

Lecture 6: Column transformer and text features

Firas Moosvi (Slides adapted from Varada Kolhatkar)

Announcements

- Lecture recordings for the first two weeks have been made available - See Piazza.
- My Office Hours
- HW3 is due next week Tuesday, Oct 1st, 11:59 pm.
 - You can work in pairs for this assignment.

Quick Correction on Exercise 5.3

I accidentally said *only D is true*, but B is also true!

Select all of the following statements which are TRUE.

- a. You can have scaling of numeric features, one-hot encoding of categorical features, and scikit-learn estimator within a single pipeline.
- b. Once you have a scikit-learn pipeline object with an estimator as the last step, you can call fit, predict, and score on it.
- c. You can carry out data splitting within scikit-learn pipeline.
- d. We have to be careful of the order we put each transformation and model in a pipeline.

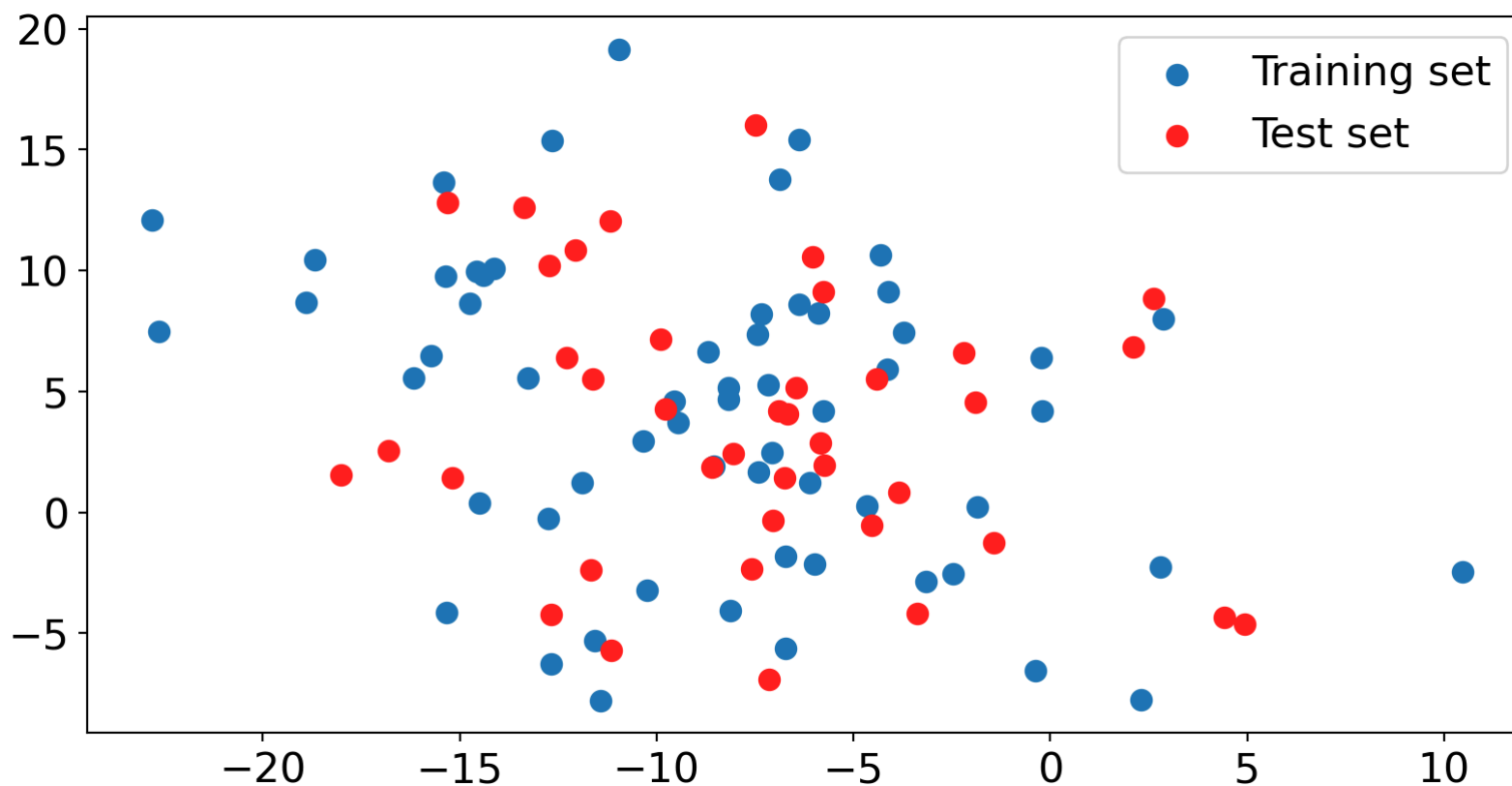
Recap: Preprocessing mistakes

Data

```

1 X, y = make_blobs(n_samples=100, centers=3, random_state=12, cluster_std=5) # make synthetic data
2 X_train_toy, X_test_toy, y_train_toy, y_test_toy = train_test_split(
3     X, y, random_state=5, test_size=0.4) # split it into training and test sets
4 # Visualize the training data
5 plt.scatter(X_train_toy[:, 0], X_train_toy[:, 1], label="Training set", s=60)
6 plt.scatter(
7     X_test_toy[:, 0], X_test_toy[:, 1], color=mplotlib.cm2(1), label="Test set", s=60
8 )
9 plt.legend(loc="upper right")

```



Bad ML 1

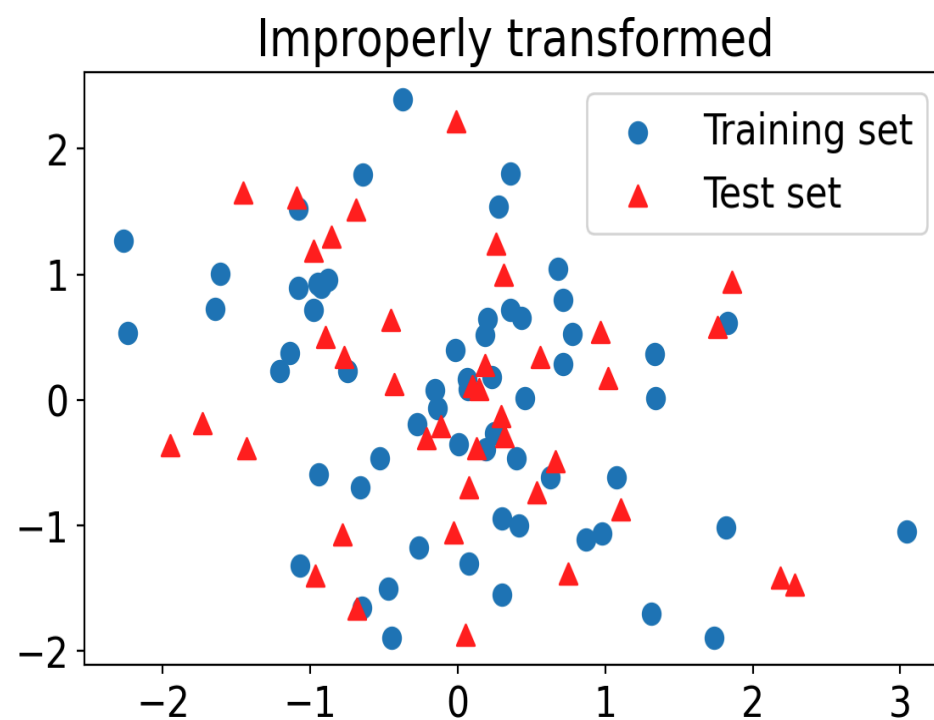
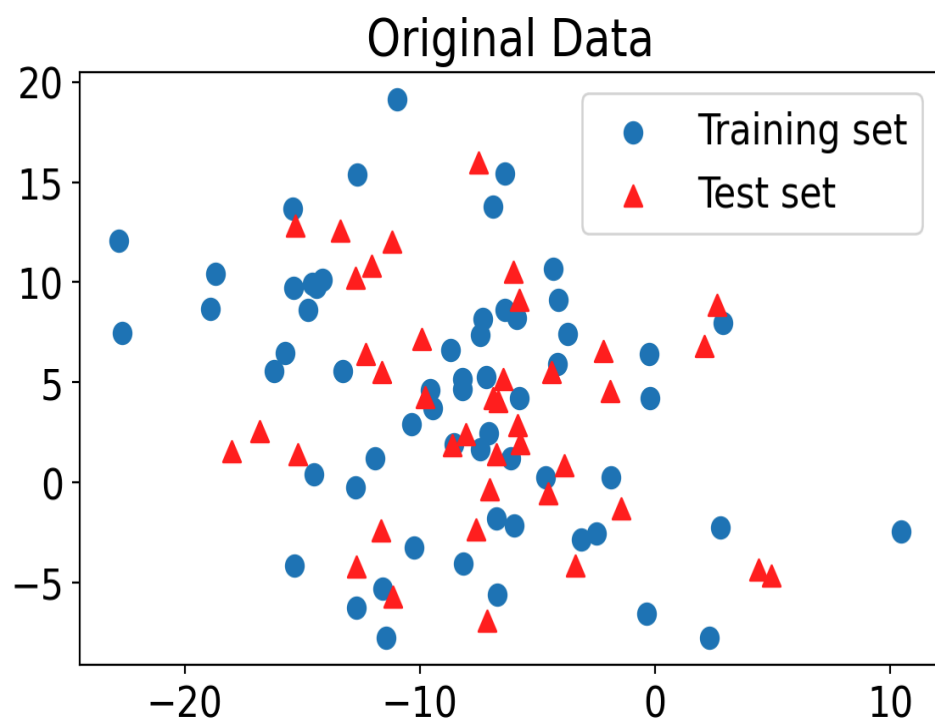
- What's wrong with the approach below?

```
1 scaler = StandardScaler() # Creating a scaler object
2 scaler.fit(X_train_toy) # Calling fit on the training data
3 train_scaled = scaler.transform(
4     X_train_toy
5 ) # Transforming the training data using the scaler fit on training data
6
7 scaler = StandardScaler() # Creating a separate object for scaling test data
8 scaler.fit(X_test_toy) # Calling fit on the test data
9 test_scaled = scaler.transform(
10     X_test_toy
11 ) # Transforming the test data using the scaler fit on test data
12
13 knn = KNeighborsClassifier()
14 knn.fit(train_scaled, y_train_toy)
15 print(f"Training score: {knn.score(train_scaled, y_train_toy):.2f}")
16 print(f"Test score: {knn.score(test_scaled, y_test_toy):.2f}") # misleading scores
```

Training score: 0.63

Test score: 0.60

Scaling train and test data separately



Bad ML 2

- What's wrong with the approach below?

```
1 # join the train and test sets back together
2 XX = np.vstack((X_train_toy, X_test_toy))
3
4 scaler = StandardScaler()
5 scaler.fit(XX)
6 XX_scaled = scaler.transform(XX)
7
8 XX_train = XX_scaled[:X_train_toy.shape[0]]
9 XX_test = XX_scaled[X_train_toy.shape[0]:]
10
11 knn = KNeighborsClassifier()
12 knn.fit(XX_train, y_train_toy)
13 print(f"Training score: {knn.score(XX_train, y_train_toy):.2f}") # Misleading score
14 print(f"Test score: {knn.score(XX_test, y_test_toy):.2f}") # Misleading score
```

Training score: 0.63

Test score: 0.55

Bad ML 3

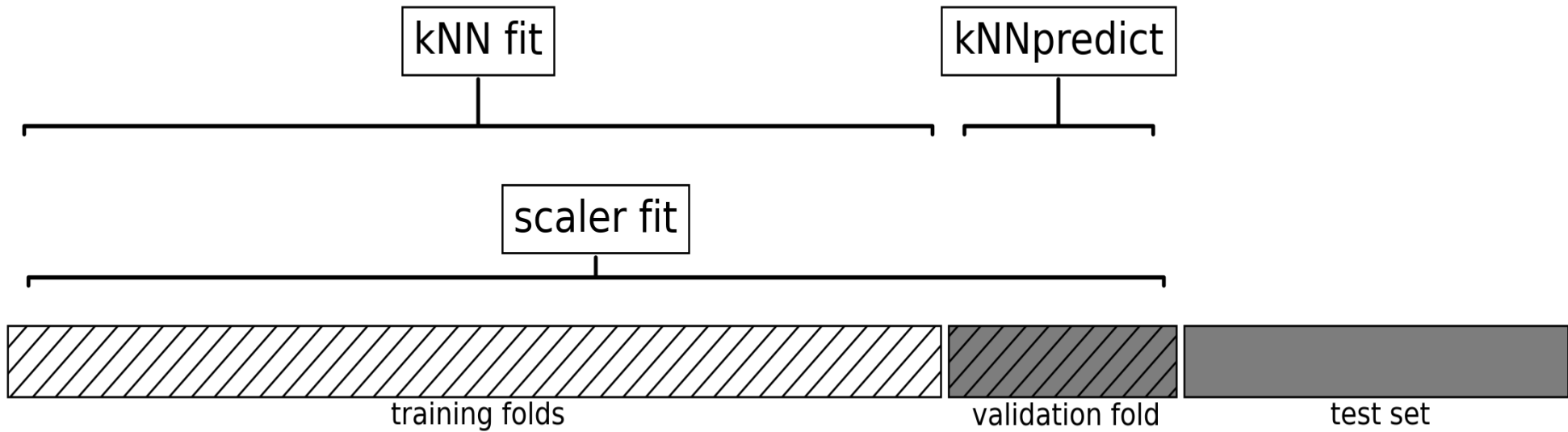
- What's wrong with the approach below?

```
1 knn = KNeighborsClassifier()  
2  
3 scaler = StandardScaler()  
4 scaler.fit(X_train_toy)  
5 X_train_scaled = scaler.transform(X_train_toy)  
6 X_test_scaled = scaler.transform(X_test_toy)  
7 cross_val_score(knn, X_train_scaled, y_train_toy)
```

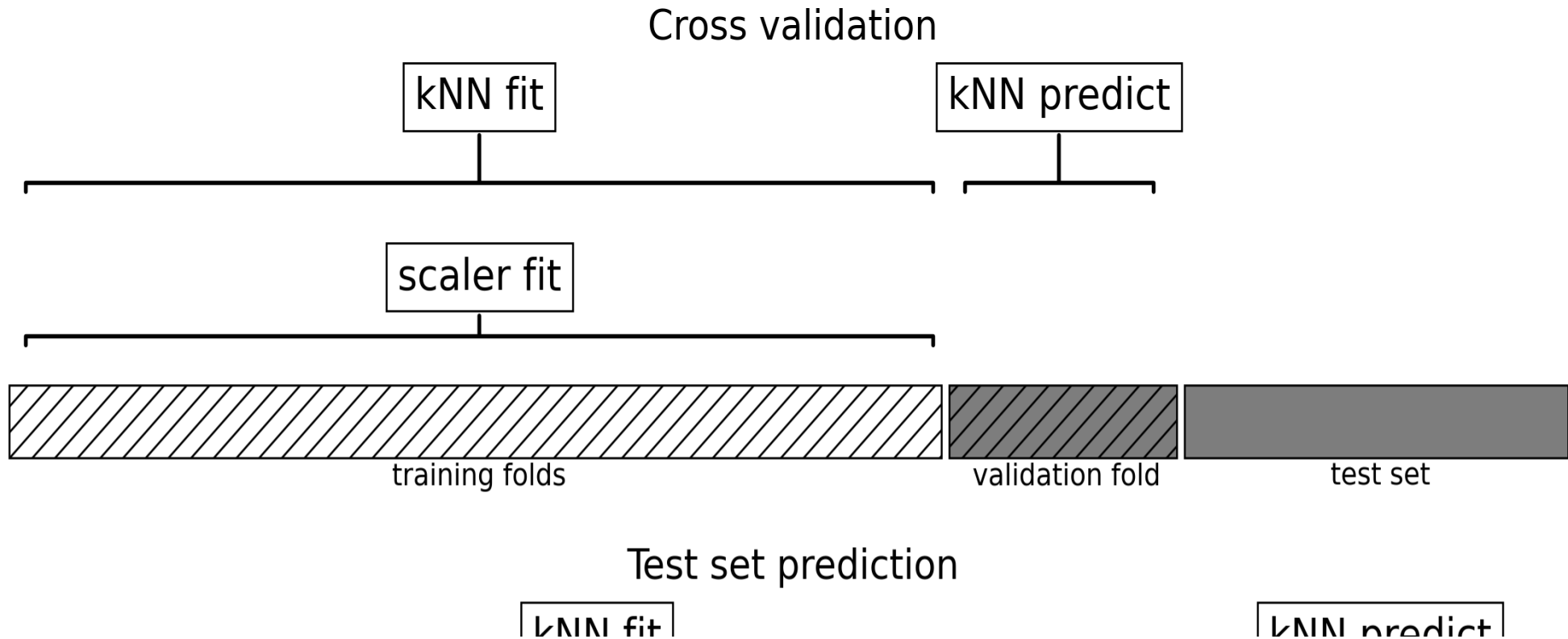
```
array([0.25      , 0.5      , 0.58333333, 0.58333333, 0.41666667])
```

Improper preprocessing

Cross validation



Proper preprocessing



Recap: **sklearn** Pipelines

- Pipeline is a way to chain multiple steps (e.g., preprocessing + model fitting) into a single workflow.
- Simplify the code and improves readability.
- Reduce the risk of data leakage by ensuring proper transformation of the training and test sets.
- Automatically apply transformations in sequence.
- **Example:**
 - Chaining a **StandardScaler** with a **KNeighborsClassifier** model.

```
1 from sklearn.pipeline import make_pipeline
2
3 pipe_knn = make_pipeline(StandardScaler(), KNeighborsClassifier())
4
5 # Correct way to do cross validation without breaking the golden rule.
6 cross_val_score(pipe_knn, X_train_toy, y_train_toy)
```

```
array([0.25      , 0.5      , 0.5      , 0.58333333, 0.41666667])
```

Group Work: Class Demo & Live Coding

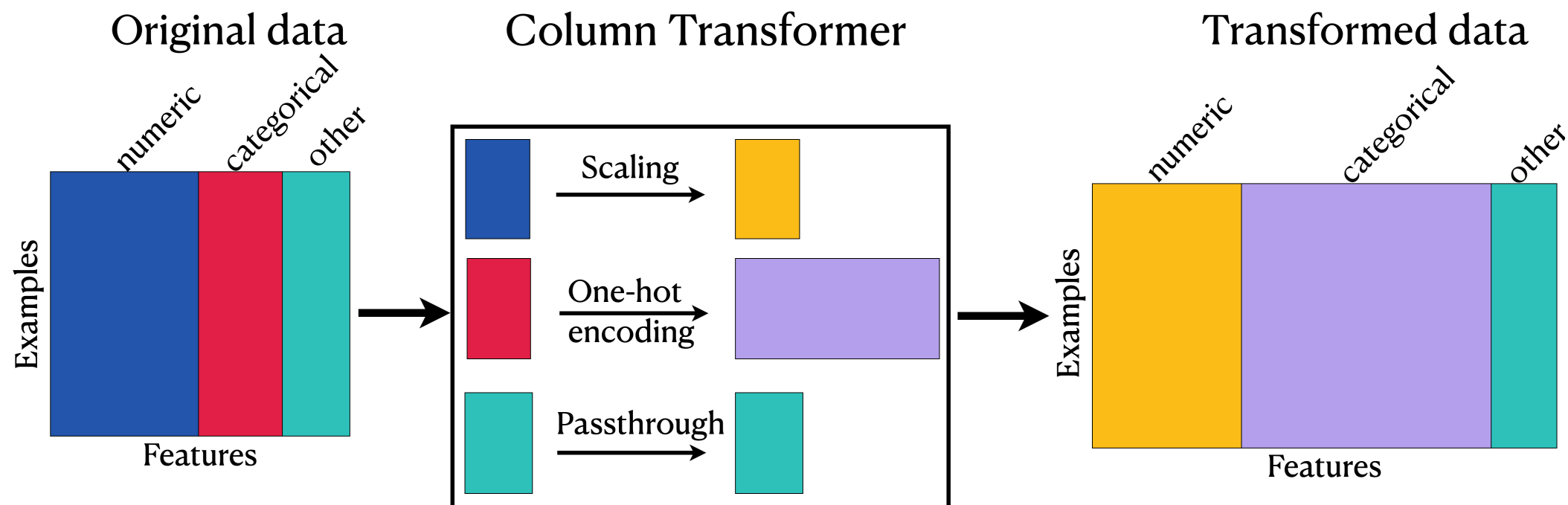
For this demo, each student should [click this link](#) to create a new repo in their accounts, then clone that repo locally to follow along with the demo from today.

If you really don't want to create a repo,

- Navigate to the [cpsc330-2024W1](#) repo
- run `git pull` to pull the latest files in the course repo
- Look for the demo file here: [lectures/102-Firas-lectures/class_demos/](#).

sklearn's ColumnTransformer

- Use ColumnTransformer to build all our transformations together into one object



- Use a column transformer with sklearn pipelines.

(iClicker) Exercise 6.1

iClicker cloud join link: <https://join.iclicker.com/VYFJ>

Select all of the following statements which are TRUE.

- a. You could carry out cross-validation by passing a `ColumnTransformer` object to `cross_validate`.
- b. After applying column transformer, the order of the columns in the transformed data has to be the same as the order of the columns in the original data.
- c. After applying a column transformer, the transformed data is always going to be of different shape than the original data.
- d. When you call `fit_transform` on a `ColumnTransformer` object, you get a numpy ndarray.