ROBOFIED

Cross Validation and its Types

FUNDAMENTALS EXPLAINED
BY ROBOFIED

ROBOFIED

01

What is Cross-Validation?

Cross-Validation is a data re-sampling technique in which we keep some percentage of the data for testing and the remaining data is used for training.

Steps:

1. Keep some part of data as testing set.
2. Train the model on rest of the data aka training set.
3. Test the model on the testing set.

ROBOFIED

02

Need?

Used to check the predictive performance of the model by investigating how well it performs on the test or unseen data.

Allows us to compare different machine learning model and get a sense of how well they will perform in practice.

ROBOFIED

03

Types of Cross-Validation

1. Hold-out Validation
2. Leave-One-Out Cross-Validation (LOOCV)
3. K-Fold Cross-Validation
4. Stratified K-Fold Cross-Validation
5. Cross-validation for Time series

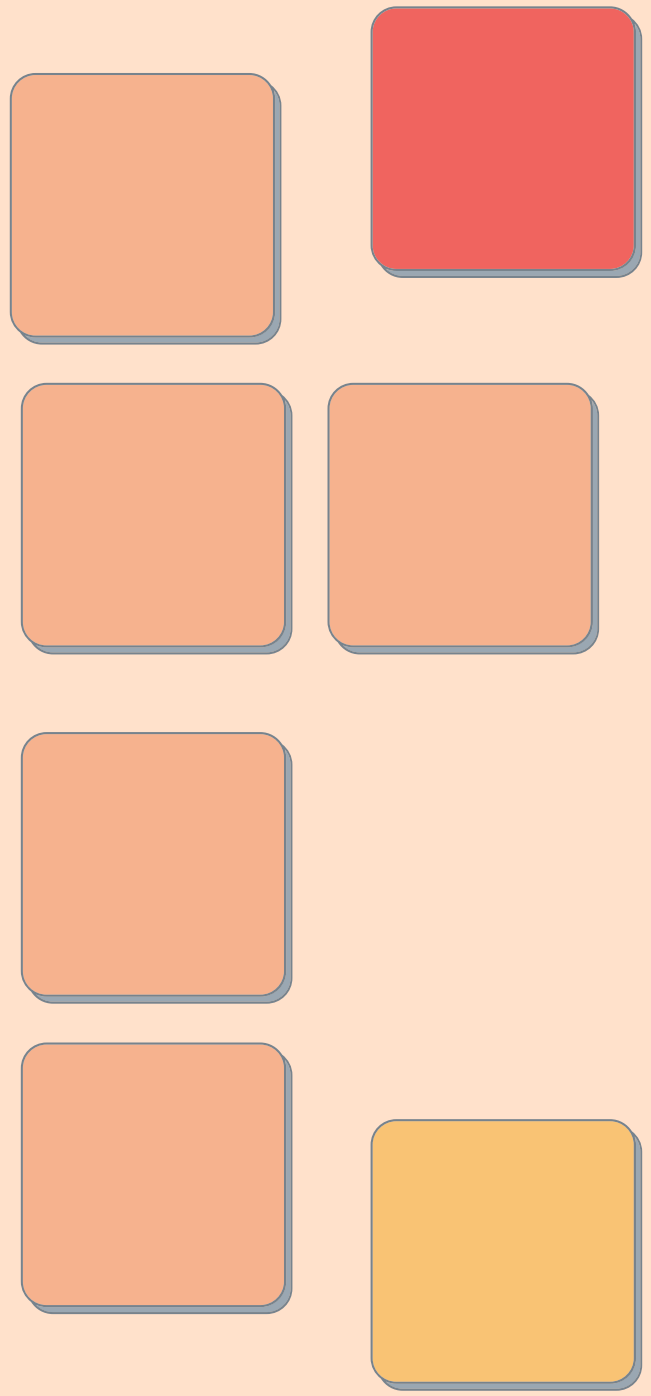
ROBOFIED

04

Hold-Out Cross Validation

Simplest kind of Cross-Validation.

Dataset is separated into training set and testing set. Then, training set is again separated into training set and validation set.



05

ROBOFIED

Code

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y,
                                                    test_size = 0.20,
                                                    random_state = 42)
```

ROBOFIED

06 Pros & Cons

Pros:

Needs only one iteration hence the computation costs are much lower.

Cons:

Provides a high variance estimate since changing which observations happen to be in the testing set can significantly change testing accuracy.

ROBOFIED

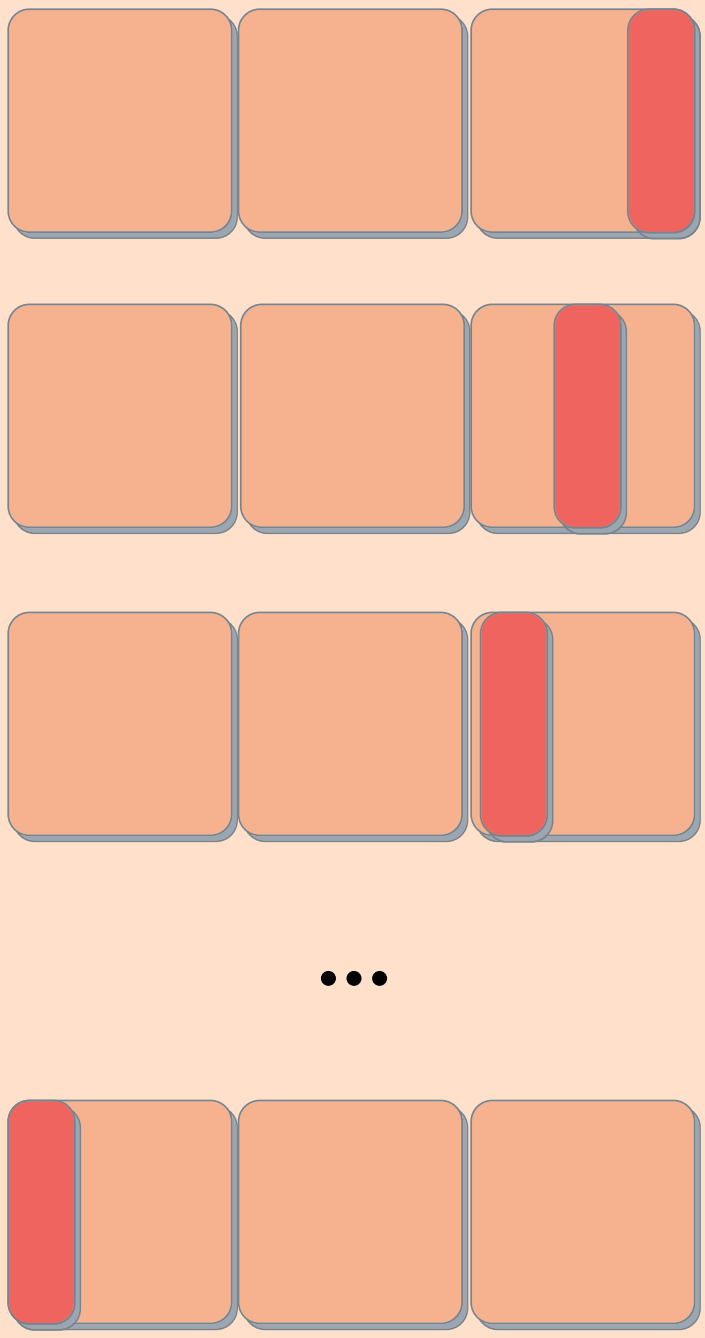
07

Leave-One-Out Cross Validation (LOOCV)

Only one data point is kept as testing set and model is trained on rest of the data.

This process is iterated for each data point.

For example: If we have 1000 data it will repeat 1000 times, at each iteration one data point for testing and the remaining for training.



ROBOFIED

08 Pros & Cons

Pros:

Covers all the data points and learns everything, hence the bias will be very low.

Cons:

Since we repeat the process n time where n is the total number of data points, it results in high execution time.

09

ROBOFIED
Code

```
from sklearn.model_selection import LeaveOneOut

loo = LeaveOneOut()
loo.get_n_splits(X)

for train_index, test_index in loo.split(X):
  X_train, X_test = X.iloc[train_index], X.iloc[test_index]
  y_train, y_test = y[train_index], y[test_index]
```

ROBOFIED

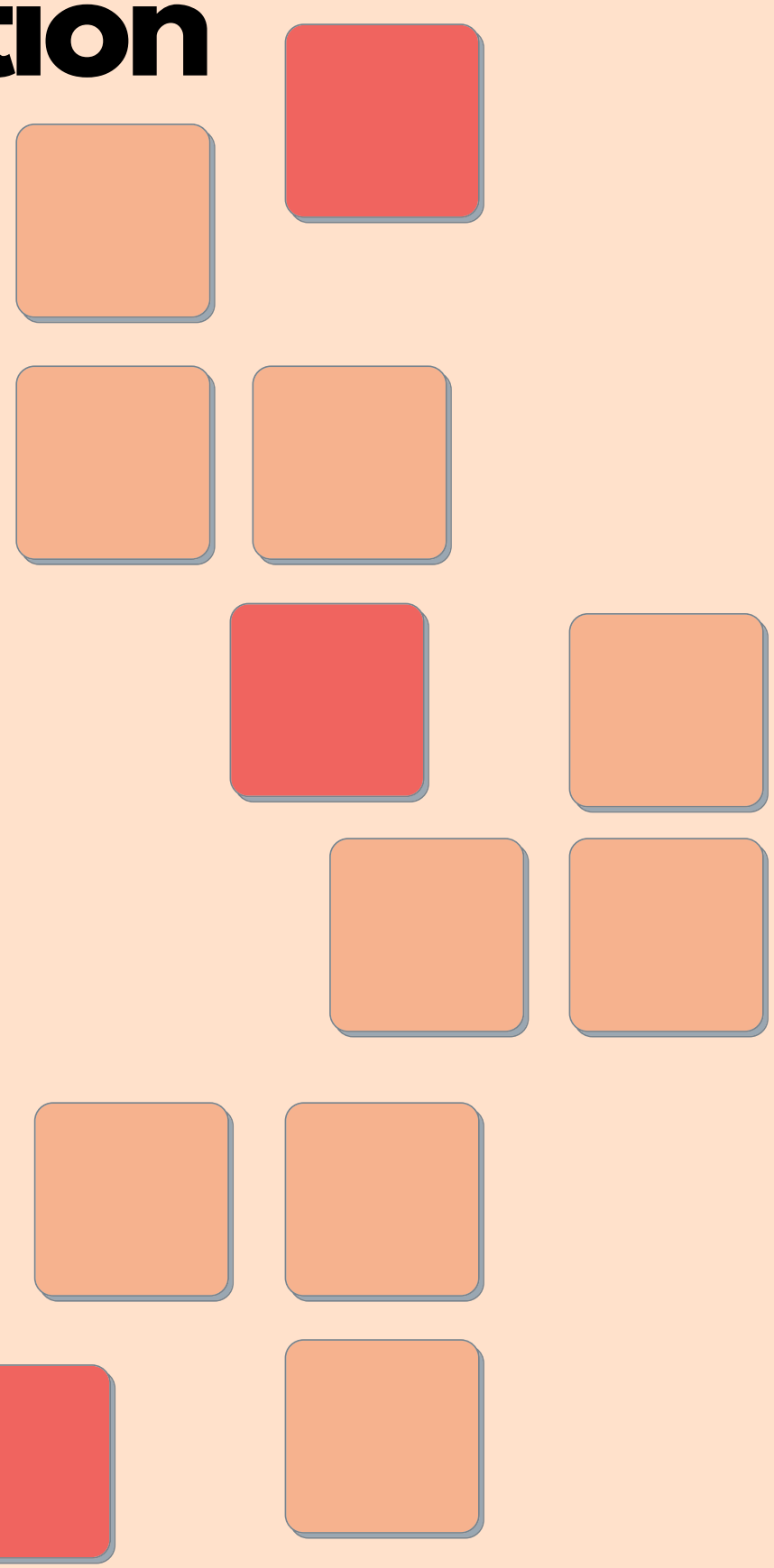
10

k-Folds Cross-Validation

Simply splitting data into train and test doesn't help due to variance it provided.

Creating a bunch of train/test splits, calculating the testing accuracy for each, and averaging the results together?

This is k-Folds Cross Validation.



ROBOFIED

11 Steps

1. Split the dataset into **k** groups randomly. Take one group as testing set and rest as training set.
2. Fit the model on the training set and evaluate it on test set.
3. Set aside this evaluation score.
4. Take another group as test set and rest as training set.
5. Repeat the train test model fit setup until each group has become a test set.

ROBOFIED

08 Pros & Cons

Pros:

Validates the performance of your model on multiple **folds** of your data.

Cons:

Doesn't work well with sequential data like time series.

12

ROBOFIED

Code

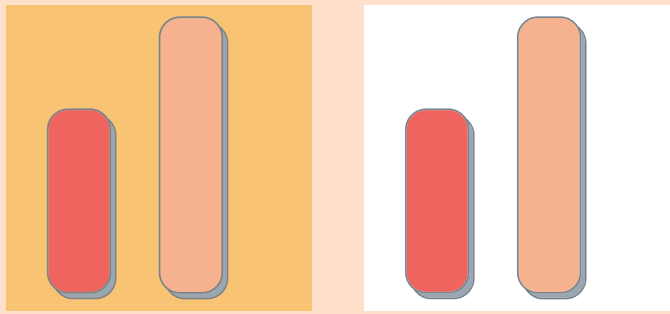
```
from sklearn.model_selection import KFold
kf = KFold(n_splits = 5,
           random_state = None,
           shuffle = False)

for train_index, test_index in kf.split(X):
  print("Train:", train_index, "Test:", test_index)
  X_train, X_test = X[train_index], X[test_index]
  y_train, y_test = y[train_index], y[test_index]
```

ROBOFIED

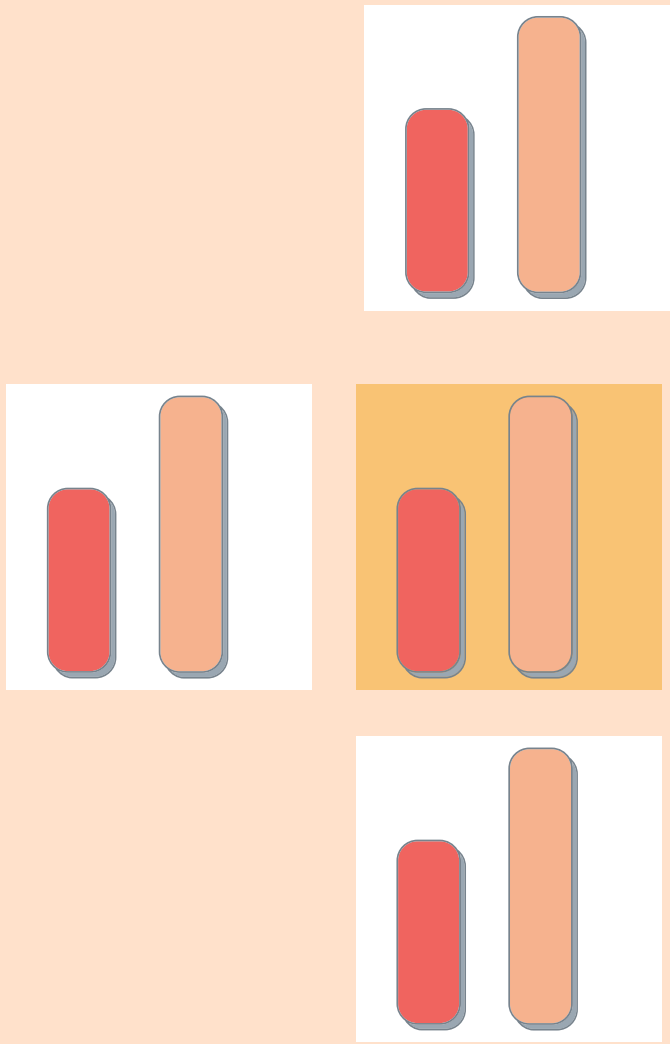
13

Stratified k-Folds Cross-Validation

Due to random sampling, there could be improper division of data between train and test sets.



For example, in case of binary classification problems, there could be more negative samples in one test set and in another test set, it could be very less.



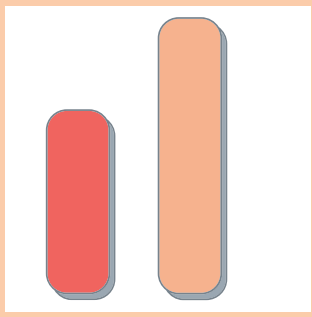
To rectify this problem, stratification is required which means to rearrange in such a way that each fold is good representative of whole.

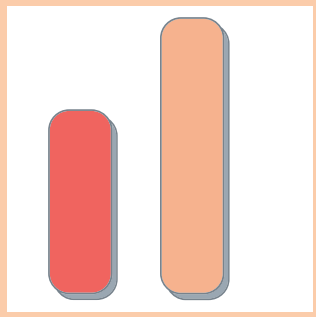
14

ROBOFIED

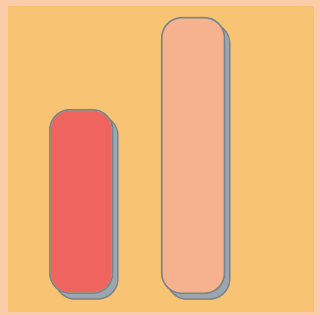
Cases

In case of categorical variables:



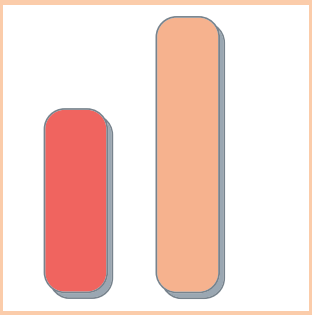


In each set, there are equal or close to results from each categories.



In case of continuous variables:

Means of all outcomes are comparable.



Class Distributions

15

ROBOFIED Code

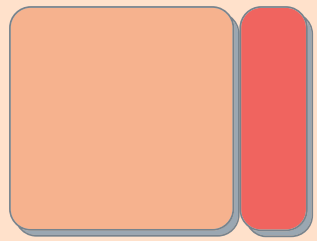
```python
from sklearn.model_selection import StratifiedKFolds

kf = StratifiedKFold(n_splits=5, random_state=None)

# X is the feature set and y is the target
for train_index, test_index in skf.split(X,y):
  print("Train:", train_index, "Test:", test_index)
  X_train, X_test = X.iloc[train_index], X.iloc[test_index]
  y_train, y_test = y[train_index], y[test_index]
```

ROBOFIED

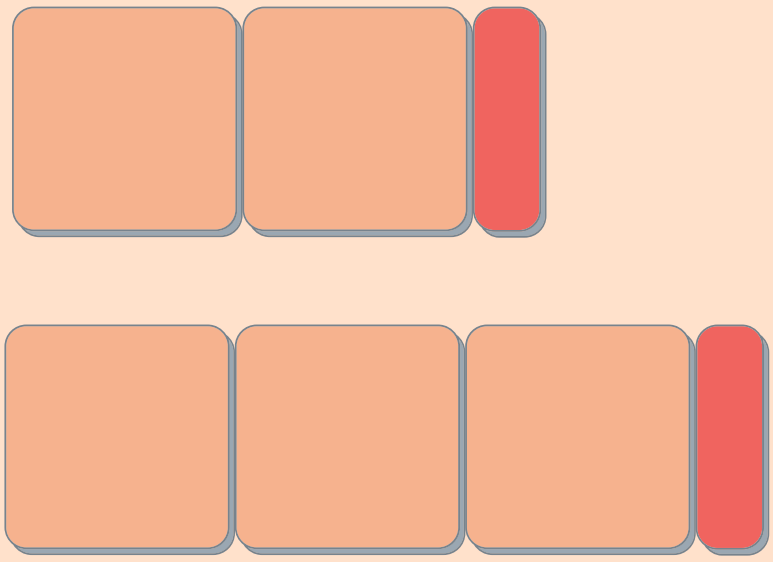
Cross-Validation for Time-Series

16

Splitting simply into train and test sets, doesn't help due to the time aspect of the data.



Folds for time-series data are created in forward chaining fashion. In each time step, train and test sets are changed. Later, we can average the forecast accuracy of test sets.



17

ROBOFIED
Code

```python
from sklearn.model_selection import TimeSeriesSplit

tscv = TimeSeriesSplit(n_splits=3)

for train_index, test_index in tscv.split(X):
  print("Train:", train_index, "Test:", val_index)
  X_train, X_test = X.iloc[train_index], X.iloc[val_index]
  y_train, y_test = y[train_index], y[val_index]
```

ROBOFIED

At **Robofied,**
we aspire to democratize **AI**
Education

Join our channels:

www.robofied.com
https://instagram.com/robofied
https://twitter.com/Robofied