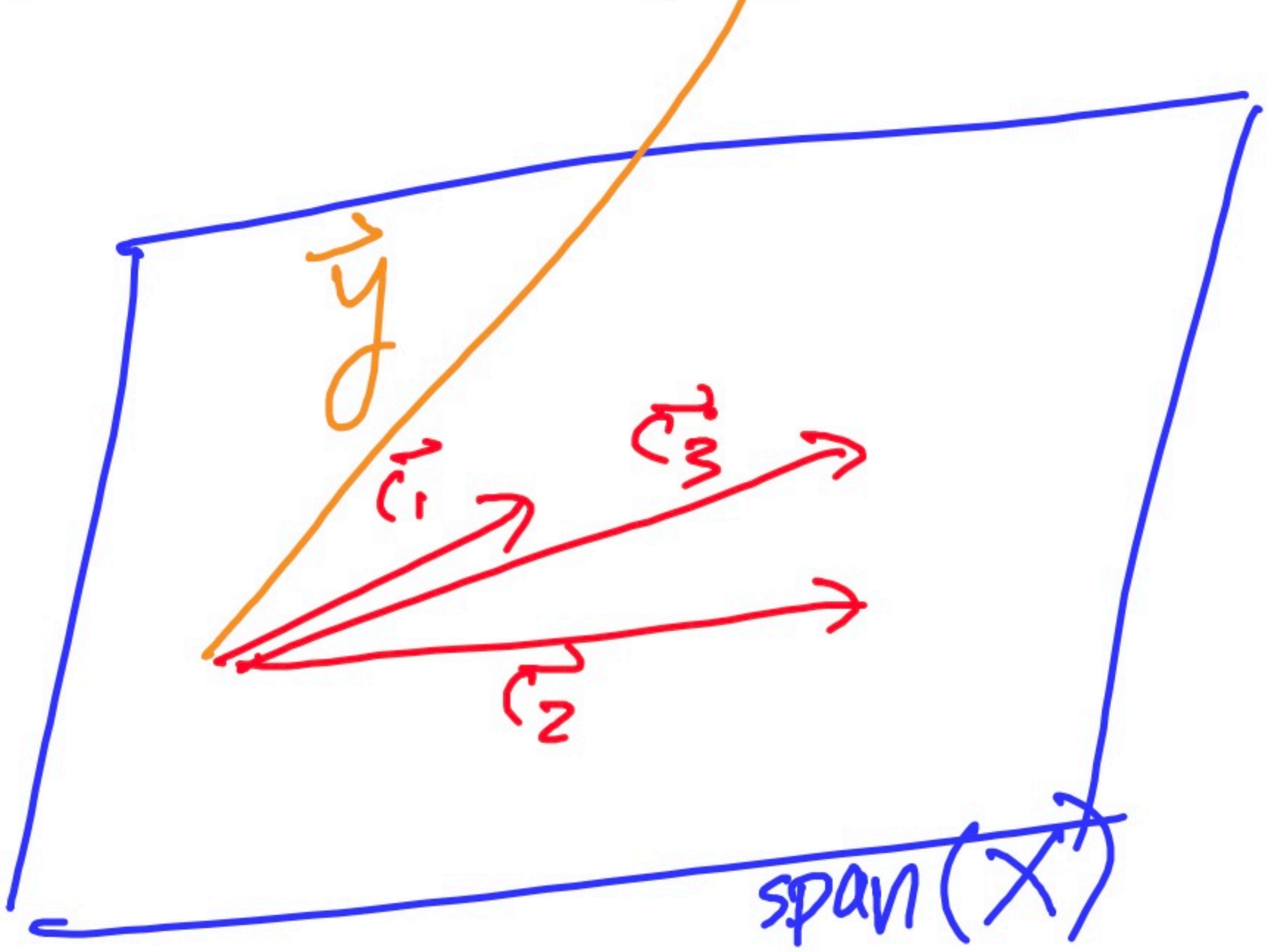
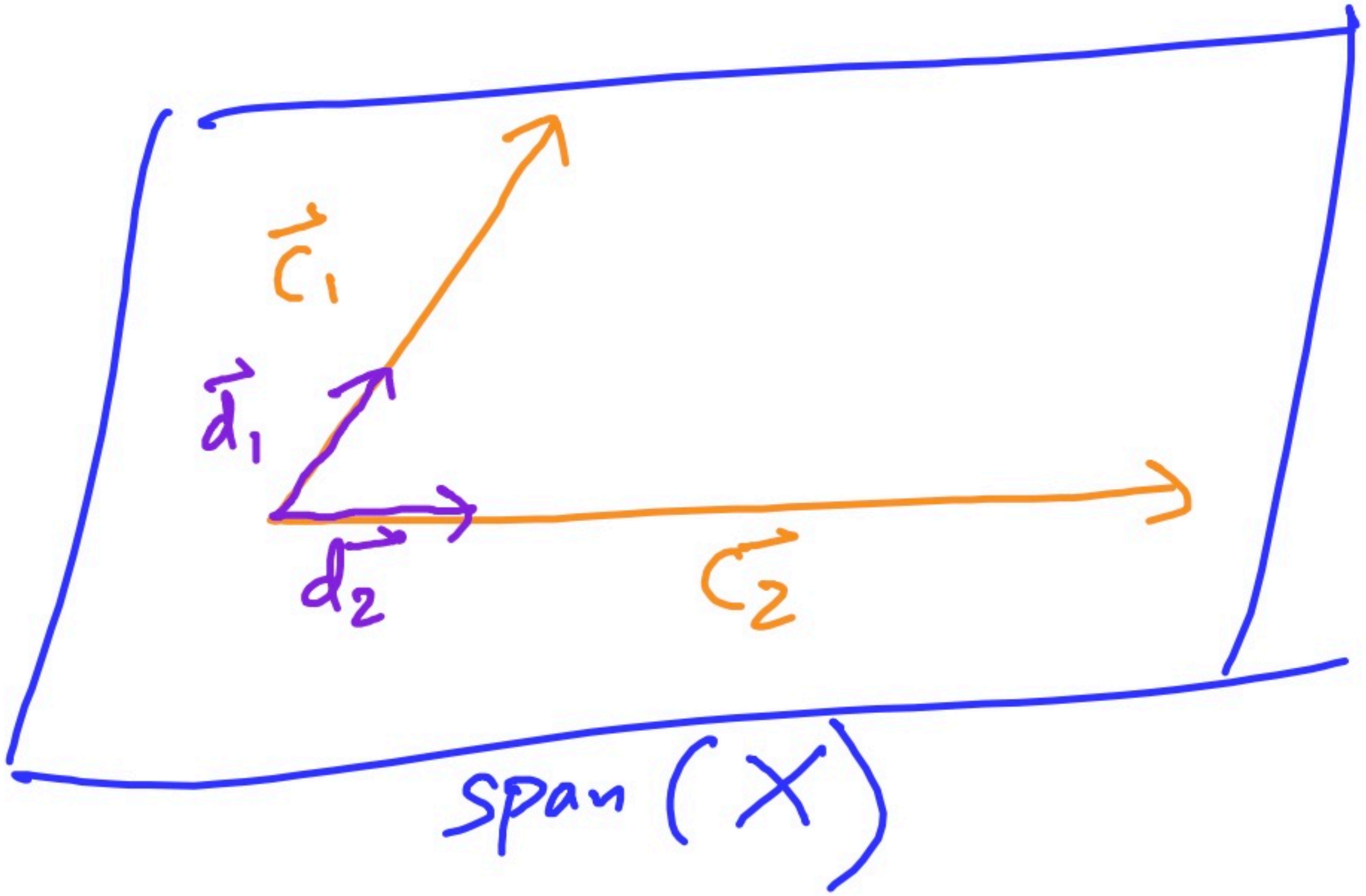
Brief recap: standardization (see annotated slides)



$$X = \begin{array}{c} \begin{array}{ccc} \vec{c}_1 & \vec{c}_2 & \vec{c}_3 \end{array} \\ \left[\begin{array}{ccc} 1 & x_1^{(1)} & x_1^{(2)} \\ 1 & x_2^{(1)} & x_2^{(2)} \\ \vdots & \vdots & \vdots \\ 1 & x_n^{(1)} & x_n^{(2)} \end{array}\right. \end{array}$$

Brief recap: standardization (see annotated slides)



Brief recap: standardization (see annotated slides)

$X_1, X_2, \ldots, X_n$

$$z_i = \frac{X_i - \overline{X}}{\sigma_X}$$

$z_1, z_2, \ldots, z_n$: mean of 0
SD of 1

Out[23]:

LinearRegression

```
LinearRegression()
```

2.78×10^{12}

- What are w_0^*, w_1^*, w_2^*, and the model's MSE?

In [25]:
```
people_two_feat.intercept_, people_two_feat.coef_
```

Out[25]:
```
(-82.59155502376602, array([-2.32e+11,  2.78e+12]))
```

In [24]:
```
mean_squared_error(y, people_two_feat.predict(X2))
```

Out[24]:
```
101.58844271417476
```

-2.32×10^{11}!

Redundant features

- Suppose in the first model, $w_0^* = -80$ and $w_1^* = 3$.

$$\text{predicted weight}_i = -80 + 3 \cdot \text{height in inches}_i$$

$$\frac{\text{height inches}}{12} = \text{height in feet}$$

- In the second model, we have:

$$\text{predicted weight}_i = w_0^* + w_1^* \cdot \text{height in inches}_i + w_2^* \cdot \text{height in feet}_i$$

$$5 + \frac{(-24)}{12} = 5 - 2 = 3$$

Infinitely many parameter choices

- **Issue**: There are an infinite number of w_1^* and w_2^* that satisfy $w_1^* + \frac{w_2^*}{12} = 3$!

$$\text{predicted weight}_i = -80 + 5 \cdot \text{height in inches}_i - 24 \cdot \text{height in feet}_i$$

Infinitely many parameter choices

- **Issue**: There are an infinite number of w_1^* and w_2^* that satisfy $w_1^* + \frac{w_2^*}{12} = 3$!

$$\text{predicted weight}_i = -80 + 5 \cdot \text{height in inches}_i - 24 \cdot \text{height in feet}_i$$

$$\text{predicted weight}_i = -80 - 1 \cdot \text{height in inches}_i + 48 \cdot \text{height in feet}_i$$

$$-1 + \frac{48}{12} = -1 + 4 = 3$$

Infinitely many parameter choices

- **Issue**: There are an infinite number of w_1^* and w_2^* that satisfy $w_1^* + \frac{w_2^*}{12} = 3$!

$$\text{predicted weight}_i = -80 + 5 \cdot \text{height in inches}_i - 24 \cdot \text{height in feet}_i$$

$$\text{predicted weight}_i = -80 - 1 \cdot \text{height in inches}_i + 48 \cdot \text{height in feet}_i$$

both make the same predictions!!!

⋮

infinitely many others too.

Infinitely many parameter choices

- **Issue**: There are an infinite number of w_1^* and w_2^* that satisfy $w_1^* + \frac{w_2^*}{12} = 3$!

$$\text{predicted weight}_i = -80 + 5 \cdot \text{height in inches}_i - 24 \cdot \text{height in feet}_i$$

$$\text{predicted weight}_i = -80 - 1 \cdot \text{height in inches}_i + 48 \cdot \text{height in feet}_i$$

⋮

infinitely many others too.

both make the same predictions!!!

12.1

- **Issue**: There are an infinite number of w_1^* and w_2^* that satisfy $w_1^* + \frac{w_2^*}{12} = 3$!

$10 \quad -7$

$$\text{predicted weight}_i = -80 + 5 \cdot \text{height in inches}_i - 24 \cdot \text{height in feet}_i$$

$$\frac{w_2^*}{12} = -7$$

$$\text{predicted weight}_i = -80 - 1 \cdot \text{height in inches}_i + 48 \cdot \text{height in feet}_i$$

$$w_2^* = -84$$

- Both hypothesis functions look very different, but actually make the same predictions.

- `model.coef_` could return either set of coefficients, or any other of the infinitely many options.

- But neither set of coefficients is **has any meaning!**

In [28]:
```
-80 + 10 * people['Height (Inches)'] - 84 * people['Height (Feet)']
```

Out[28]:
```
0        117.35
1        134.55
```

- Suppose we have the following fitted model:

  For illustration, assume `'weekend'` was originally a categorical feature with two possible values, `'Yes'` or `'No'`.

$$H(\vec{x}_i) = 1 - 3 \cdot \text{departure hour}_i + 2 \cdot (\text{weekend}_i == \text{Yes}) - 2 \cdot (\text{weekend}_i == \text{No})$$

1 0

0 1

- Suppose we have the following fitted model:

For illustration, assume `'weekend'` was originally a categorical feature with two possible values, `'Yes'` or `'No'`.

A $$H(\vec{x}_i) = 1 - 3 \cdot \text{departure hour}_i + 2 \cdot (\text{weekend}_i == \text{Yes}) - 2 \cdot (\text{weekend}_i == \text{No})$$

- This is equivalent to:

$$H(\vec{x}_i) = 10 - 3 \cdot \text{departure hour}_i - 7 \cdot (\text{weekend}_i == \text{Yes}) - 11 \cdot (\text{weekend}_i == \text{No})$$

i) weekend = Yes, departure hour = x_i

A $1 - 3x_i + 2 \cdot 1 - 2 \cdot 0 = 1 - 3x_i + 2 = 3 - 3x_i$

B $10 - 3x_i - 7 \cdot 1 - 11 \cdot 0 = 3 - 3x_i$

14.1

- This is equivalent to:

$$H(\vec{x}_i) = 10 - 3 \cdot \text{departure hour}_i - 7 \cdot (\text{weekend}_i == \text{Yes}) - 11 \cdot (\text{weekend}_i == \text{No})$$

- Note that for a particular row in the dataset, $\text{weekend}_i == \text{Yes} + \text{weekend}_i == \text{No}$ is always equal to 1.

$$X = \begin{array}{c} \begin{array}{cccc} \vec{1} & & \text{“Yes”} & \text{“No”} \end{array} \\ \begin{bmatrix} 1 & 8.45 & 0 & 1 \\ 1 & 11 & 0 & 1 \\ 1 & 7.39 & 1 & 0 \\ 1 & 9.98 & 1 & 0 \\ 1 & 10.45 & 0 & 1 \end{bmatrix} \end{array}$$

A possible design matrix for this model.

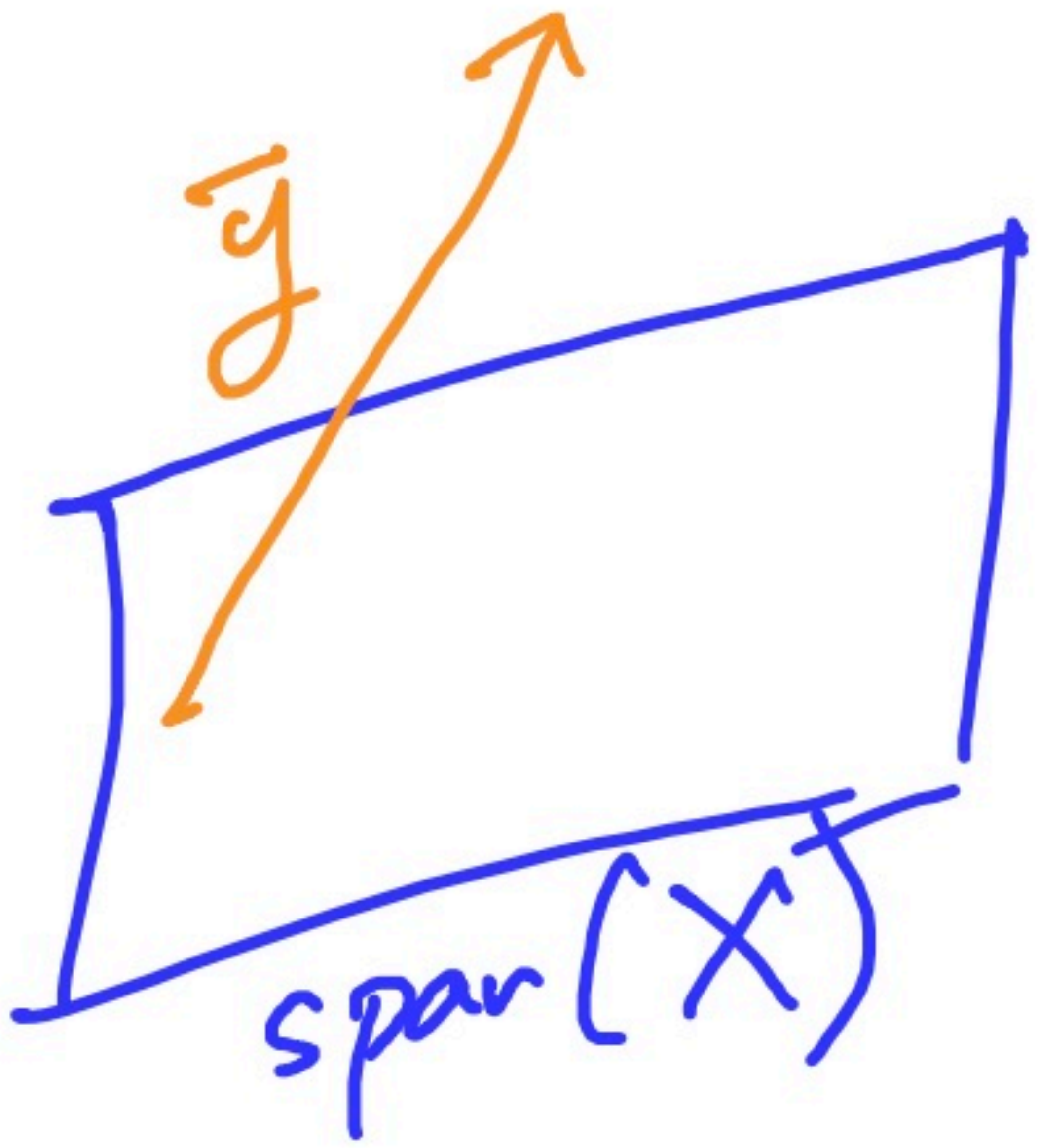
$$\text{Yes} + \text{No} = \vec{1}$$

localhost

One hot encoding and multicollinearity

$$X = \begin{bmatrix} 1 & 8.45 & 0 & 1 \\ 1 & 11 & 0 & 1 \\ 1 & 7.39 & 1 & 0 \\ 1 & 9.98 & 1 & 0 \\ 1 & 10.45 & 0 & 1 \end{bmatrix}$$

A possible design matrix for this model.



- The columns of the design matrix, X above are **not** linearly independent! The column of all 1s can be written as a linear combination of the weekend==Yes and weekend==No columns.

$$\text{column } 1 = \text{column } 3 + \text{column } 4$$

- This means that the design matrix is not **full rank**, which means that $X^T X$ is **not invertible**.

$$X = \begin{bmatrix} 1 & 8.45 & \cancel{0} & 1 \\ 1 & 11 & \cancel{0} & 1 \\ 1 & 7.39 & \cancel{1} & 0 \\ 1 & 9.98 & \cancel{1} & 0 \\ 1 & 10.45 & \cancel{0} & 1 \end{bmatrix}$$

$\vec{1}$ $\vec{No}$

$\vec{1} - \vec{No} = \vec{Yes}$

A possible design matrix for this model.

- The columns of the design matrix, X above are **not** linearly independent! The column of all 1s can be written as a linear combination of the weekend==Yes and weekend==No columns.

$$\text{column 1} = \text{column 3} + \text{column 4}$$

- This means that the design matrix is not **full rank**, which means that X^TX is **not invertible**.

- This means that there are **infinitely many possible solutions $\vec{w}^*$ to the normal equations, $(X^TX)\vec{w} = X^T\vec{y}$!**
  That's a problem, because we don't know which of these infinitely many solutions `model.coef_` will find for us, and it's impossible to interpret the resulting coefficients, as we saw two slides ago.

- **Solution**: Drop one of the one hot encoded columns. `OneHotEncoder` has an option to do this.

64	Thu	March

65 rows × 2 columns

In [32]:
```
ohe_drop_one.fit(df[['day', 'month']])
```

Out[32]:
```
OneHotEncoder
OneHotEncoder(drop='first')
```

- How many features did the resulting transformer create?

In [33]:
```
len(ohe_drop_one.get_feature_names_out())
```

Out[33]: 14

- Where did this number come from?

In [34]:
```
df['day'].nunique()
```

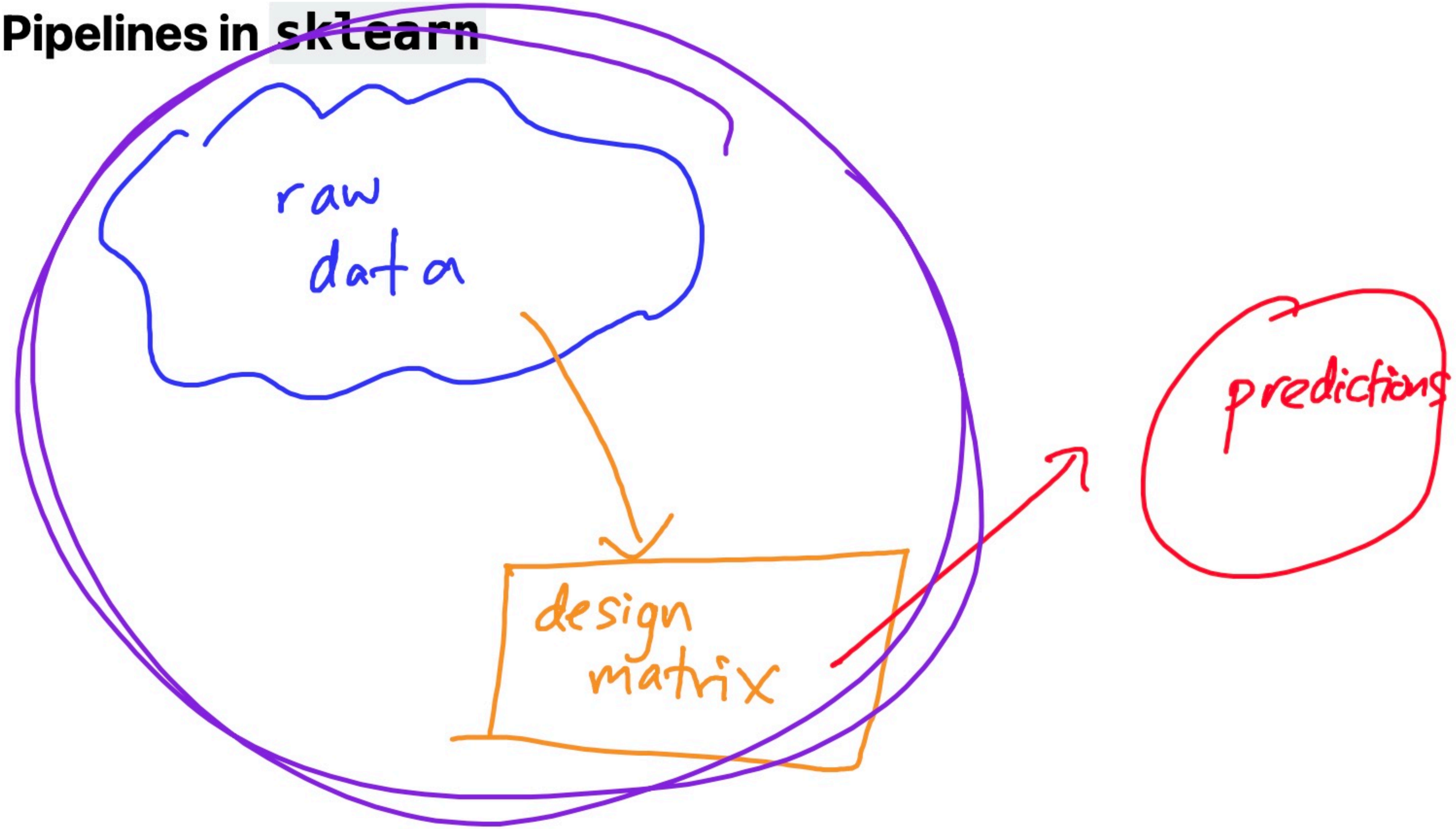
Out[34]: 5

4 + 10 = 14

In [35]:
```
df['month'].nunique()
```

Out[35]: 11

Pipelines in `sklearn`



```
example_vals
```

Out[53]:

```
60    27
61    29
62     4
63     5
64     7
Name: day_of_month, dtype: int32
```

In [76]:

```python
# Expression to convert from day of month to Week #.
def convert_to_week_num(s):
    return 'Week ' + ((s - 1) // 7 + 1).astype(str)

convert_to_week_num(example_vals)
```

Out[76]:

```
60    Week 4
61    Week 5
62    Week 1
63    Week 1
64    Week 1
Name: day_of_month, dtype: object
```

f: Series $\longrightarrow$ Series

but sometimes you need

f: DF $\longrightarrow$ DF

In [ ]:

```python
# The function that FunctionTransformer takes in
# itself takes in a Series/DataFrame, not a single element!
# Here, we're having that function return a new Series/DataFrame,
# depending on what's passed in to .tranform (experiment on your own).
from sklearn.pipeline import FunctionTransformer
week_converter = FunctionTransformer()
```

In [ ]:

```python
week_converter.transform(df[['day_of_month']])
```

`'departure_hour'` : Create degree 2 and degree 3 polynomial features.
`'day'` : One hot encode.
`'month'` : One hot encode.
`'day_of_month'` : Separate into five weeks, then one hot encode. ✅ **Use `day_of_month_transformer`**.

13 → Week 2 → OHE.

`'departure_hour'` : Create degree 2 and degree 3 polynomial features.
`'day'` : One hot encode.
`'month'` : One hot encode.
`'day_of_month'` : Separate into five weeks, then one hot encode. ✅ **Use `day_of_month_transformer`.**

important!

- Every other column only needs a single transformation.

To specify which transformations to apply to which columns, create a `ColumnTransformer`.

In [91]:
```
from sklearn.compose import ColumnTransformer, make_column_transformer
from sklearn.preprocessing import PolynomialFeatures
```

In [ ]:
```
preprocessing = make_column_transformer(
    (PolynomialFeatures(3, include_bias=False), ['departure_hour']),
    (OneHotEncoder(drop='first'), ['day', 'month']),
    (day_of_month_transformer, ['day_of_month']),
    remainder='drop'
)
preprocessing
```

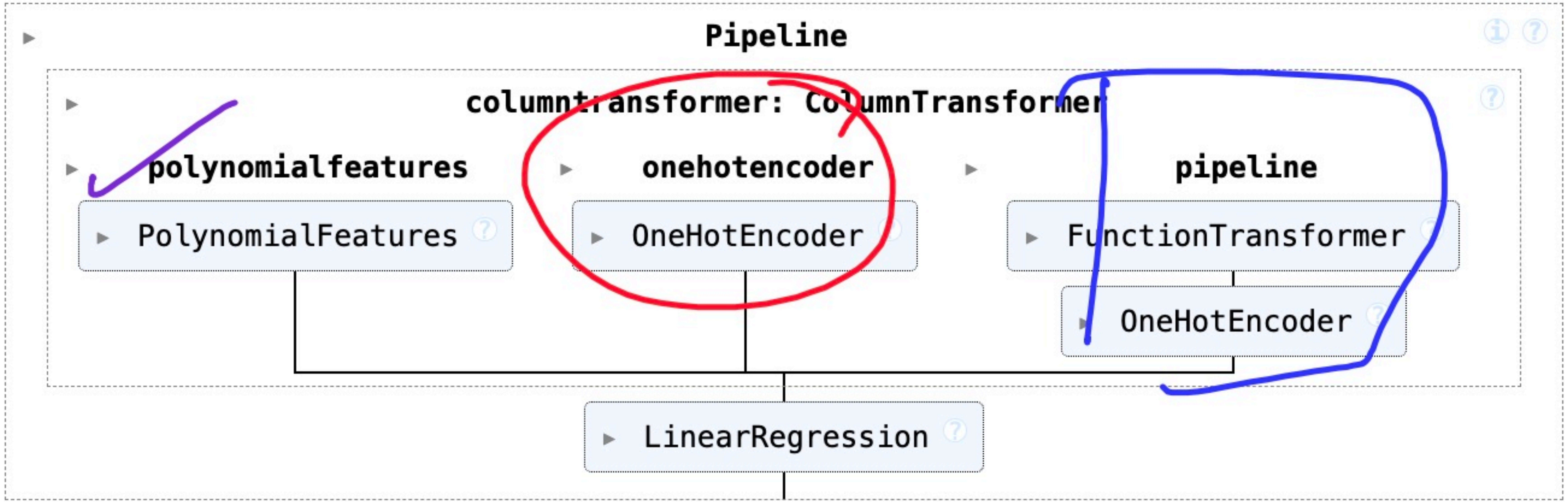
$$1 + 3 + (5-1) + (11-1) + (5-1)$$
$$= 4 + 4 + 10 + 4 = 22$$

Activity

How many columns does the final design matrix that `model` creates have? If you write code to determine the answer, make sure you can walk through the steps over the past few slides to figure out **why** the answer is what it is.

In [96]:
```
model
```

Out[96]:



How many columns does the final design matrix that `model` creates have? If you write code to determine the answer, make sure you can walk through the steps over the past few slides to figure out **why** the answer is what it is.

x_i = departure hour$_i$

$$\begin{bmatrix} - & x_1 & x_1^2 & x_1^3 & - \\ - & x_2 & x_2^2 & x_2^3 & - \\ & \vdots & \vdots & \vdots & \\ & x_n & x_n^2 & x_n^3 & \end{bmatrix}$$

In [100]:
```
len(model[-1].coef_)
```
Out[100]: 21

In [101]:
```
model[-1].intercept_
```
Out[101]: 397.93551417686405

In [96]:
```
model
```
Out[96]: