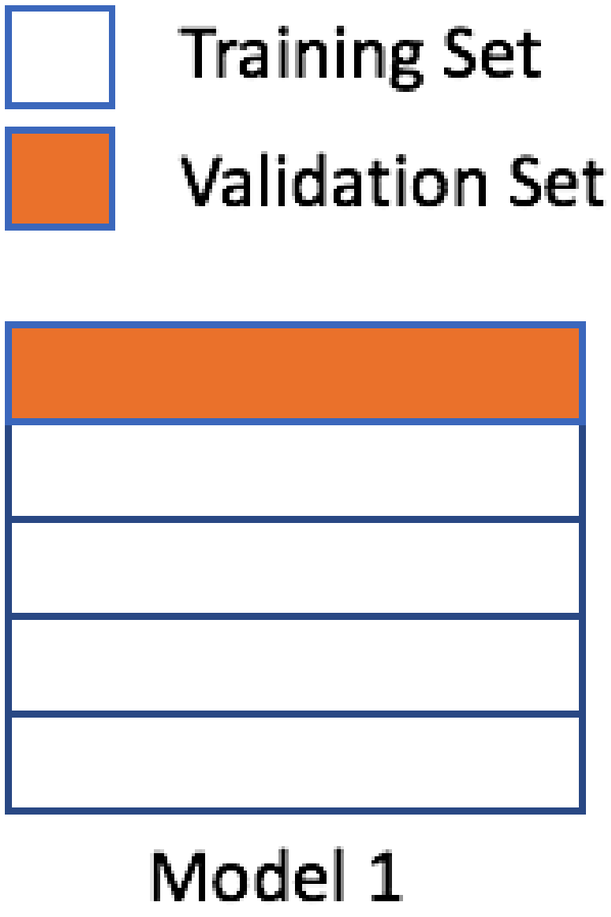
The problems with holdout sets

MODEL VALIDATION IN PYTHON

Kasey Jones
Data Scientist

datacamp

Transition validation




```
X_train, X_val, y_train, y_val =
    train_test_split(X, y,
    test_size=0.2)

rf = RandomForestRegressor()

rf.fit(X_train, y_train)
out_of_sample = rf.predict(X_test)
print(mae(y_test, out_of_sample))
```

```
10.24
```

Traditional training splits

```
cd = pd.read_csv("candy-data.csv")
s1 = cd.sample(60, random_state=1111)
s2 = cd.sample(60, random_state=1112)
```

Overlapping candies:

```
print(len([i for i in s1.index if i in s2.index]))
```

```
39
```

Traditional training splits

Chocolate Candies:

```
print(s1.chocolate.value_counts()[0])
print(s2.chocolate.value_counts()[0])
```

```
34
30
```

The split matters

Sample 1 Testing Error

```
print('Testing error: {0:.2f}'.format(mae(s1_y_test, rfr.predict(s1_X_test))))
```

```
10.32
```

Sample 2 Testing Error

```
print('Testing error: {0:.2f}'.format(mae(s2_y_test, rfr.predict(s2_X_test))))
```

```
11.56
```

Train, validation, test

```
X_temp, X_val, y_temp, y_val = train_test_split(..., random_state=1111)
X_train, X_test, y_train, y_test = train_test_split(..., random_state=1111)

rfr = RandomForestRegressor(n_estimators=25, random_state=1111, max_features=4)
rfr.fit(X_train, y_train)
print('Validation error: {0:.2f}'.format(mae(y_test, rfr.predict(X_test))))
```

```
9.18
```

```
print('Testing error: {0:.2f}'.format(mae(y_val, rfr.predict(X_val))))
```

```
8.98
```

Round 2

```
X_temp, X_val, y_temp, y_val = train_test_split(..., random_state=1171)
X_train, X_test, y_train, y_test = train_test_split(..., random_state=1171)

rfr = RandomForestRegressor(n_estimators=25, random_state=1111, max_features=4)
rfr.fit(X_train, y_train)
print('Validation error: {0:.2f}'.format(mae(y_test, rfr.predict(X_test))))
```

```
8.73
```

```
print('Testing error: {0:.2f}'.format(mae(y_val, rfr.predict(X_val))))
```

```
10.91
```

Holdout set exercises

MODEL VALIDATION IN PYTHON

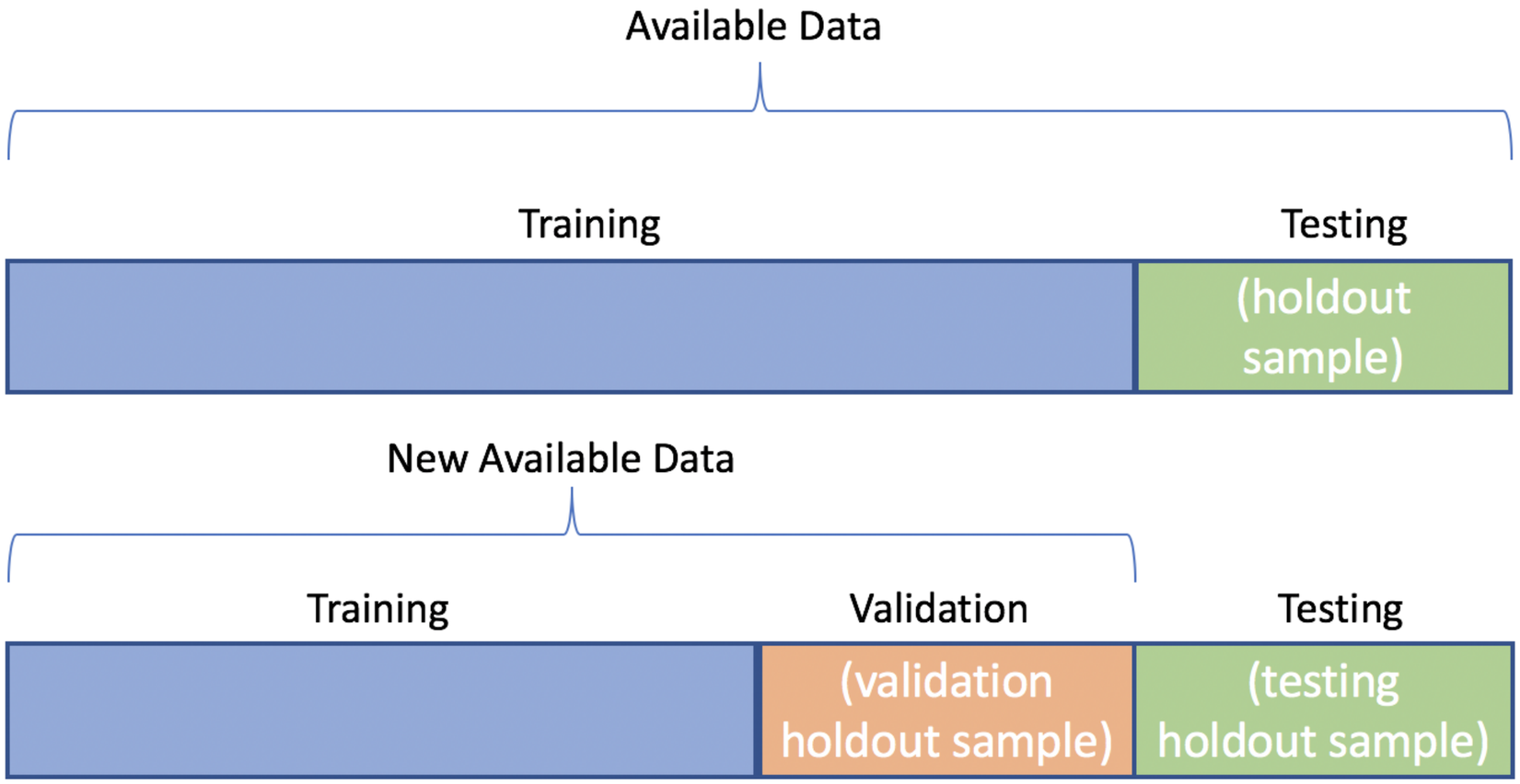
Cross-validation

MODEL VALIDATION IN PYTHON

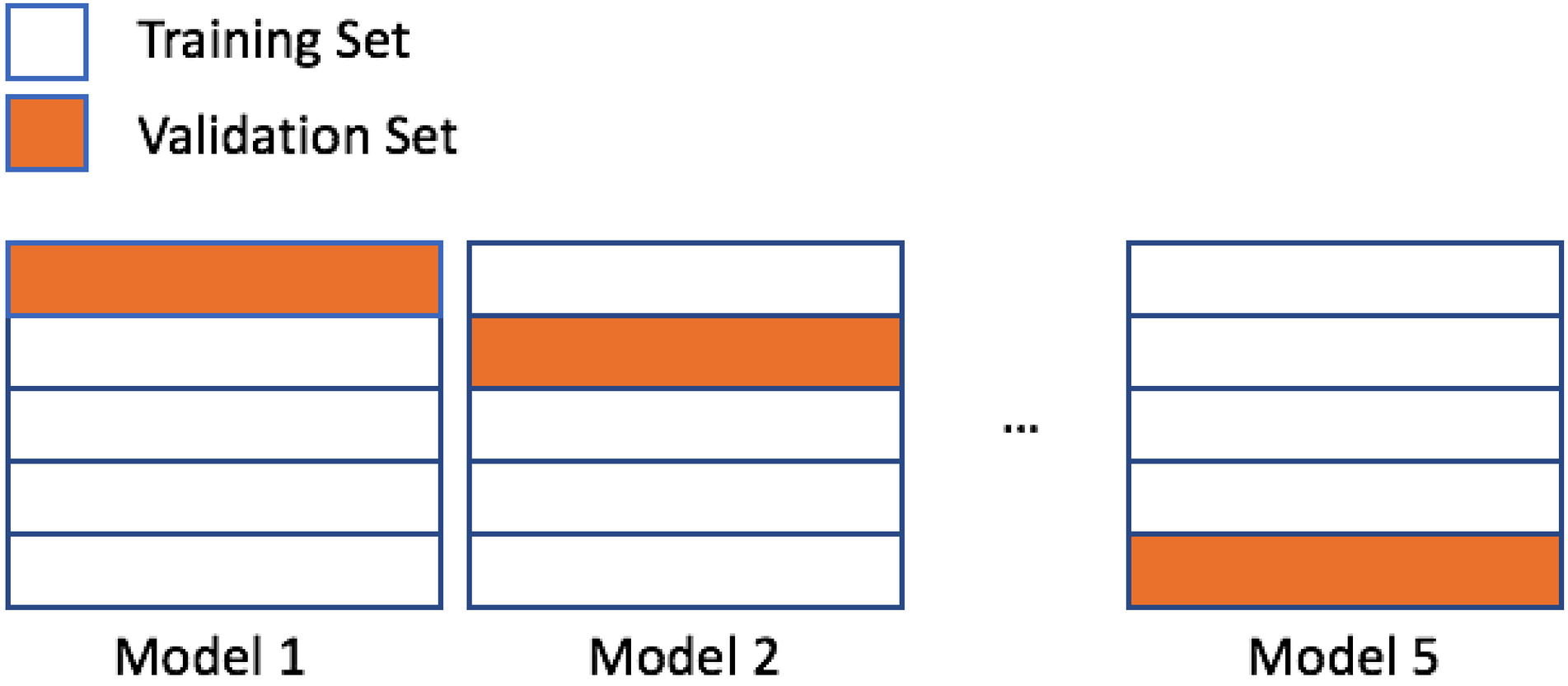
Kasey Jones
Data Scientist

datacamp

Cross-validation



Cross-validation



`n_splits` : number of cross-validation splits

`shuffle` : boolean indicating to shuffle data before splitting

`random_state` : random seed

```
from sklearn.model_selection import KFold

X = np.array(range(40))
y = np.array([0] * 20 + [1] * 20)

kf = KFold(n_splits=5)
splits = kf.split(X)
```

```
kf = KFold(n_splits=5)
splits = kf.split(X)
for train_index, test_index in splits:
    print(len(train_index), len(test_index))
```

```
32 8 32 8 32 8 32 8 32 8
```

```
# Print one of the index sets:
print(train_index, test_index)
```

```
[ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 ...]
[32 33 34 35 36 37 38 39]
```

```
rfr = RandomForestRegressor(n_estimators=25, random_state=1111)
errors = []
for train_index, val_index in splits:
    X_train, y_train = X[train_index], y[train_index]
    X_val, y_val = X[val_index], y[val_index]

    rfr.fit(X_train, y_train)
    predictions = rfr.predict(X_val)
    errors.append(<some_accuracy_metric>)
print(np.mean(errors))
```

```
4.25
```

Practice time

MODEL VALIDATION IN PYTHON

sklearn's cross_val_score()

MODEL VALIDATION IN PYTHON

Kasey Jones
Data Scientist

datacamp

cross_val_score()

```
from sklearn.model_selection import cross_val_score
from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier()
```

`estimator` : the model to use

`X` : the predictor dataset

`y` : the response array

`cv` : the number of cross-validation splits

```
cross_val_score(estimator=rfc, X=X, y=y, cv=5)
```

Using scoring and make_scorer

The cross_val_score `scoring` parameter:

```
# Load the Methods
from sklearn.metrics import mean_absolute_error, make_scorer
```

```
# Create a scorer
mae_scorer = make_scorer(mean_absolute_error)
```

```
# Use the scorer
cross_val_score(<estimator>, <X>, <y>, cv=5, scoring=mae_scorer)
```

Load all of the `sklearn` methods

```
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import cross_val_score
from sklearn.metrics import mean_squared_error, make_scorer
```

Create a model and a scorer

```
rfc = RandomForestRegressor(n_estimators=20, max_depth=5, random_state=1111)
mse = make_scorer(mean_squared_error)
```

Run `cross_val_score()`

```
cv_results = cross_val_score(rfc, X, y, cv=5, scoring=mse)
```

Accessing the results

```
print(cv_results)
```

```
[196.765, 108.563, 85.963, 222.594, 140.942]
```

Report the mean and standard deviation:

```
print('The mean: {}'.format(cv_results.mean()))
print('The std: {}'.format(cv_results.std()))
```

```
The mean: 150.965
```

```
The std: 51.676
```

Let's practice!

MODEL VALIDATION IN PYTHON

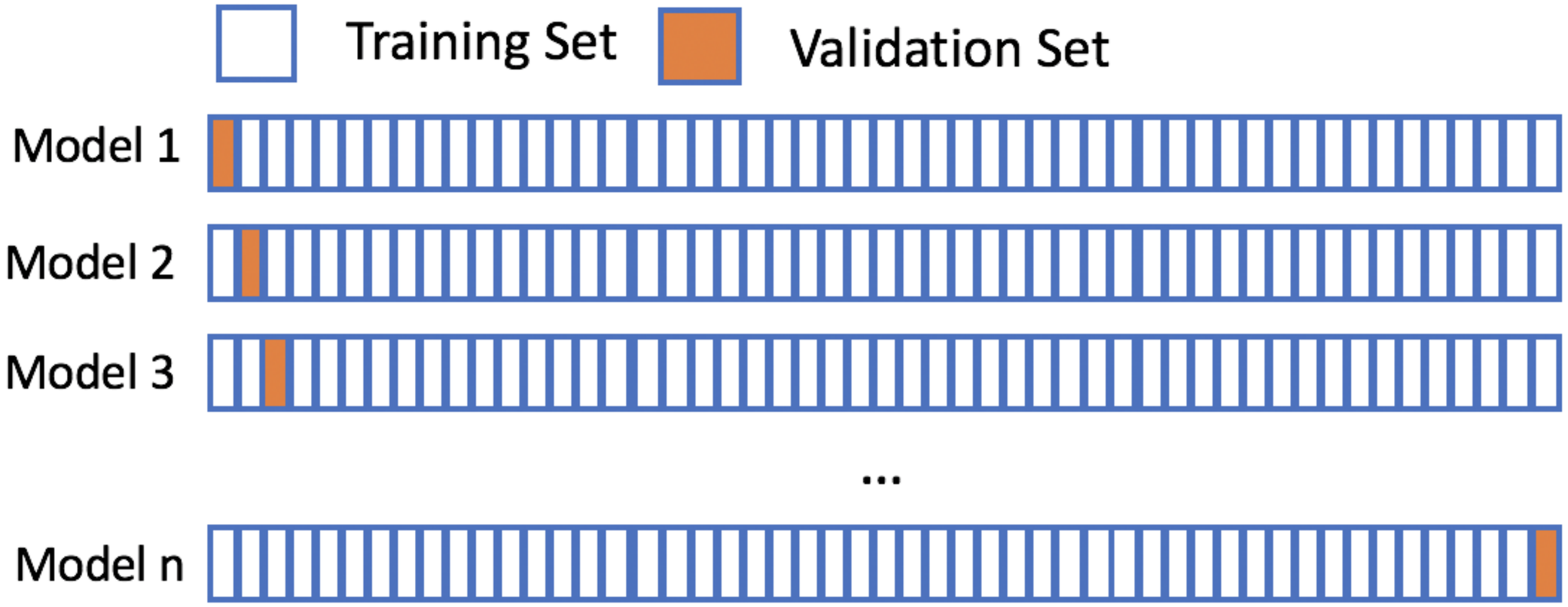
Leave-one-out-cross-validation (LOOCV)

MODEL VALIDATION IN PYTHON

Kasey Jones
Data Scientist

datacamp

LOOCV



When to use LOOCV?

Use when:

- The amount of training data is limited
- You want the absolute best error estimate for new data

Be cautious when:

- Computational resources are limited
- You have a lot of data
- You have a lot of parameters to test

LOOCV Example

```
n = X.shape[0]
mse = make_scorer(mean_squared_error)
cv_results = cross_val_score(estimator, X, y, scoring=mse, cv=n)
```

```
print(cv_results)
```

```
[5.45, 10.52, 6.23, 1.98, 11.27, 9.21, 4.65, ... ]
```

```
print(cv_results.mean())
```

```
6.32
```

Let's practice

MODEL VALIDATION IN PYTHON