
multiverse

multiverse

Data Science Unit 3

Train-Test-Split

Before we start…

- ➔ Make sure you are comfortable
- ➔ Have water and maybe a strong coffee handy
- ➔ If you need a break… take it!
- ➔ If you need a stretch – please go ahead!
- ➔ Please mute yourselves if you are not talking
- ➔ Have your video on at all times

…and let's get started!

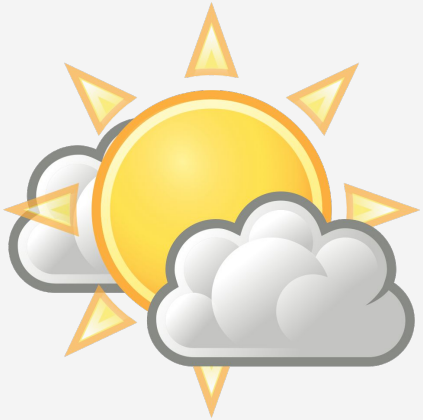
In this session we will…

1. **Define** error due to bias and variance
2. **Identify** the bias-variance trade off
3. **Describe** what overfitting and underfitting means in the context of model building
4. **Grasp** why train/test split is necessary
5. **Explore** k-folds

multiverse

Bias –Variance Trade Off

Bias – Variance Trade Off



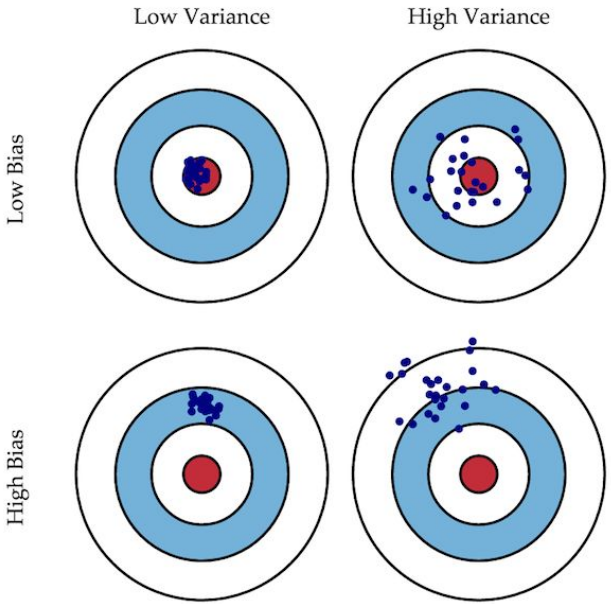


Fig. 1 Graphical illustration of bias and variance.

Bias – Variance Trade Off

Bias? Variance?

Conceptual Definitions

- ➔ Bias: How close are predictions to the actual values?
  - Roughly, whether or not our model aims on target
  - If the model cannot represent the data structure, our predictions could be consistent, but not accurate
- ➔ Variance: How variable are our predictions?
  - Roughly, whether or not our model is reliable
  - We will make slightly different predictions given slightly different training sets.

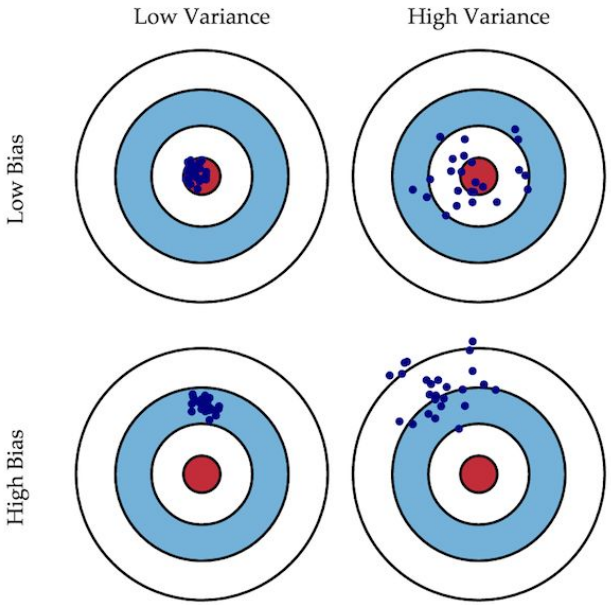


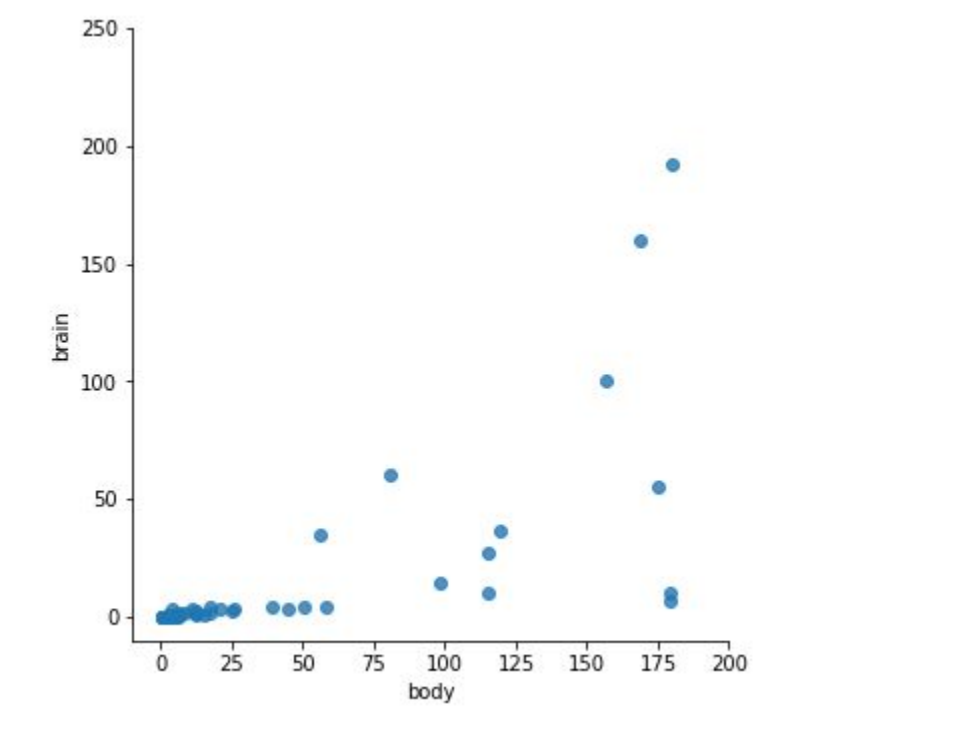
Fig. 1 Graphical illustration of bias and variance.

multiverse

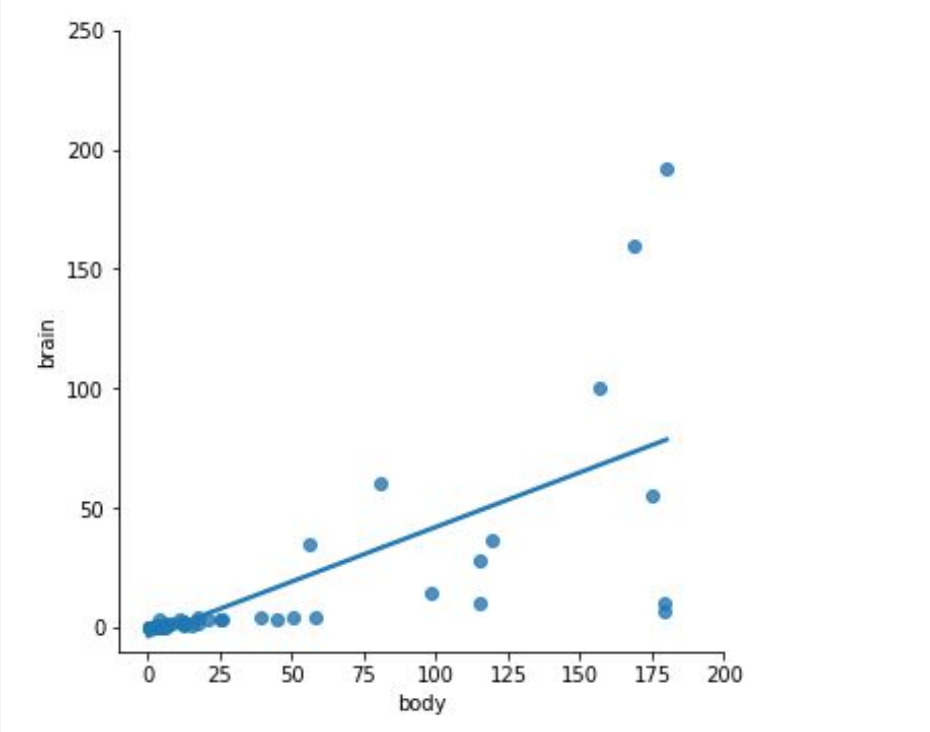
Mammal Mia!

Mammal Mia!

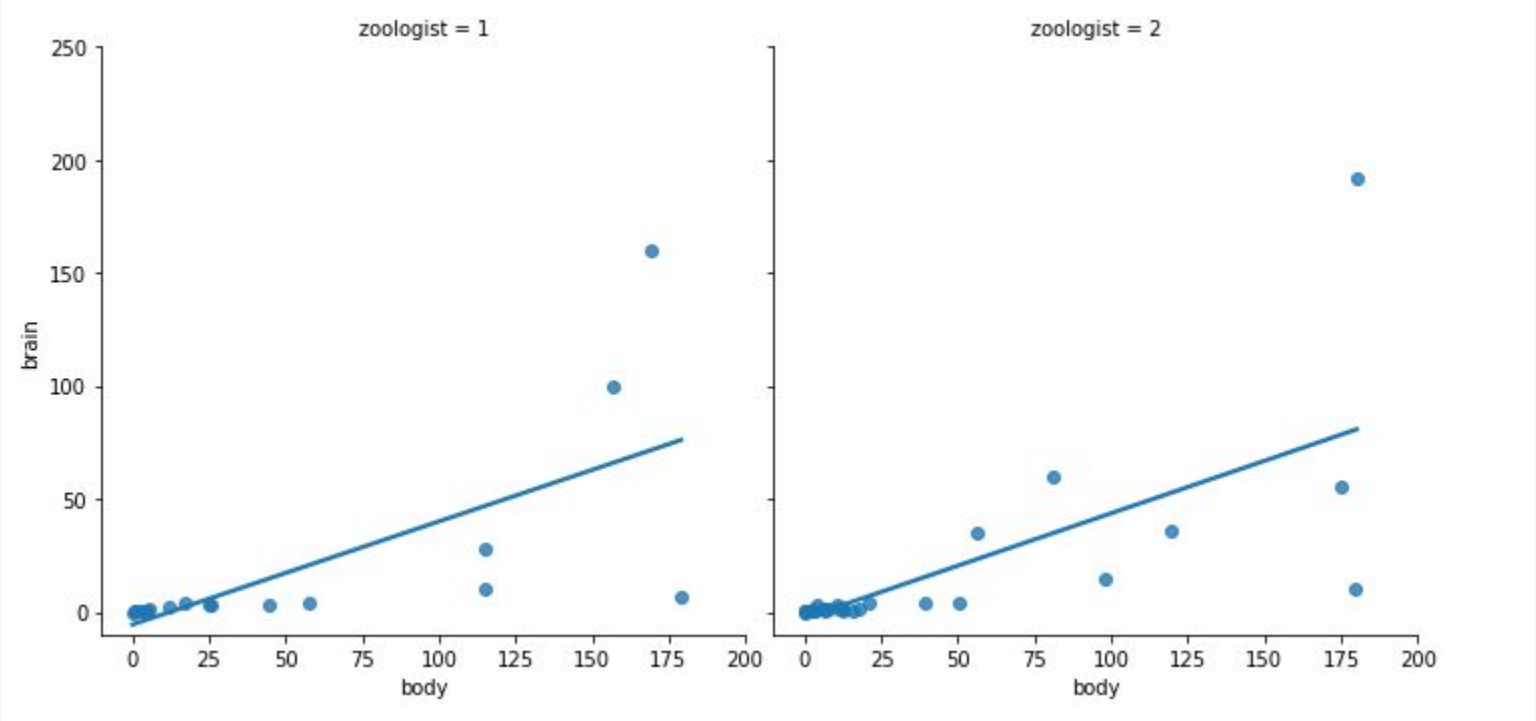
Bias – Variance Trade Off



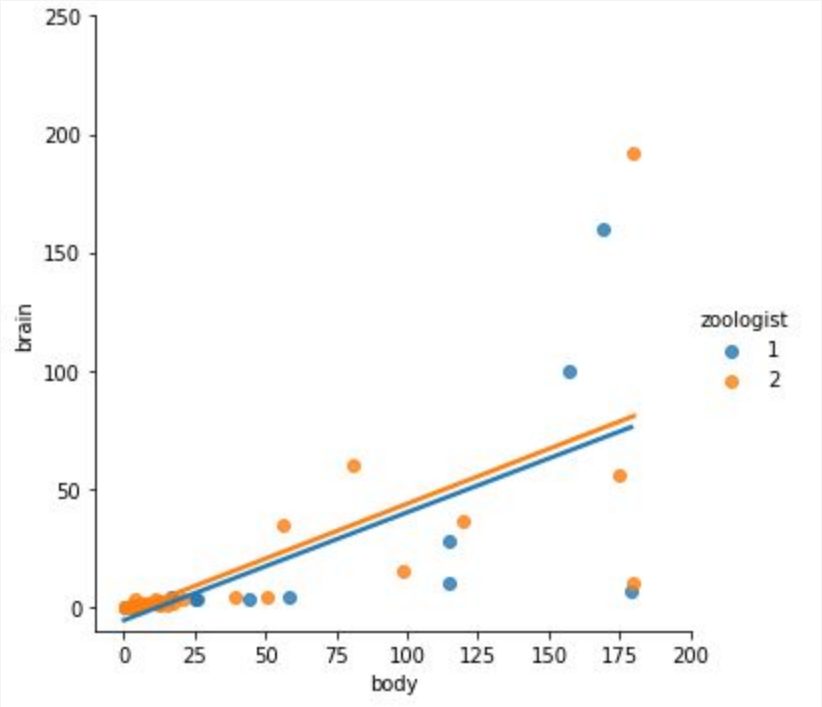
Bias – Variance Trade Off



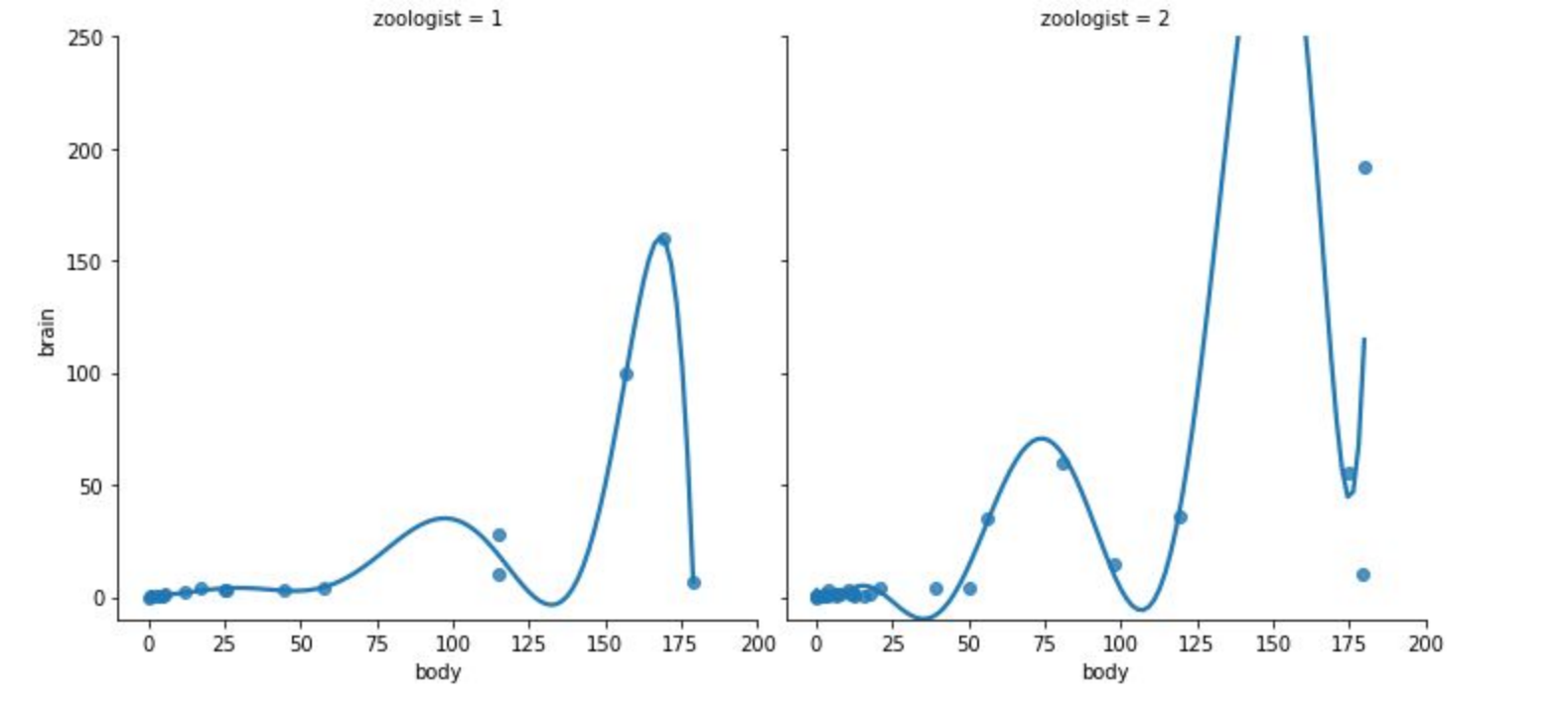
Bias – Variance Trade Off



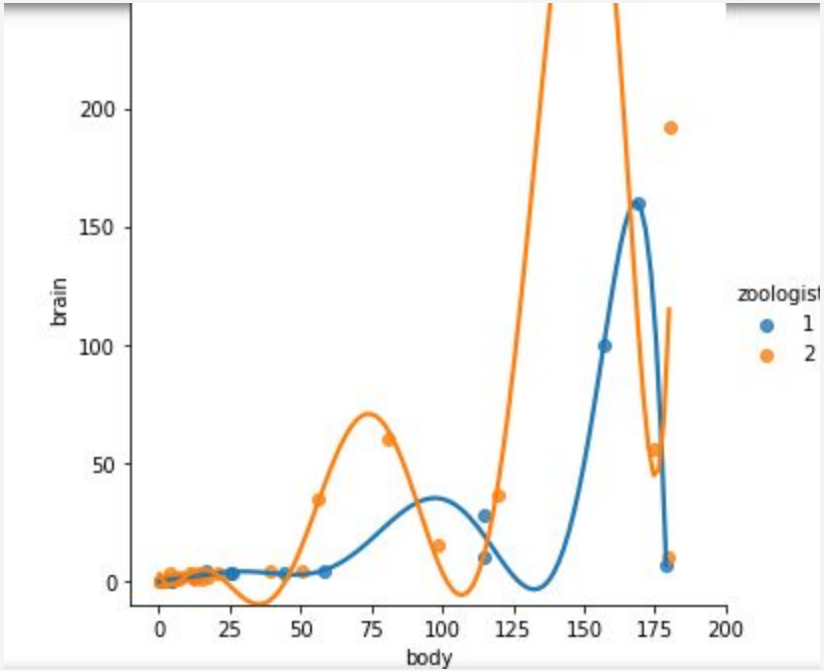
Bias – Variance Trade Off



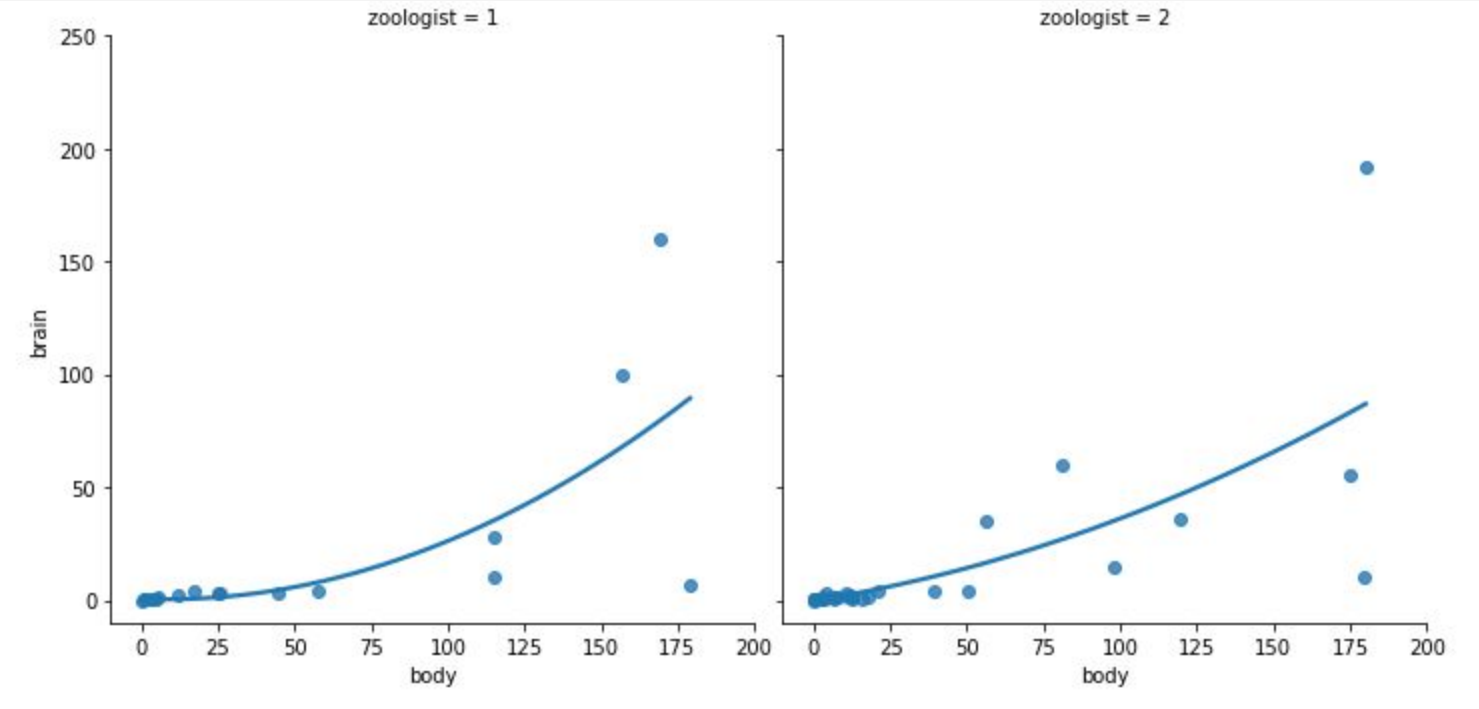
Bias – Variance Trade Off



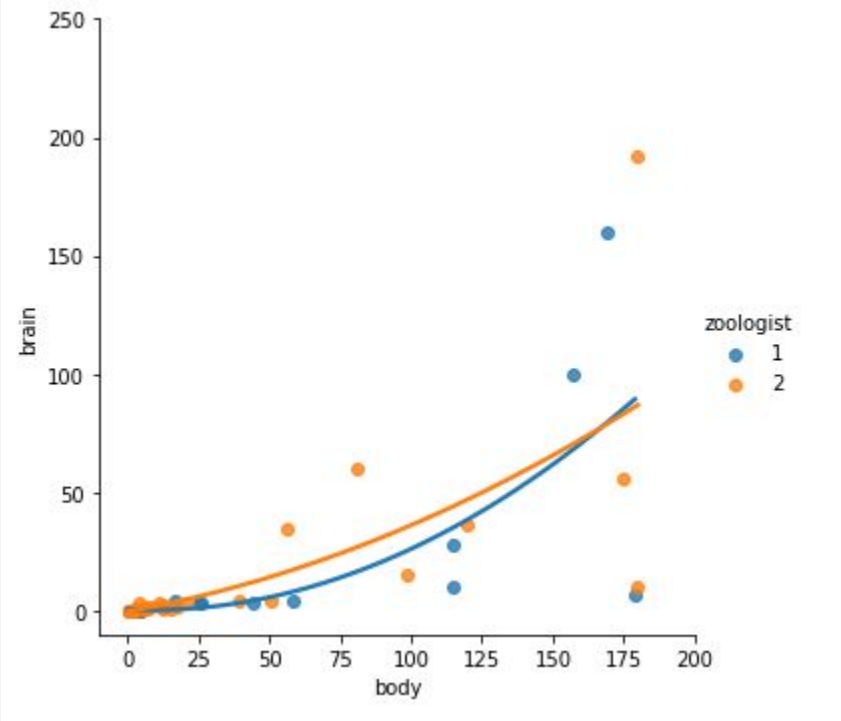
Bias – Variance Trade Off



Bias – Variance Trade Off



Bias – Variance Trade Off



Bias-Variance Trade Off

Zero Bias – Zero Variance?

multiverse

Train/Test Split

Train/Test Split

Problems with training and testing on the same data

➔ Our goal is to produce a model which works with **out-of-sample data**

➔ If we build the best possible model with the all the data we have, we will likely end up with **overly complex models** that won't necessarily generalise to new data

➔ Unnecessarily complex models **overfit** the training data

- They will do well when tested using the in-sample data
- They will do poorly with out-of-sample data

Train/Test Split

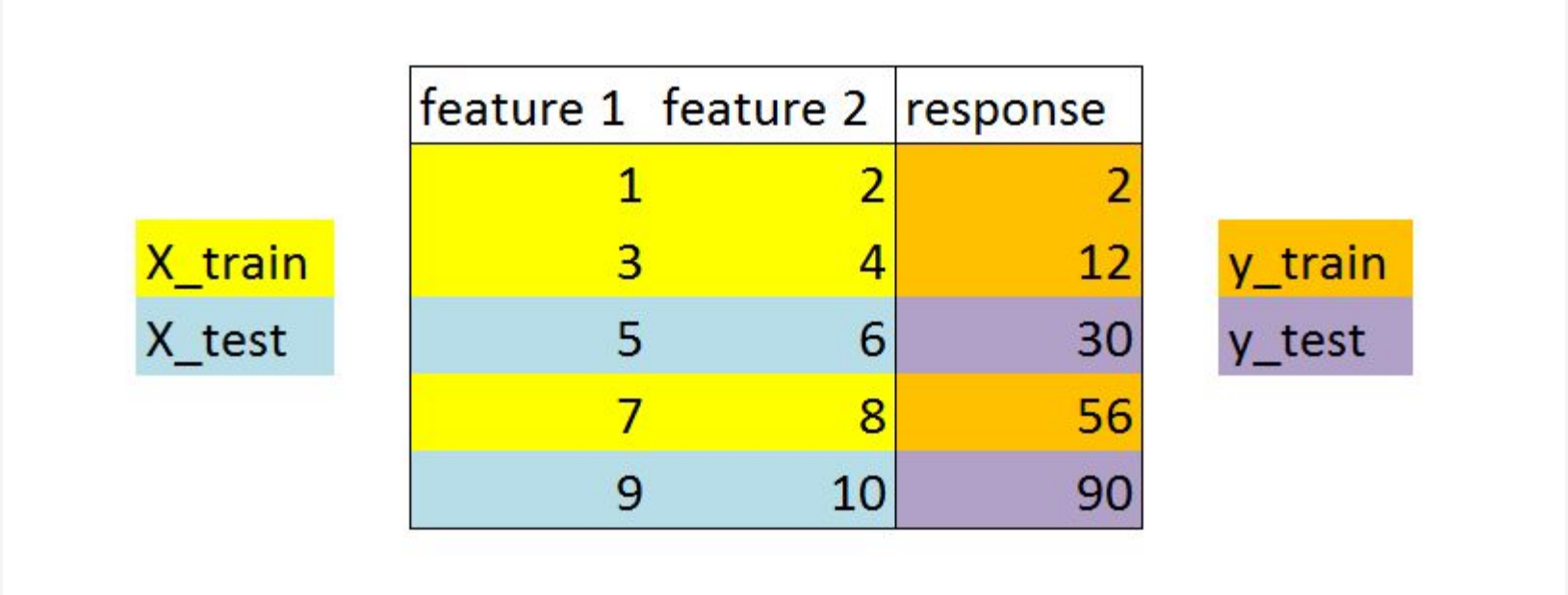
Training Set

Used to train the model

Testing Set

Used to evaluate the performance of the model

Train/Test Split



Train/Test Split

70/30? 80/20? 50/50?

Train/Test Split

```
df.head()
```

	x	y	z	a	b
0	0	46	73	-5	14
1	19	41	68	-6	0
2	13	59	41	-8	23
3	23	29	73	10	25
4	9	38	46	-2	-3

```
df.shape
```

```
(30, 5)
```

```
X=df[['x','y','z','a']]
y=df['b']
```

```
from sklearn.model_selection import train_test_split
```

```
X_train, X_test, y_train, y_test = train_test_split(X,y,train_size=0.8)
```

```
X_train.shape
```

```
(24, 4)
```

```
X_test.shape
```

```
(6, 4)
```

Train/Test Split

```
from sklearn.linear_model import LinearRegression
```

```
lr=LinearRegression().fit(X_train,y_train)
```

```
print('Training Set: ' +str(lr.score(X_train,y_train)))
print('Testing Set: ' +str(lr.score(X_test,y_test)))
```

```
Training Set: 0.2574217373671076
Testing Set: 0.2944580917043539
```

```
from sklearn.metrics import mean_squared_error
import numpy as np
```

```
pred=lr.predict(X_test)
print('Test RMSE: ' + str(np.sqrt(mean_squared_error(pred,y_test))))
```

```
Test RMSE: 12.873381642137028
```

Train/Test Split

```
from sklearn.preprocessing import StandardScaler
```

```
scaler=StandardScaler()
X_train_std=scaler.fit_transform(X_train)
X_test_std=scaler.transform(X_test)
```

```
lr=LinearRegression().fit(X_train_std,y_train)
```

```
print('Training Set: ' +str(lr.score(X_train_std,y_train)))
print('Testing Set: ' +str(lr.score(X_test_std,y_test)))
```

```
Training Set: 0.2574217373671076
Testing Set: 0.2944580917043538
```

multiverse

Let’s Practice

multiverse

Cross – Validation

Cross – Validation

Cross – Validation

```
from sklearn.model_selection import cross_val_score
```

```
lr=LinearRegression()
```

```
print(cross_val_score(lr,X,y))
```

```
[-0.49289844  0.02137015  0.53441265  0.32079123  0.16775272]
```

```
print(cross_val_score(lr,X,y).mean())
```

```
0.11028566112311125
```

Cross – Validation

```
from sklearn.model_selection import KFold
```

```
kf=KFold(n_splits=10,shuffle=True)
```

```
print(cross_val_score(lr, X, y, cv=kf))
```

```
[  0.19857958  -1.20001715   0.56336159 -12.54072382  -0.18276669
   0.61448061 -10.82490416  -0.99649346  -3.95017985   0.34803703]
```

```
print(cross_val_score(lr, X, y, cv=kf).mean())
```

```
-0.5875140464737795
```

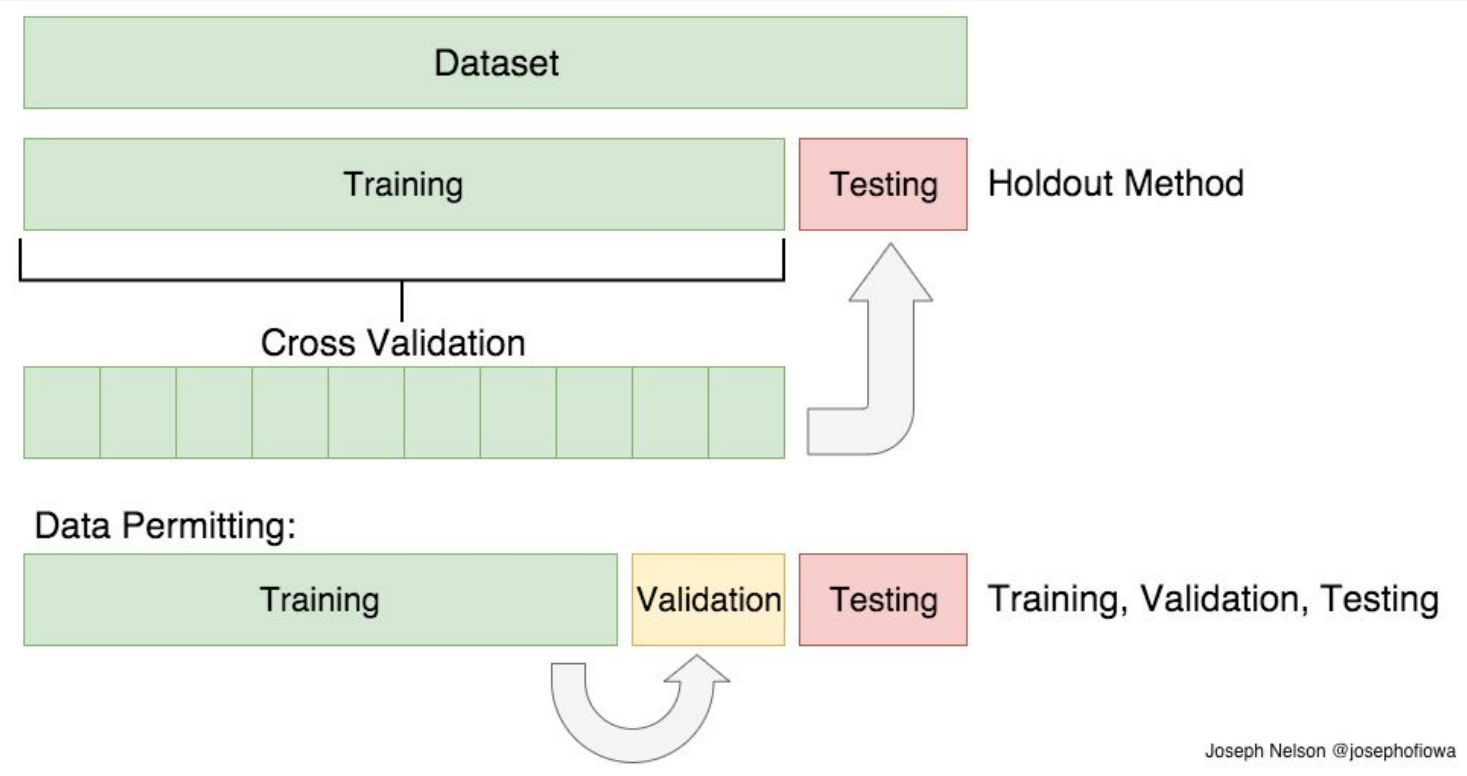
multiverse

Let’s Practice

multiverse

Three Way Split

Three Way Split



Three Way Split

```
from sklearn.linear_model import RidgeCV
```

```
X_train, X_test, y_train, y_test=train_test_split(X,y,train_size=0.7)
```

```
ridge=RidgeCV(alphas=[0.1,1.0,10.0],cv=5)
```

```
ridge.fit(X_train,y_train)
```

```
RidgeCV(alphas=array([ 0.1,  1. , 10. ]), cv=5, fit_intercept=True,
        gcv_mode=None, normalize=False, scoring=None, store_cv_values=False)
```

```
ridge.score(X_test,y_test)
```

```
0.7585875138886132
```

```
ridge.alpha_
```

```
0.1
```

Three Way Split

```
: ridge=RidgeCV()
```

```
Init signature:
RidgeCV(
    alphas=(0.1, 1.0, 10.0),
    fit_intercept=True,
```

```
ridge=RidgeCV()
```

```
Init signature:
RidgeCV(
    alphas=(0.1, 1.0, 10.0),
    fit_intercept=True,
    normalize=False,
    scoring=None,
    cv=None,
    gcv_mode=None,
    store_cv_values=False,
)
```

multiverse

Let’s Practice

multiverse

Summary

Summary

1. Train on the entire dataset
2. Train/Test Split
3. Cross-Validation
4. Three-Way Split

multiverse

Let’s Practice