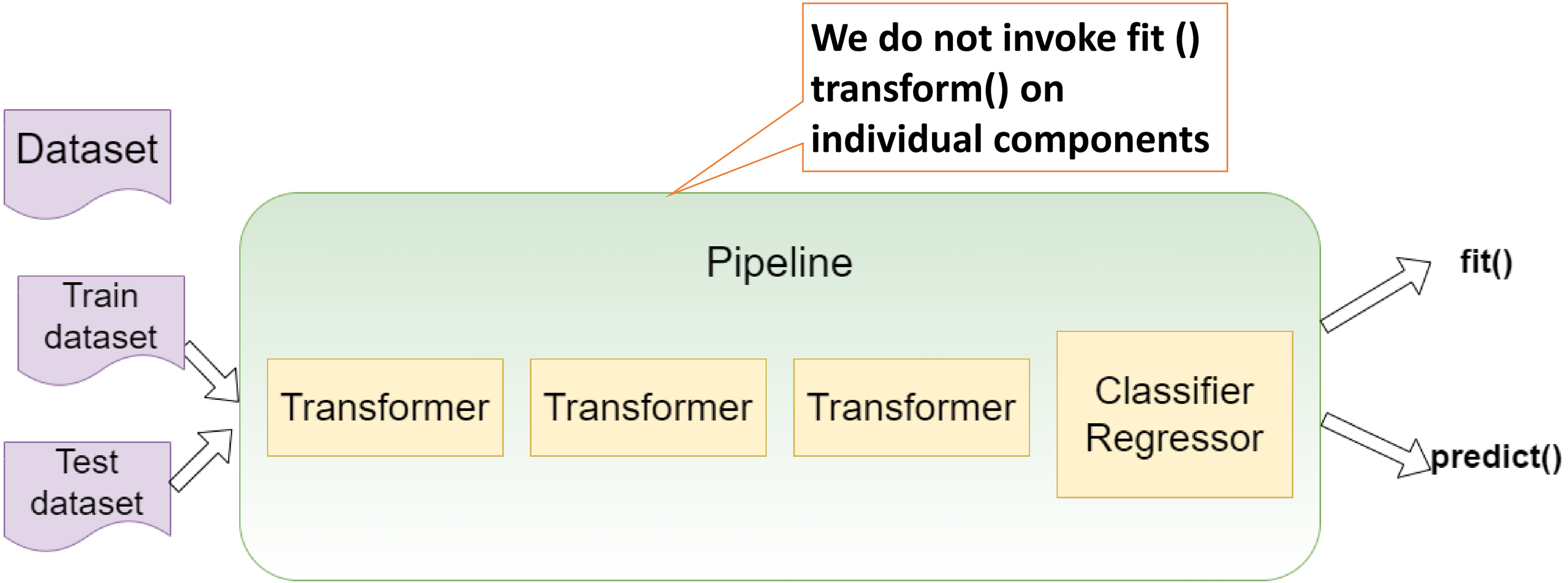
- Preprocessing generally 500-1000 features
  - To be grouped in about 25-50 features
  - E.g. Diff numeric fields need diff imputation
  - Diff groups need diff encoding
  - Encoding as embedding
  - Feature Selection, Elimination, Importance, extraction etc.
  - Distribution transformation
  - Cross validation
- Output of one should be fed into next
- Dynamic code, rearranged often
- Error prone by design

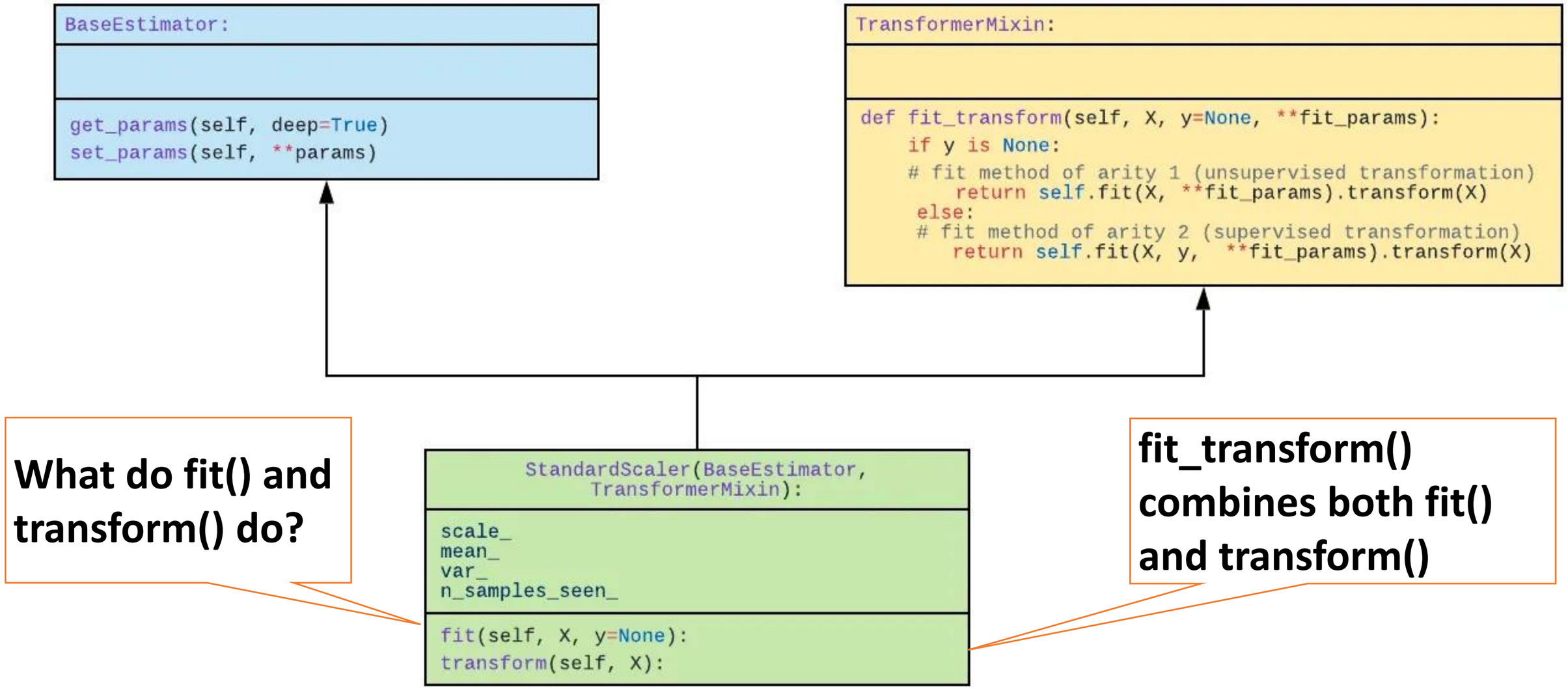
```
# Predict on the test set-using the trained classifier-still need to do the transformations
X_test_filled = imputer.transform(X_test[numeric_features])
X_test_scaled = scaler.transform(X_test_filled)
X_test_one_hot = one_hot.transform(X_test[categorical_features])
X_test=pd.DataFrame(np.concatenate((X_test_scaled, X_test_one_hot), axis=1))
lr.score(X_test,y_test)
```

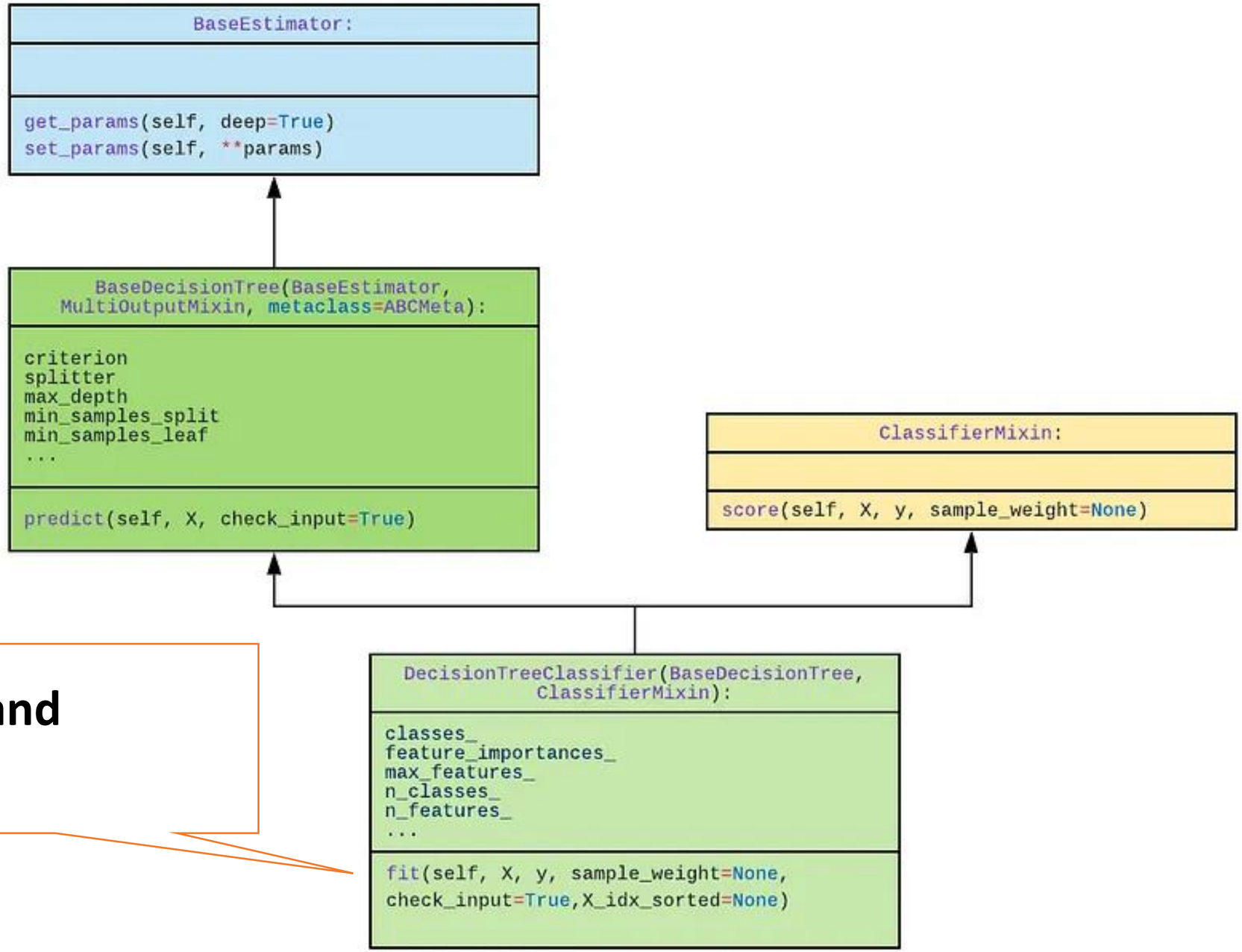
- Same order in predict
- Predict done by some other team
  - Two dozen models
  - Provide all configured imputers
  - High cost of ownership & maintenance
  - Brittle code - Touch and it breaks
  - Logistics nightmare
- Data leakage (What is it?)

With pipeline


Without pipeline
With pipeline
X
1. imputer
X_imputed
2. encoder
X_encoded
3. model
Several dozen intermediate steps here
Repeat during prediction
Last component in pipeline is ML Mixin
Pickle the pipeline & share
Colour code:
transformer
estimator
pipeline


We do not invoke fit () transform() on individual components
Dataset
Train dataset
Test dataset
Pipeline
Transformer
Transformer
Transformer
Classifier Regressor
fit()
predict()


BaseEstimator:
get_params(self, deep=True)
set_params(self, **params)
TransformerMixin:
def fit_transform(self, X, y=None, **fit_params):
if y is None:
# fit method of arity 1 (unsupervised transformation)
return self.fit(X, **fit_params).transform(X)
else:
# fit method of arity 2 (supervised transformation)
return self.fit(X, y, **fit_params).transform(X)
What do fit() and transform() do?
StandardScaler(BaseEstimator, TransformerMixin):
scale_
mean_
var_
n_samples_seen_
fit(self, X, y=None):
transform(self, X):
fit_transform() combines both fit() and transform()



Only fit() and predict()

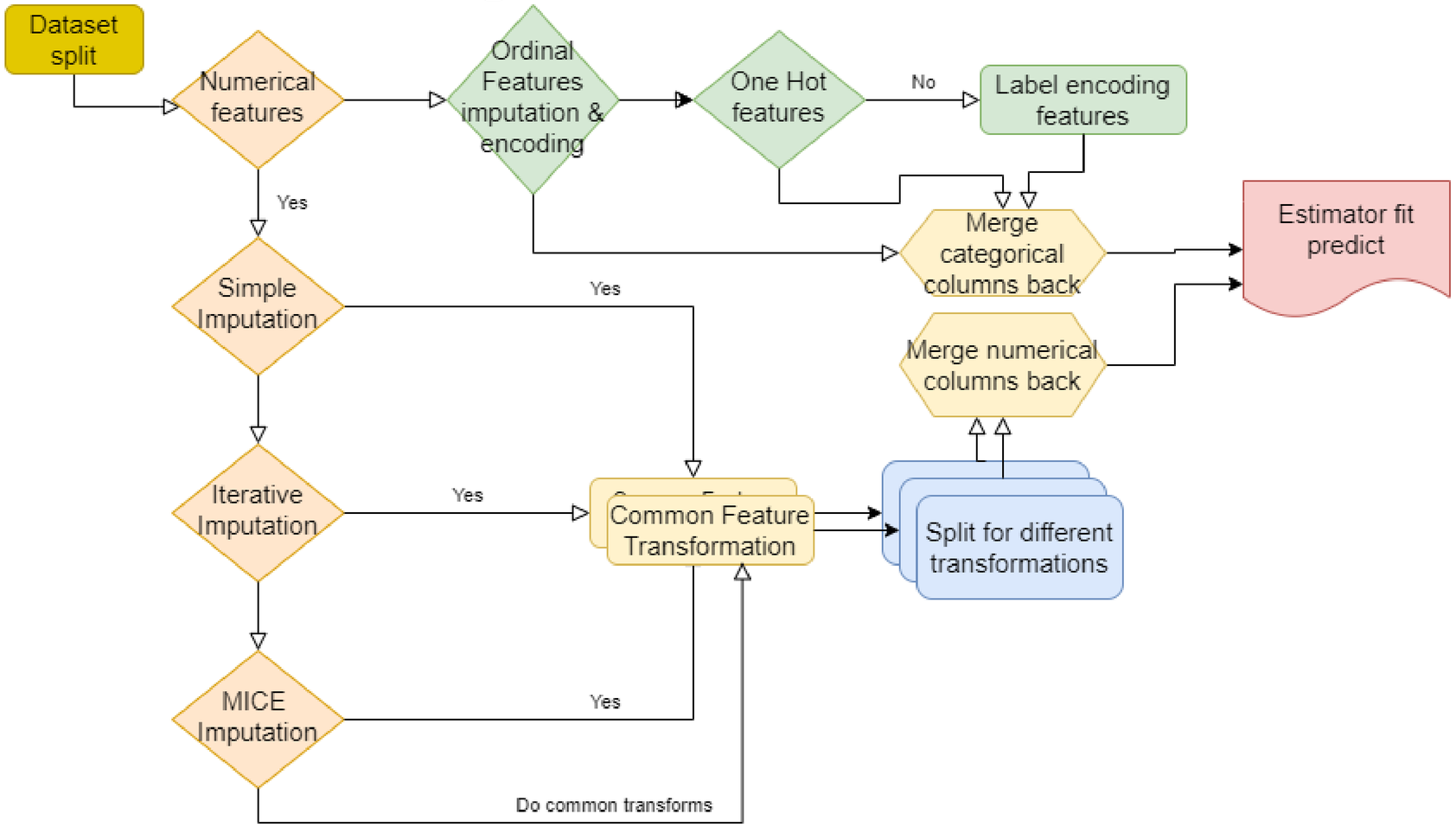
fit(), fit_transform() and predict()

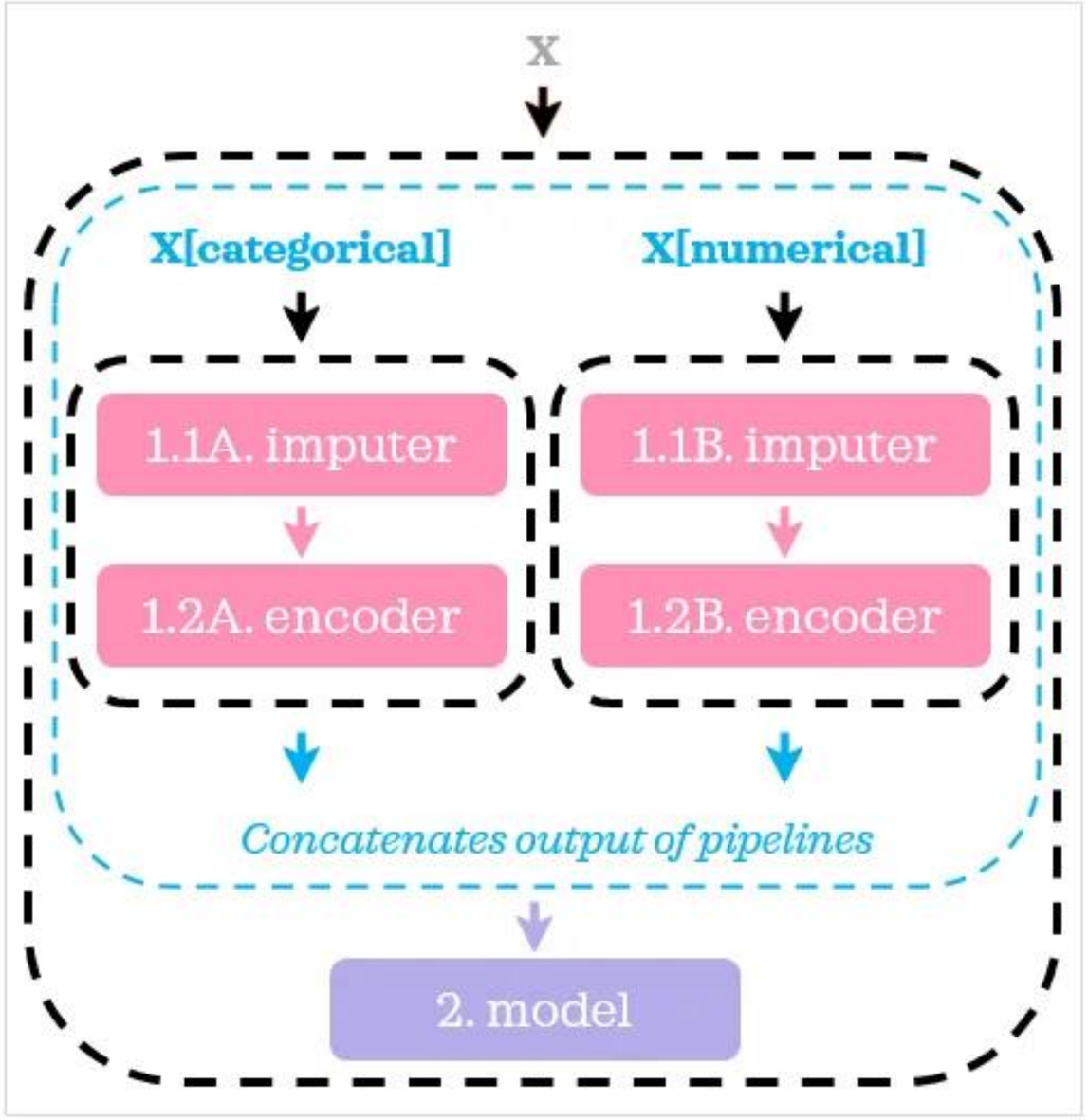
- Always configure pipeline with
  - Last component as Classifier/Regressor/Cluster Mixin
  - Other components are Transformers
- Invoke fit() on pipeline during training
  - Sklearn invokes fit_transform() on all but last component
  - Sklearn invokes fit() on last component
- Invoke predict() on pipeline during test
  - Sklearn invokes transform() on all but last component
  - Sklearn invokes predict on last component
- Sklearn relies on python duck typing than class hierarchy

Parallel pipelines

Mixture Model clustering

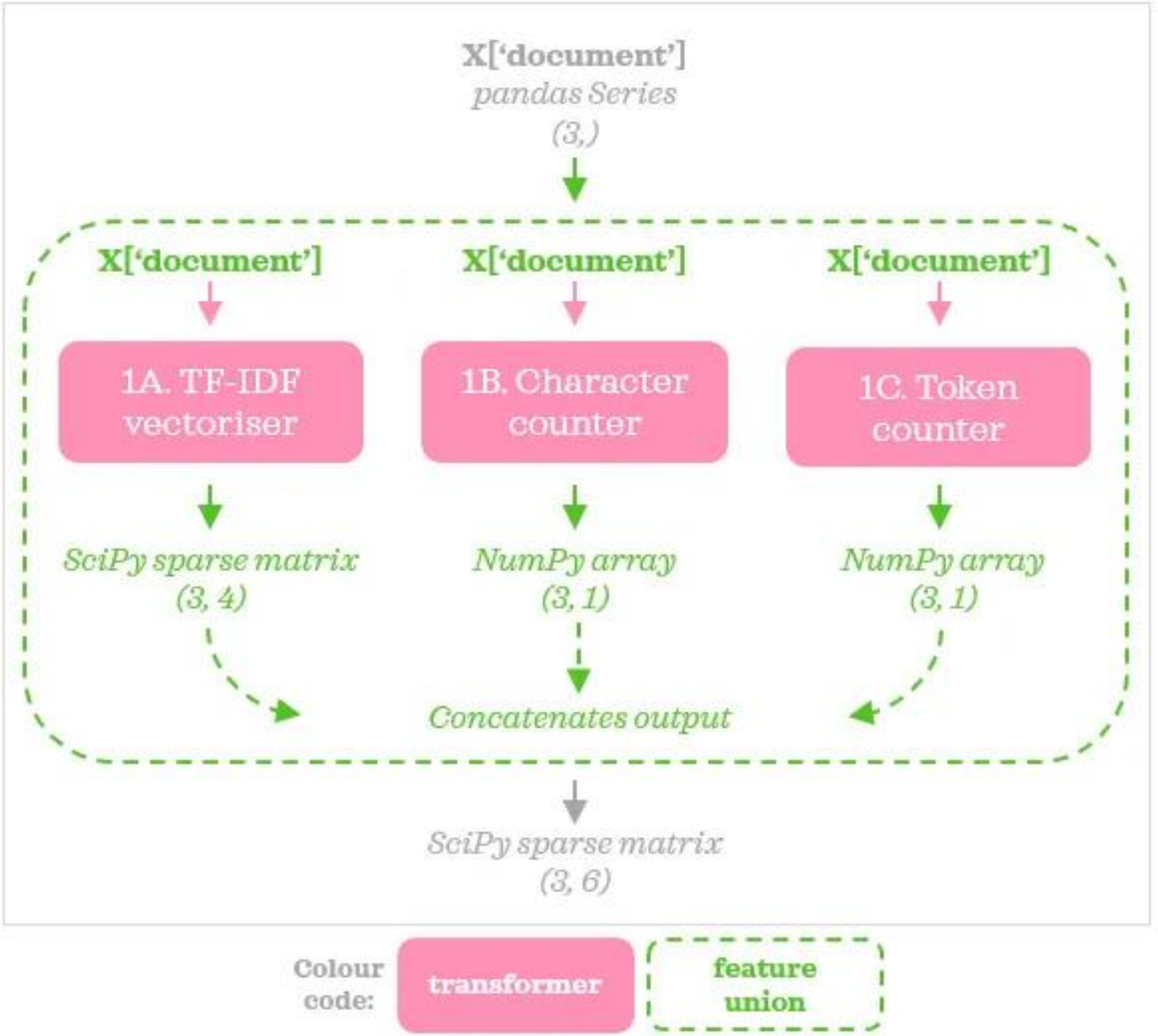
- Apply different transformations on numerical & categorical features. Then merge back

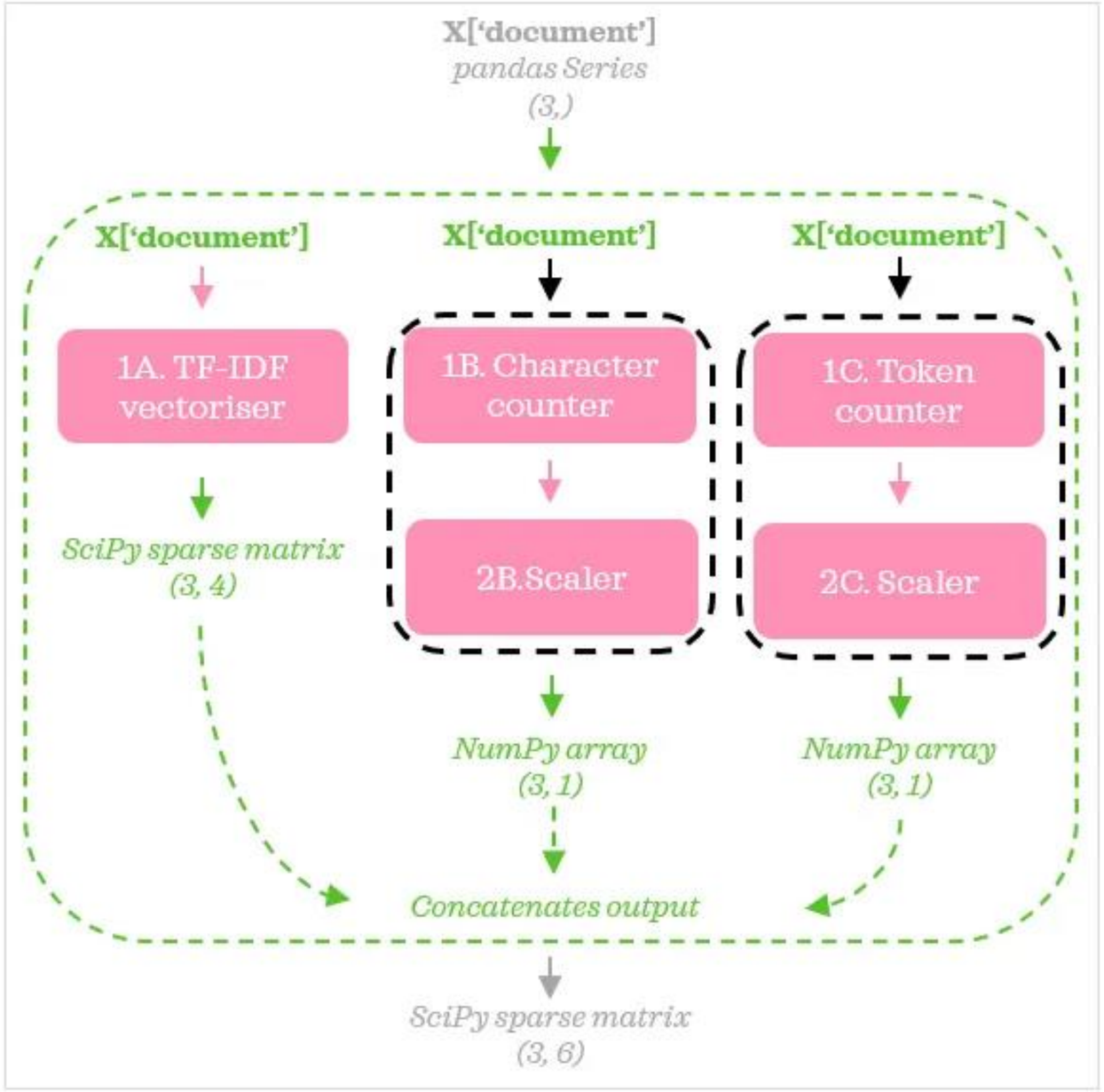


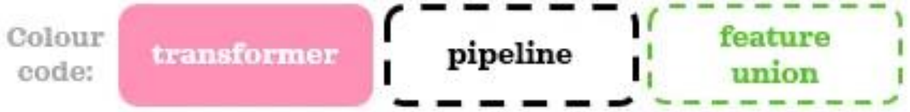

X
X[categorical]
X[numerical]
1.1A. imputer
1.1B. imputer
1.2A. encoder
1.2B. encoder
Concatenates output of pipelines
2. model


Colour code:
transformer
estimator
pipeline
column transformer

```
from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer

numeric_transformer = Pipeline(steps=[
                                      ('meanimputer',     SimpleImputer(strategy='mean')),
                                      ('stdscaler', StandardScaler())
                                      ])

categorical_transformer = Pipeline(steps=[
                                          ('onehotenc', OneHotEncoder(handle_unknown='ignore'))
                                          ])

col_transformer = ColumnTransformer(transformers=[('numeric_processing',numeric_transformer,
                                                   numeric_features),
                                                  ('categorical_processing',categorical_transformer,
                                                   categorical_features)



pipeline = Pipeline([
                     ('transform_column', col_transformer),
                     ('logistics', LogisticRegression())
                    ])


pipeline.fit(X_train, y_train)
pipeline.score(X_test, y_test)
```


X['document']
pandas Series
(3,)
X['document']
1A. TF-IDF vectoriser
SciPy sparse matrix
(3, 4)
X['document']
1B. Character counter
NumPy array
(3, 1)
X['document']
1C. Token counter
NumPy array
(3, 1)
Concatenates output
SciPy sparse matrix
(3, 6)
Colour code:
transformer
feature union


X['document']
pandas Series
(3,)
X['document']
X['document']
X['document']
1A. TF-IDF vectoriser
1B. Character counter
1C. Token counter
SciPy sparse matrix
(3, 4)
2B.Scaler
2C. Scaler
NumPy array
(3, 1)
NumPy array
(3, 1)
Concatenates output
SciPy sparse matrix
(3, 6)


Colour code:
transformer
pipeline
feature union

GridSearch with Pipeline

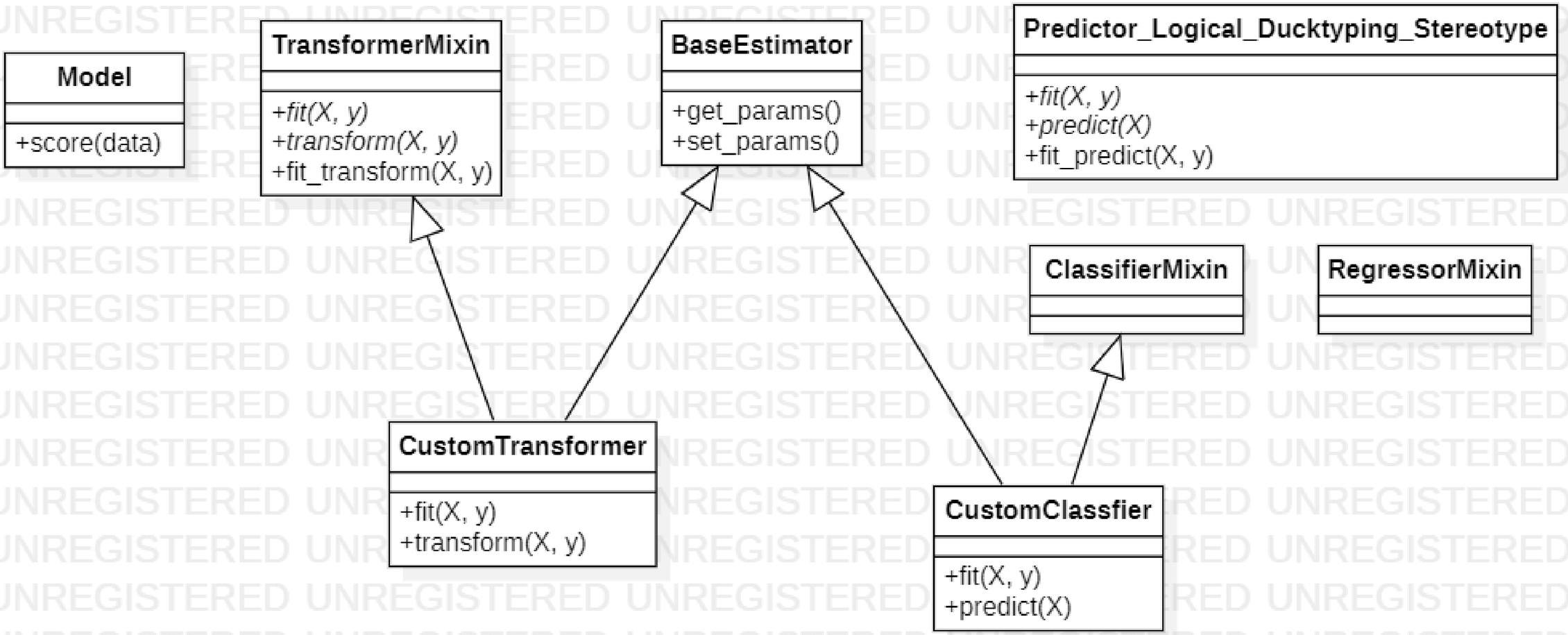
```
from sklearn.model_selection import GridSearchCV

X_train, X_test, y_train, y_test = train_test_split(df, y, random_state=42)

#create new a knn model
knn = KNeighborsClassifier()
#create a dictionary of all values we want to test for n_neighbors
param_grid = {"n_neighbors": np.arange(1, 25),
              "weights": ["uniform", "distance"]}

loocv = LeaveOneOut()

#use gridsearch to test all values for n_neighbors
knn_gscv = GridSearchCV(knn, param_grid, cv=loocv)
#fit model to data
knn_gscv.fit(X_train, y_train)
```

Replace with pipeline

Odds and ends

- PowerTransformer
- FunctionTransformer
- BYOT – Bring Your Own Transformer
  - Custom Transformer, Custom Predictor

Custom Transformer & Classifier/Regressor




QUESTIONS