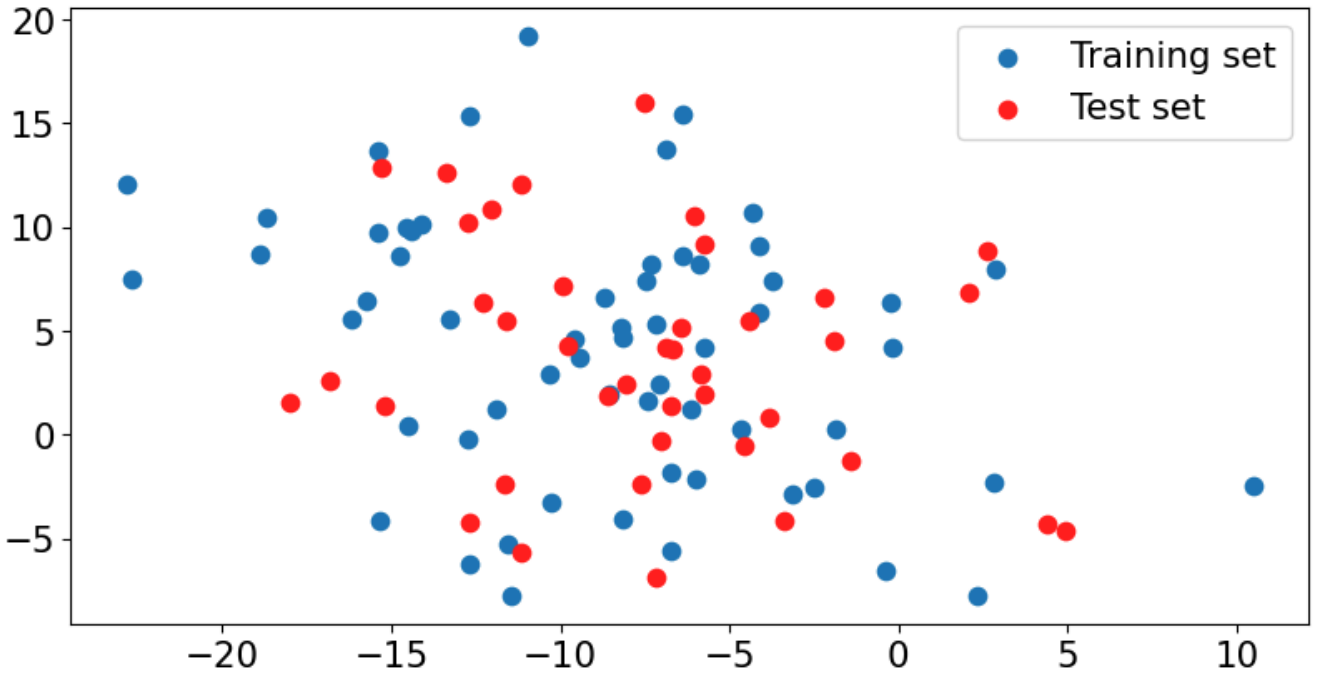
Lecture 6: Column transformer and text features

Firas Moosvi (Slides adapted from Varada Kolhatkar)

Recap: Preprocessing mistakes

Data

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



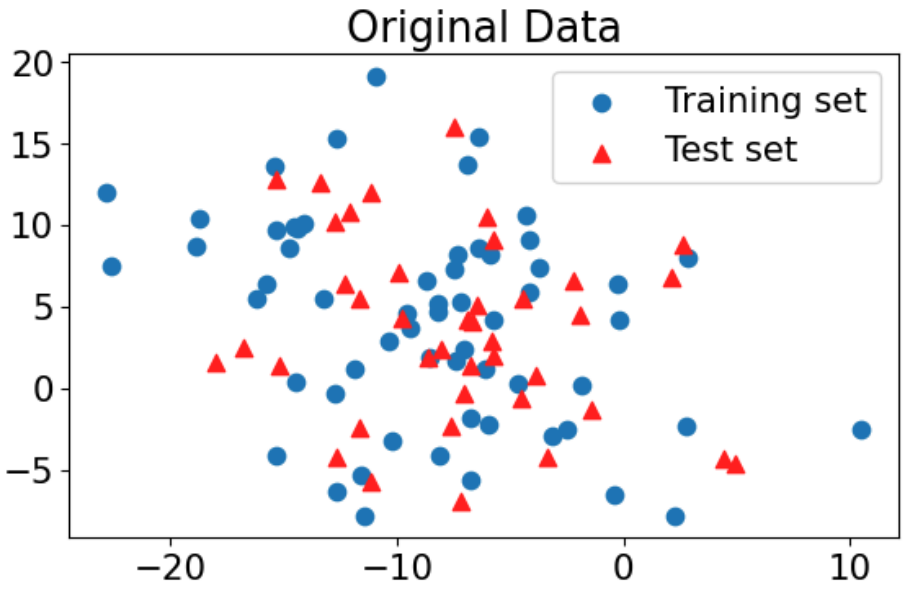
❌ Bad ML 1

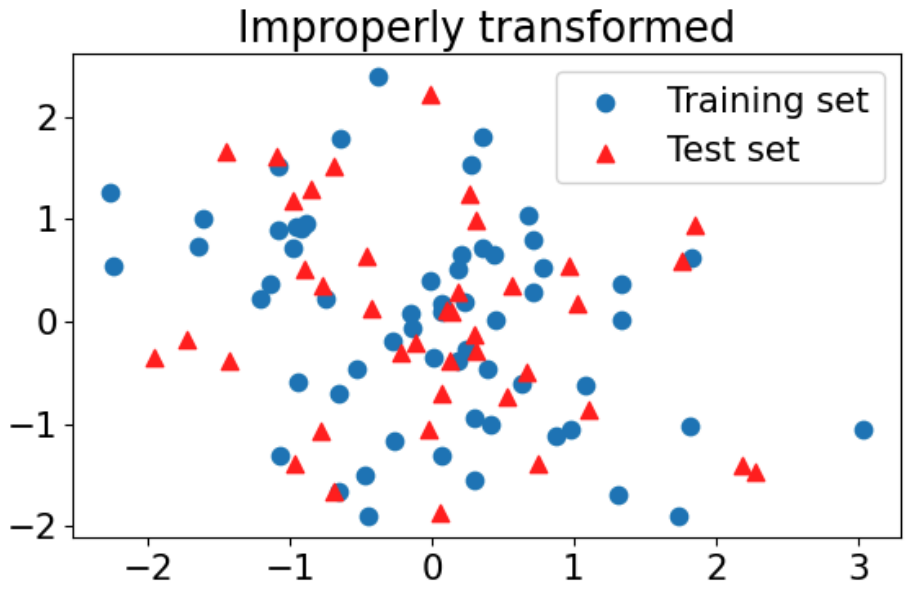
- What's wrong with the approach below?

```
scaler = StandardScaler() # Creating a scalert object
scaler.fit(X_train_toy) # Calling fit on the training data
train_scaled = scaler.transform(
    X_train_toy
)  # Transforming the training data using the scaler fit on training data

scaler = StandardScaler()  # Creating a separate object for scaling test data
scaler.fit(X_test_toy)  # Calling fit on the test data
test_scaled = scaler.transform(
    X_test_toy
)  # Transforming the test data using the scaler fit on test data

knn = KNeighborsClassifier()
knn.fit(train_scaled, y_train_toy)
print(f"Training score: {knn.score(train_scaled, y_train_toy):.2f}")
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?

```
# join the train and test sets back together
XX = np.vstack((X_train_toy, X_test_toy))

scaler = StandardScaler()
scaler.fit(XX)
XX_scaled = scaler.transform(XX)

XX_train = XX_scaled[:X_train_toy.shape[0]]
XX_test = XX_scaled[X_train_toy.shape[0]:]

knn = KNeighborsClassifier()
knn.fit(XX_train, y_train_toy)
print(f"Training score: {knn.score(XX_train, y_train_toy):.2f}")  # Misleading score
print(f"Test score: {knn.score(XX_test, y_test_toy):.2f}")  # Misleading score
```

```
Training score: 0.63
Test score: 0.55
```

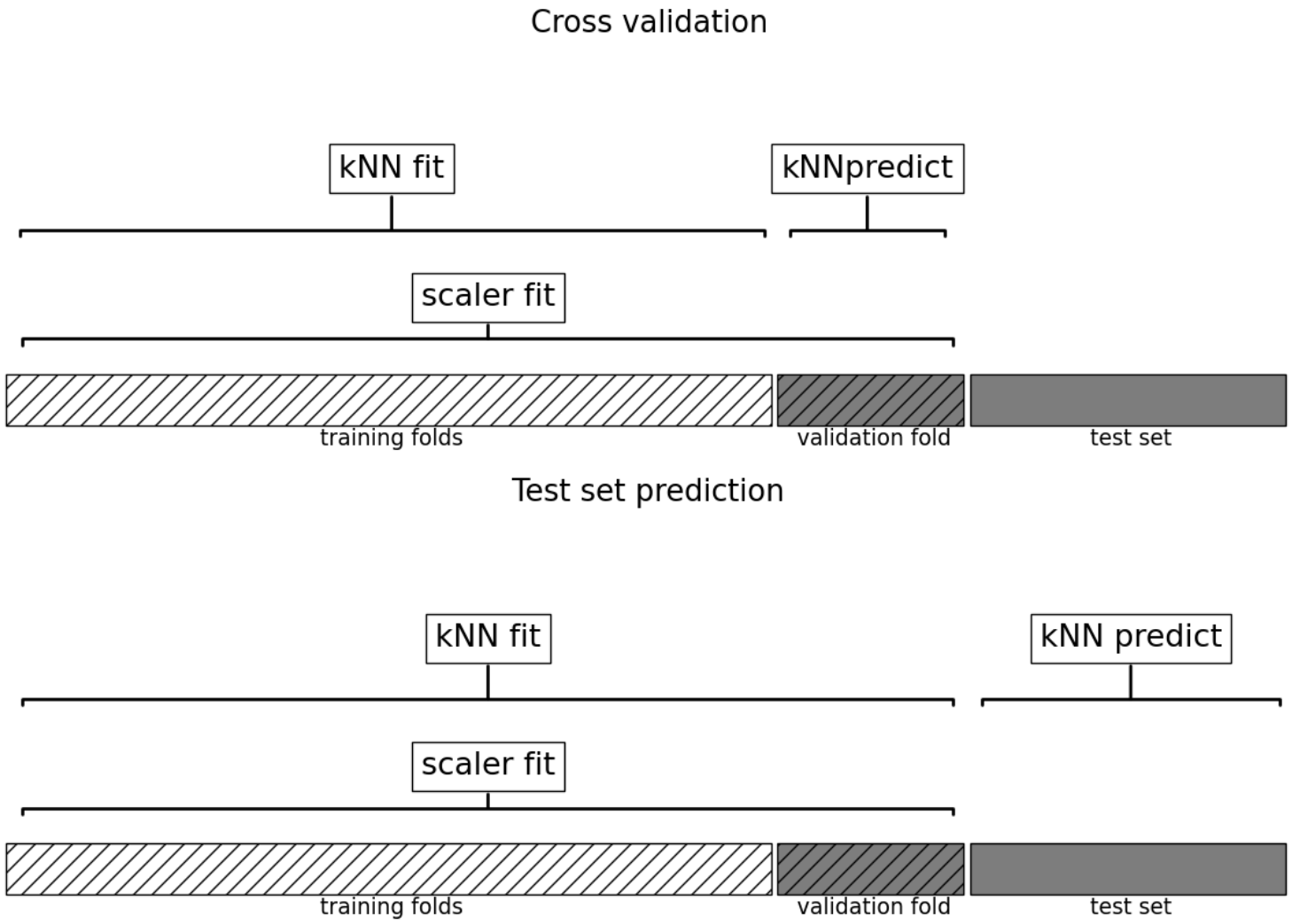
❌ Bad ML 3

- What's wrong with the approach below?

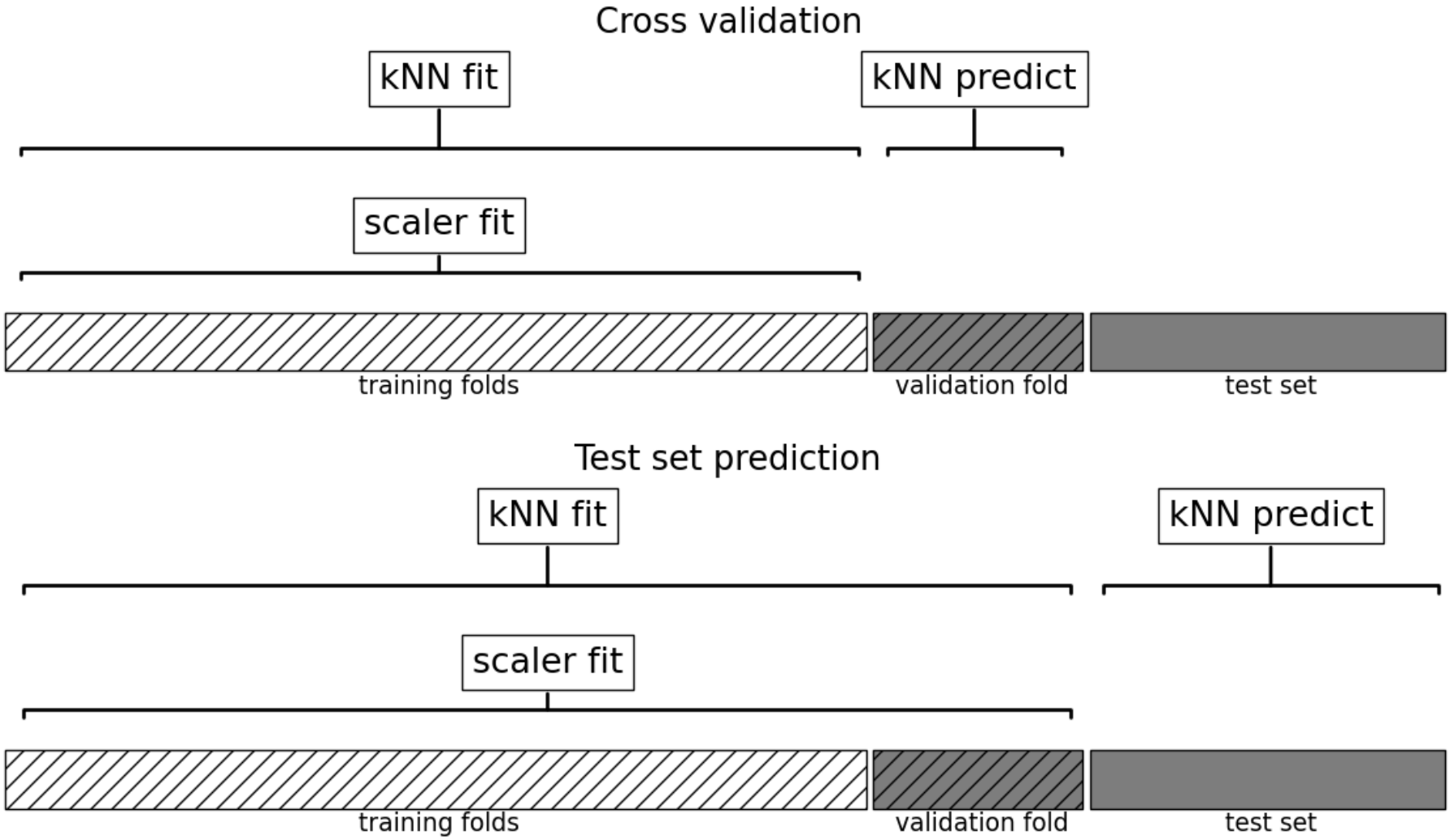
```
knn = KNeighborsClassifier()

scaler = StandardScaler()
scaler.fit(X_train_toy)
X_train_scaled = scaler.transform(X_train_toy)
X_test_scaled = scaler.transform(X_test_toy)
cross_val_score(knn, X_train_scaled, y_train_toy)
```

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

Improper preprocessing



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.

```
from sklearn.pipeline import make_pipeline

pipe_knn = make_pipeline(StandardScaler(), KNeighborsClassifier())

# Correct way to do cross validation without breaking the golden rule.
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

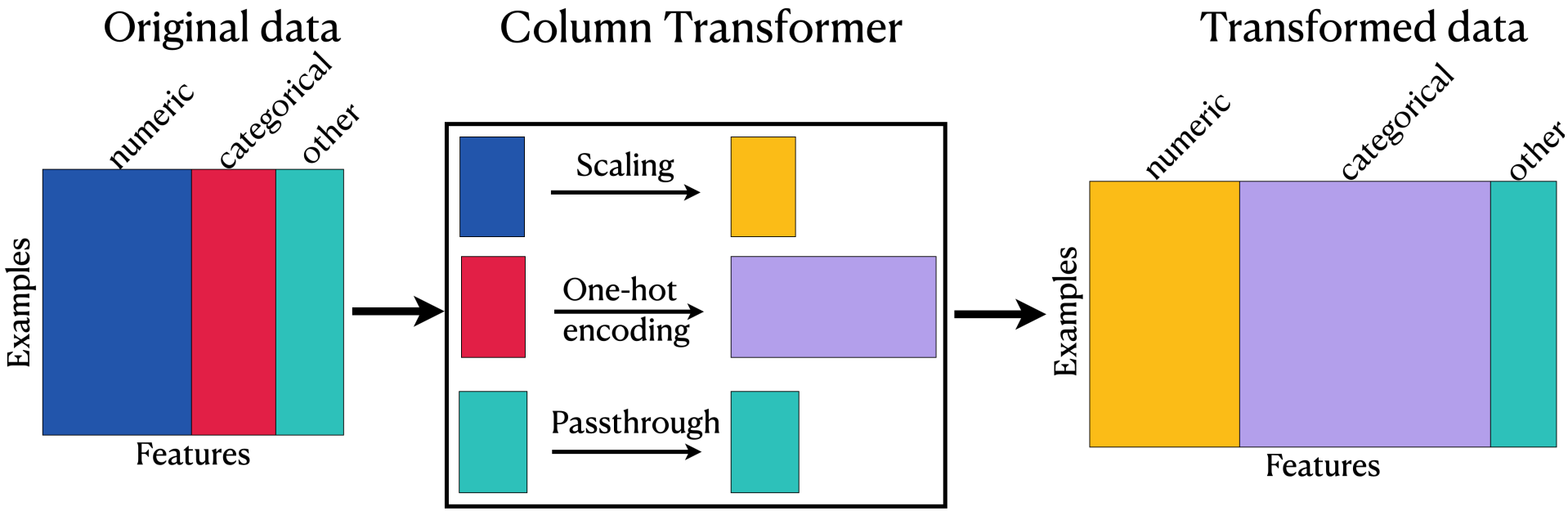
For this demo, each student should click this link to create a new repo in their accounts, then clone that repo locally to follow along with the demo from today.

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

- Navigate to the `cpsc330-2025S1` 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/YJHS

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.