# EM 538-001: PRACTICAL MACHINE LEARNING FOR ENGINEERING ANALYTICS

LECTURE 006

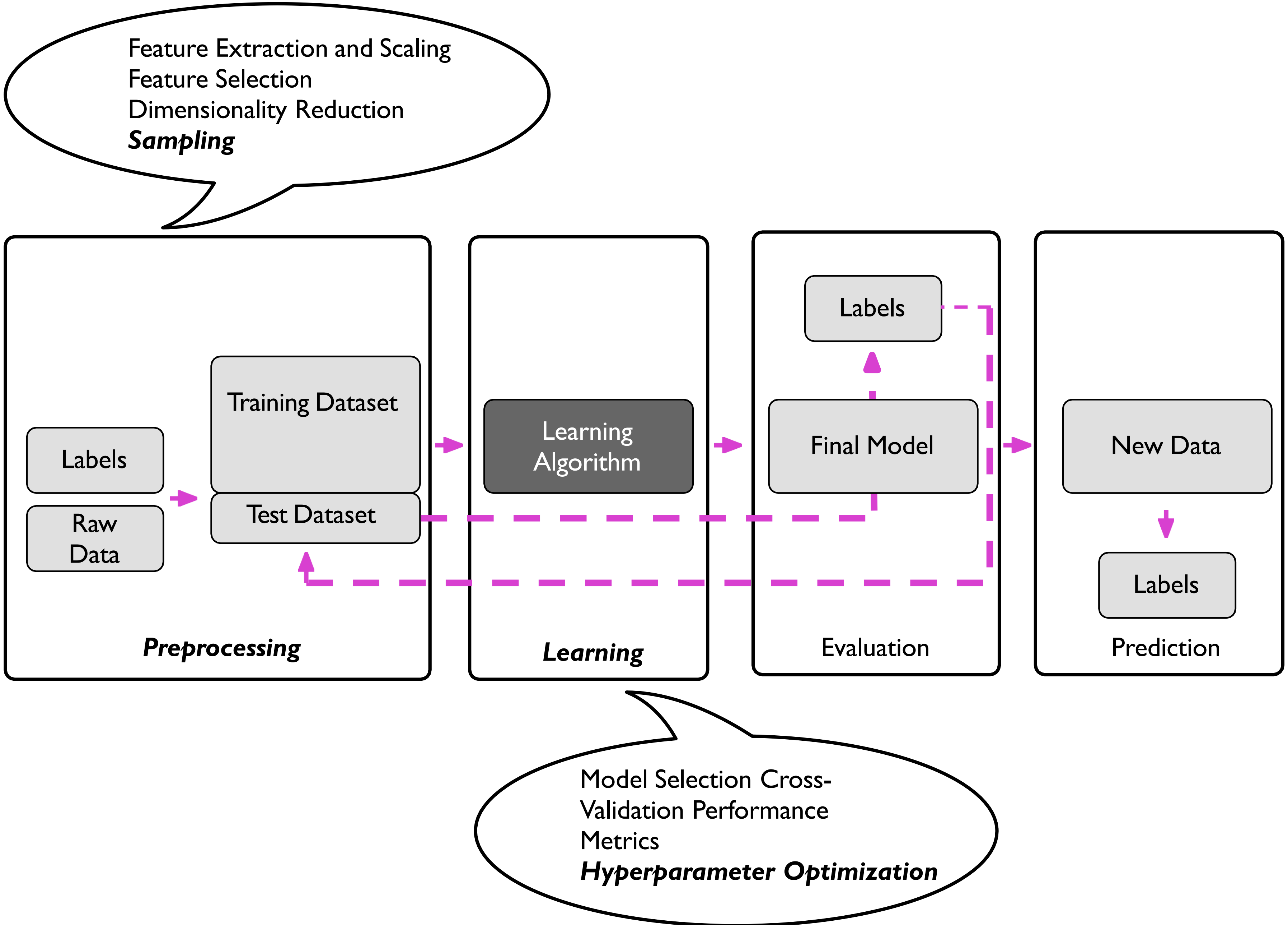
Fred Livingston, Ph.D.

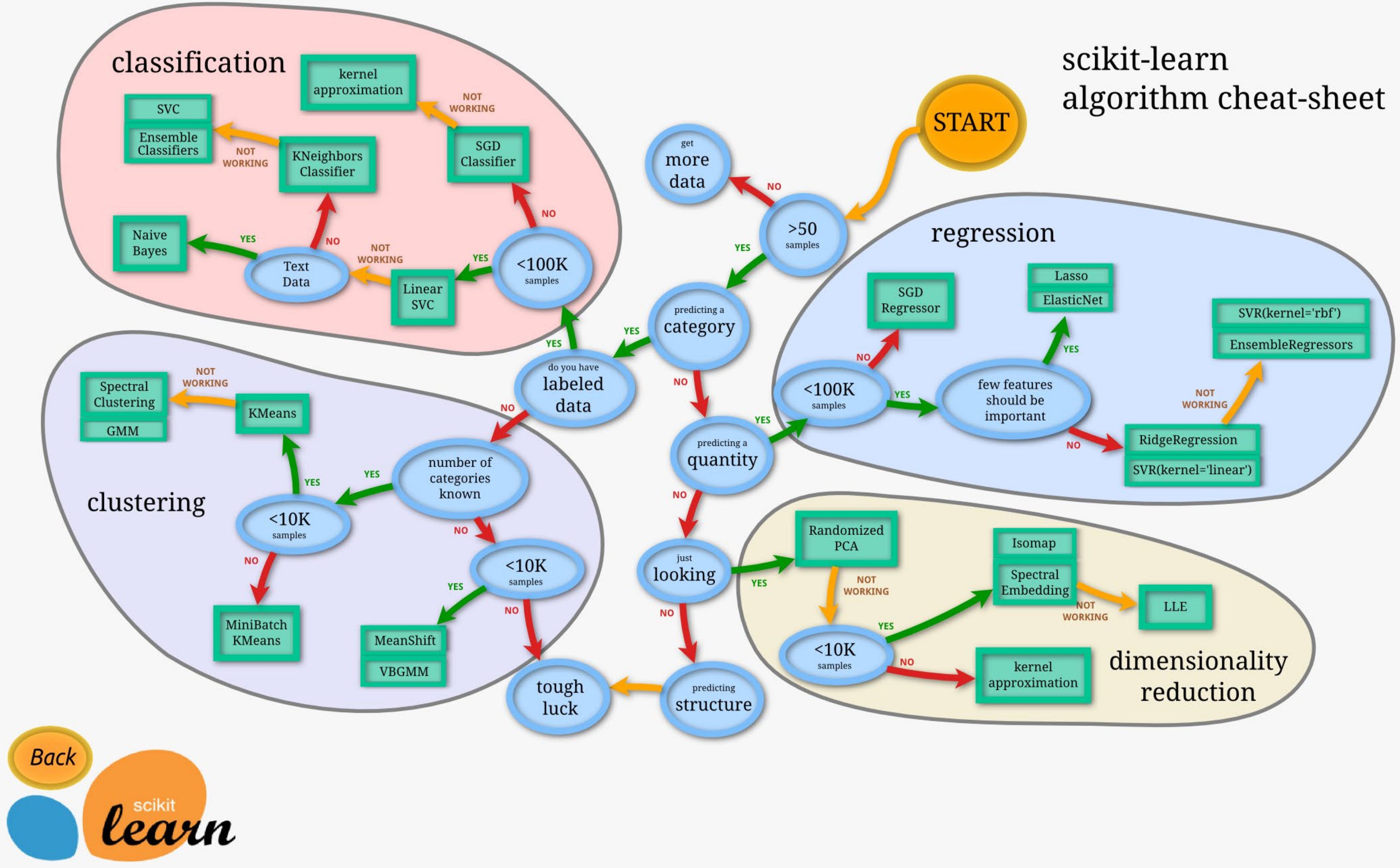
# KNN – SUPERVISED LEARNING MODEL (CONT)

- Preprocessing and Hyperparameter Tuning
  - Simple Holdout
  - 3-way Holdout
  - K-Fold Cross-Validation

- Model Performance
- Homework 1

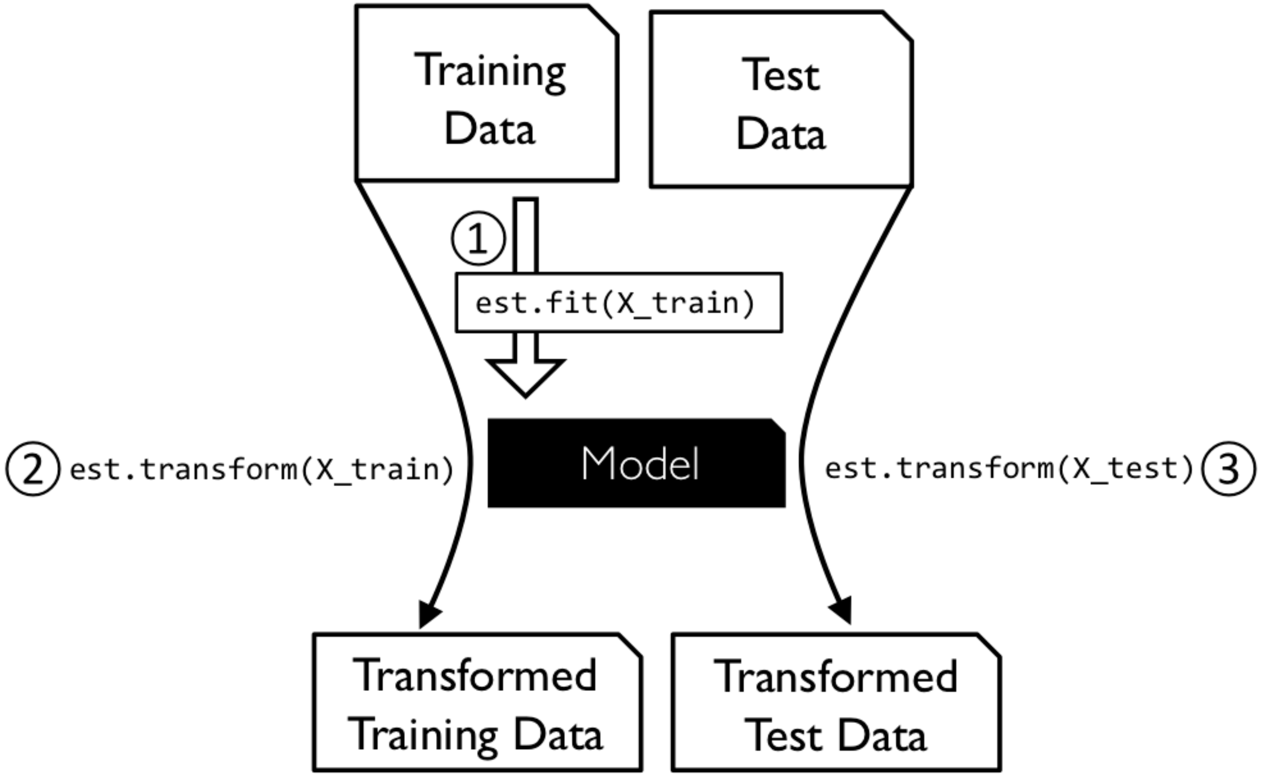
# MACHINE LEARNING WORKFLOW





scikit-learn
algorithm cheat-sheet
START
>50 samples
NO
get more data
YES
predicting a category
YES
do you have labeled data
YES
classification
<100K samples
NO
SGD Classifier
NOT WORKING
kernel approximation
YES
Linear SVC
NOT WORKING
Text Data
YES
Naive Bayes
NO
KNeighbors Classifier
NOT WORKING
SVC
Ensemble Classifiers
NO
clustering
number of categories known
YES
<10K samples
YES
KMeans
NOT WORKING
Spectral Clustering
GMM
NO
MiniBatch KMeans
NO
<10K samples
YES
MeanShift
VBGMM
NO
tough luck
NO
predicting a quantity
YES
regression
<100K samples
NO
SGD Regressor
YES
few features should be important
YES
Lasso
ElasticNet
NO
RidgeRegression
SVR(kernel='linear')
NOT WORKING
SVR(kernel='rbf')
EnsembleRegressors
NO
just looking
YES
dimensionality reduction
Randomized PCA
NOT WORKING
<10K samples
YES
Isomap
Spectral Embedding
NOT WORKING
LLE
NO
kernel approximation
NO
predicting structure
tough luck
Back
scikit learn

# PREPROCESSING: SAMPLING WITH SIMPLE HOLDOUT



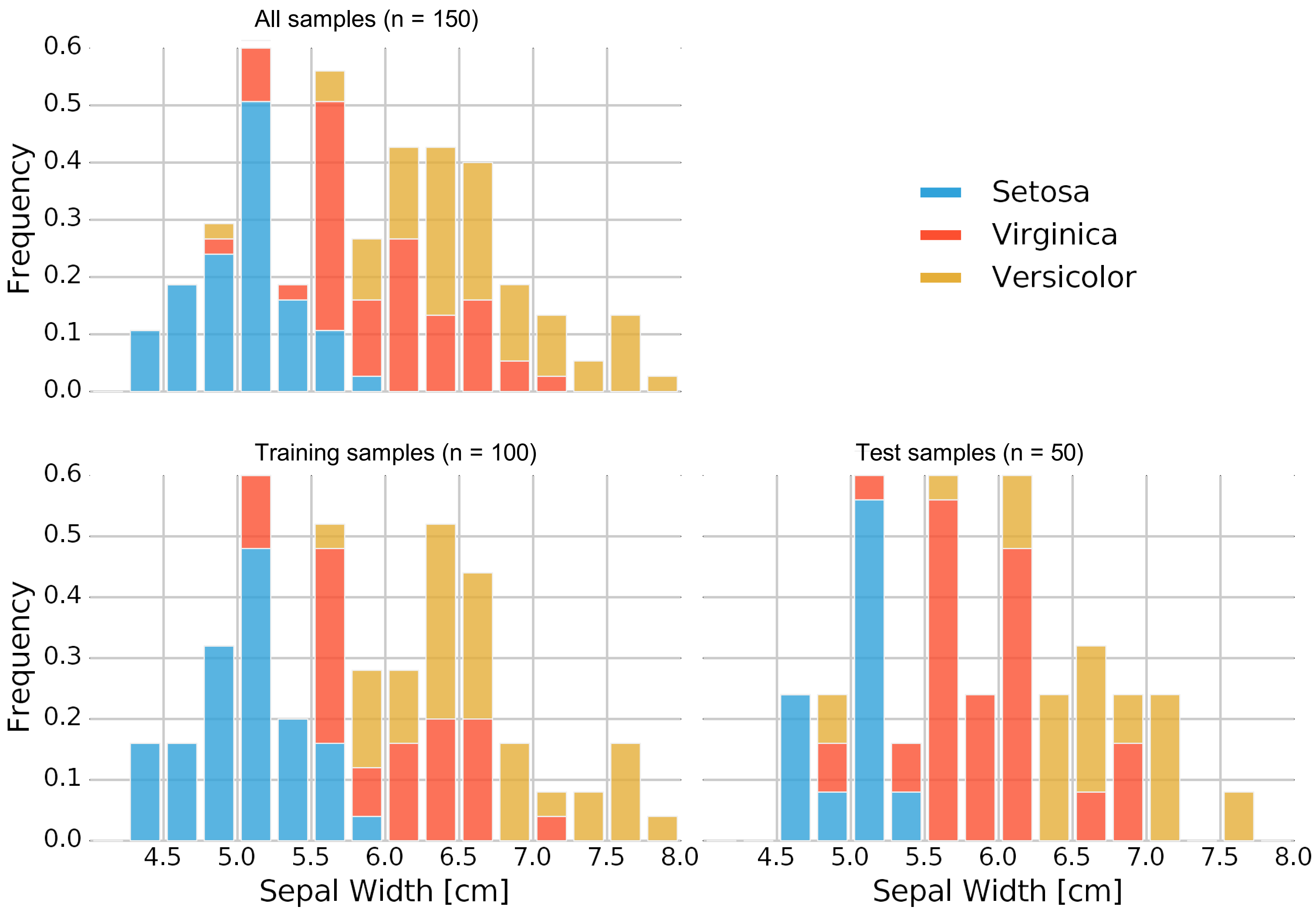
# HOLDOUT USING SKLEARN

## Simple Holdout Method

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                    test_size=0.33,
                                                    random_state=123,
                                                    shuffle=True, stratify=y)
```
]

# RANDOM SUBSAMPLING …



# SUPREVISED LEARNING: K-NEAREST NEIGHBOR

https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html

> API Reference > sklearn.neighbors > KNeighborsClassifier

## KNeighborsClassifier

*class* **sklearn.neighbors.KNeighborsClassifier**(*n_neighbors=5, *, weights='uniform', algorithm='auto', leaf_size=30, p=2, metric='minkowski', metric_params=None, n_jobs=None*) [source]

## KNeighborsRegressor

*class* **sklearn.neighbors.KNeighborsRegressor**(*n_neighbors=5, *, weights='uniform', algorithm='auto', leaf_size=30, p=2, metric='minkowski', metric_params=None, n_jobs=None*) [source]

Regression based on k-nearest neighbors.

The target is predicted by local interpolation of the targets associated of the nearest neighbors in the training set.

# LEARNING: ***HYPERPARAMETER OPTIMIZATION***

# HYPERPARAMETERS

- Value of *k*
- Scaling of the feature axes
- Distance measure
- Weighting of the distance measure

0-transforming_dataset_2holdout.ipynb

G: > My Drive > Academics > Machine Learning > Spring 2025 > Lectures > Lecture 006 > lecture6_code_examples > 0-transforming_dataset_2holdout.ipynb > ...

Generate + Code + Markdown | Run All Restart Clear All Outputs | View data Jupyter Variables Outline ... pyml (Python 3.11.9)

EM 538-001: Practical Machine Learning for Enginering Analystics (Spring 2025)
Instructor: Fred Livingston (fjliving@ncsu.edu)

## Install scikit-learn library

```python
# !pip install scikit-learn scipy
```

[ ] Python

# Import Data

```python
import pandas as pd

df_iris = pd.read_csv('iris.csv')
df_iris.head()
```

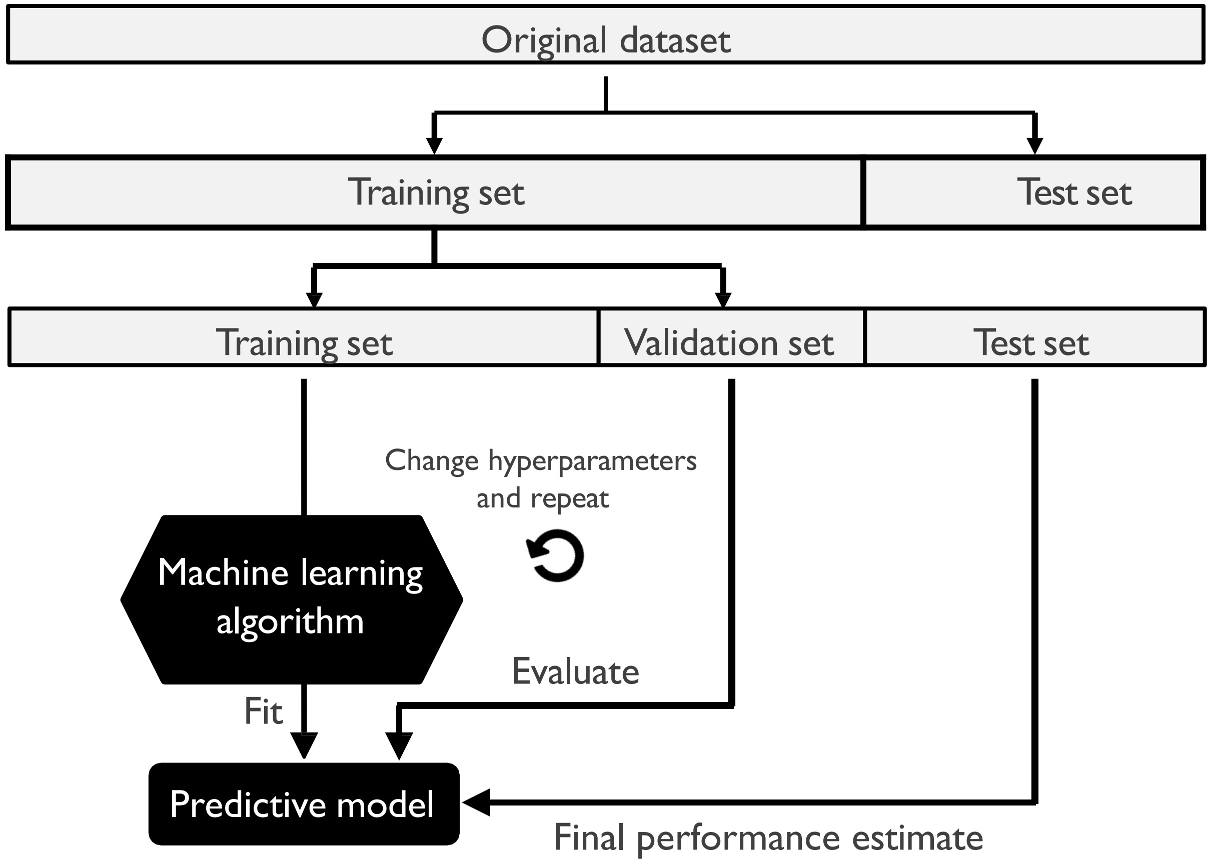
[ ] Open 'df_iris' in Data Wrangler Python

# Preprocessing

```python
df_iris_simple = df_iris.drop(['Id', 'SepalLength[cm]', 'SepalWidth[cm]'] , axis=1)
df_iris_simple.head()
```

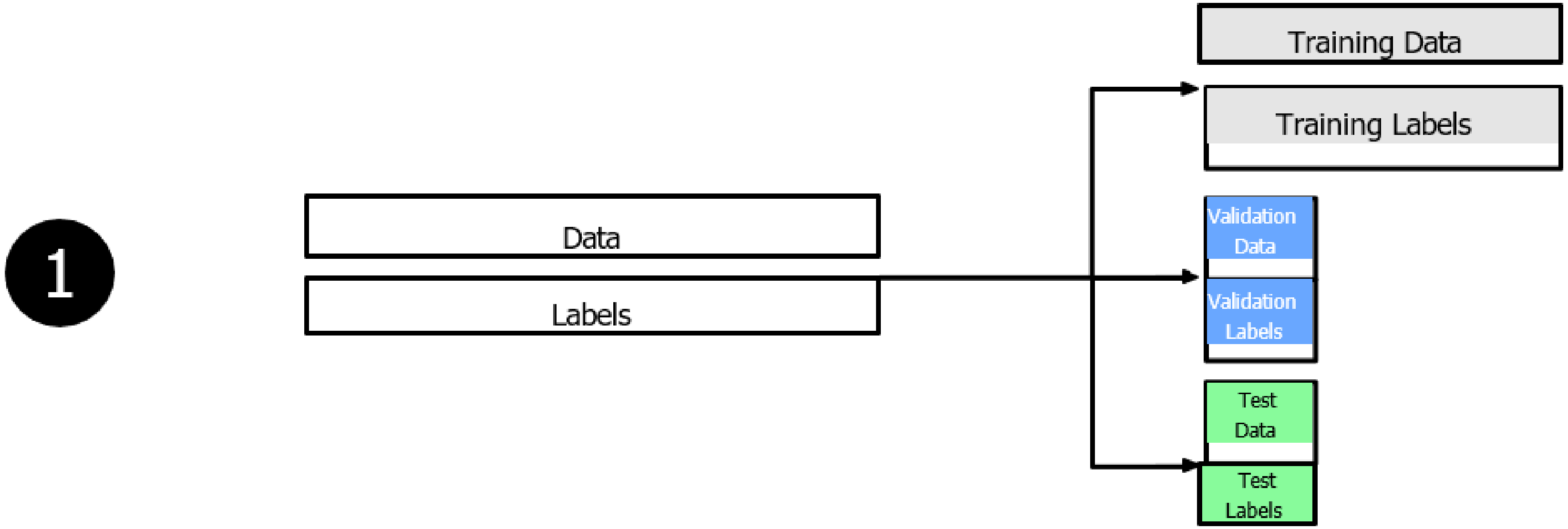
[ ] Open 'df_iris_simple' in Data Wrangler Python

# 3-WAY HOLDOUT METHOD



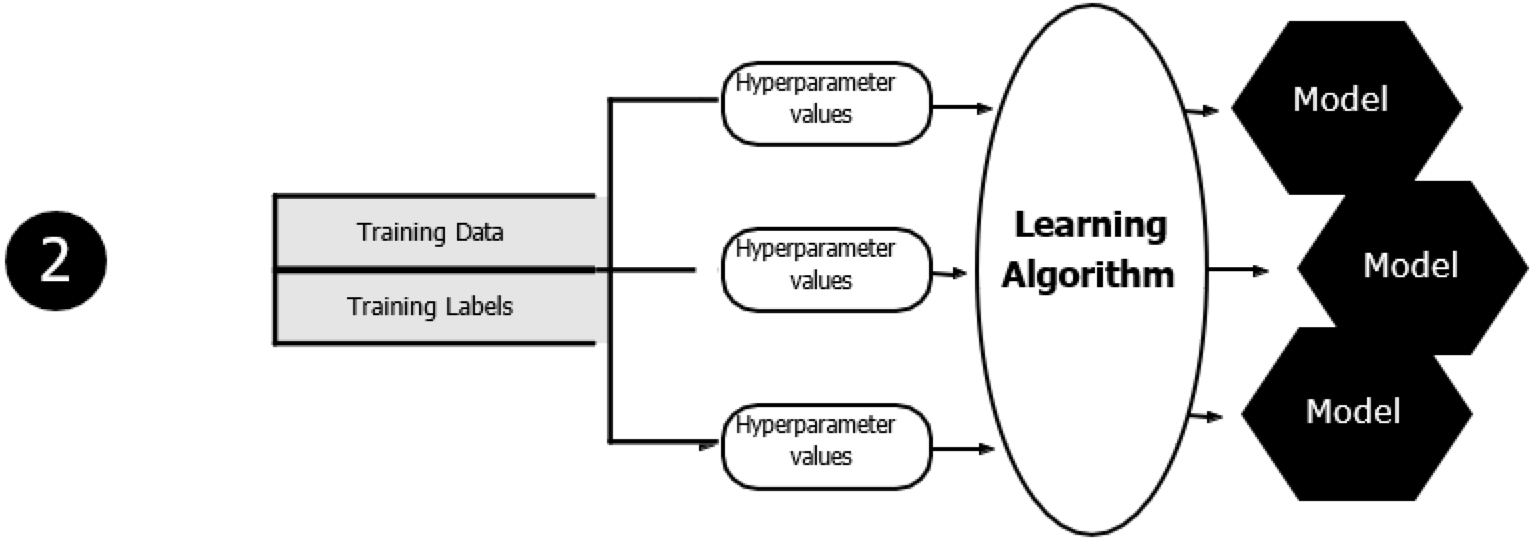
# 3-WAY HOLDOUT

instead of "regular" holdout to avoid "***data leakage***" during hyperparameter optimization



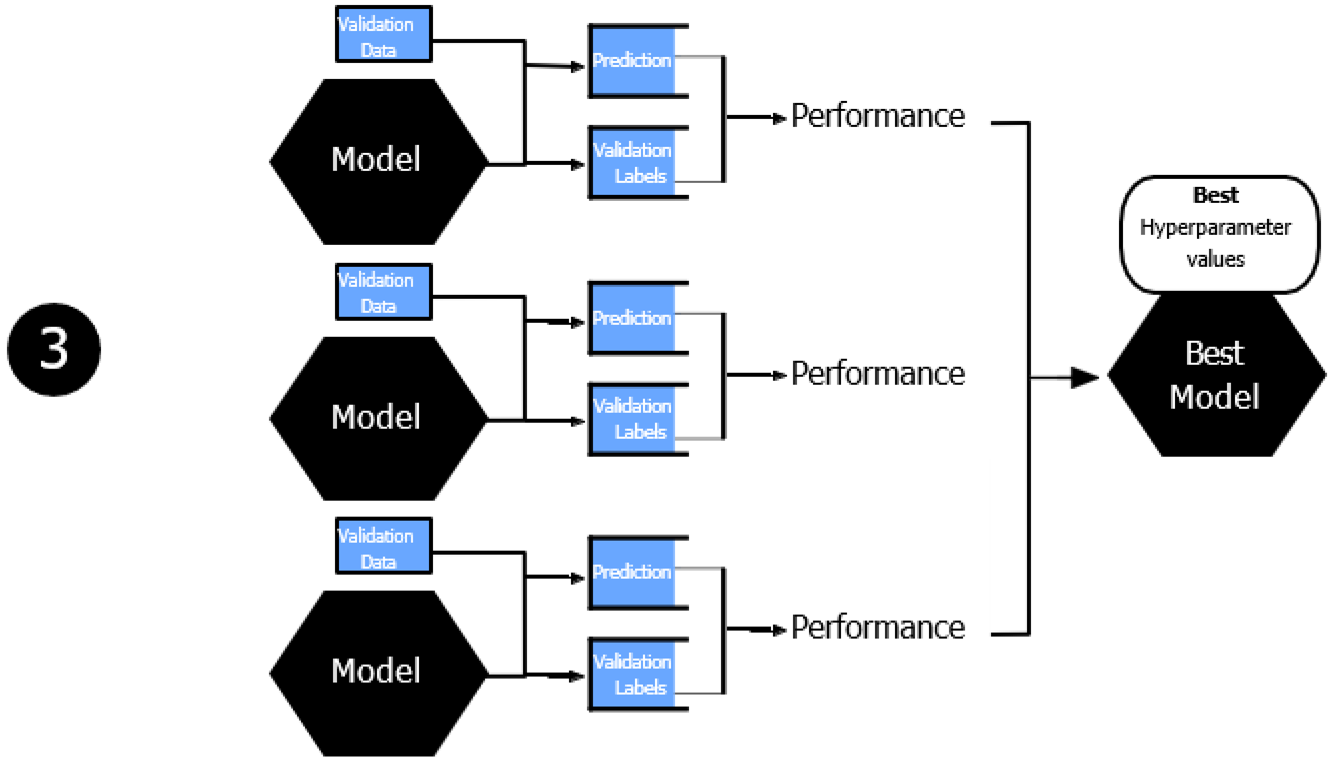
# 3-WAY HOLDOUT

instead of "regular" holdout to avoid "***data leakage***" during hyperparameter optimization



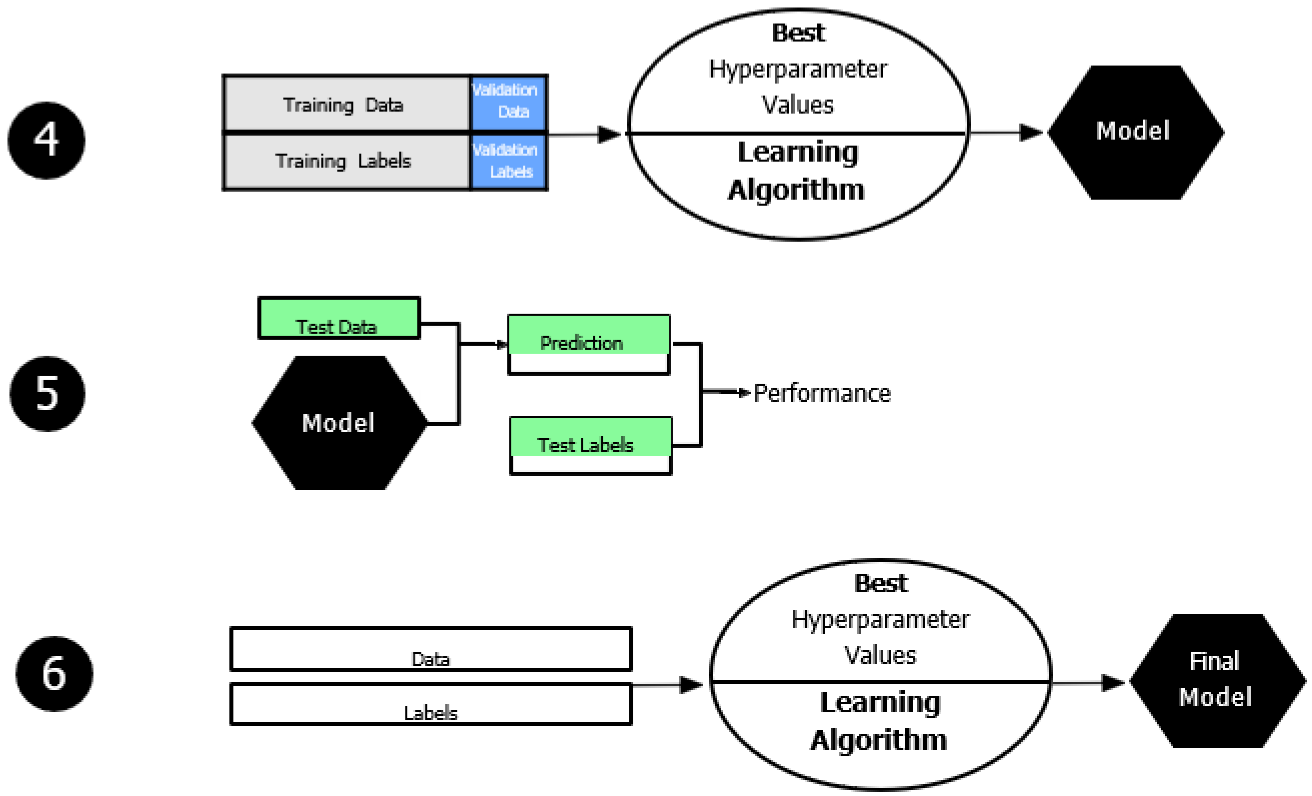
# 3-WAY HOLDOUT

instead of "regular" holdout to avoid "data leakage" during hyperparameter optimization



# 3-WAY HOLDOUT

instead of "regular" holdout to avoid "data leakage" during hyperparameter optimization



1-transforming_dataset_3holdout.ipynb ×

G: > My Drive > Academics > Machine Learning > Spring 2025 > Lectures > Lecture 006 > lecture6_code_examples > 1-transforming_dataset_3holdout.ipynb > K - Hyperparamater > from sklearn.neighbors i

Generate + Code + Markdown | Run All Restart Clear All Outputs | View data Jupyter Variables Outline ··· pyml (Python 3.11.9)

EM 538-001: Practical Machine Learning for Enginering Analytics (Spring 2025)
Instructor: Fred Livingston (fjliving@ncsu.edu)

# Install scikit-learn library

```
# !pip install scikit-learn scipy
```

[55] ✓ 0.0s Python

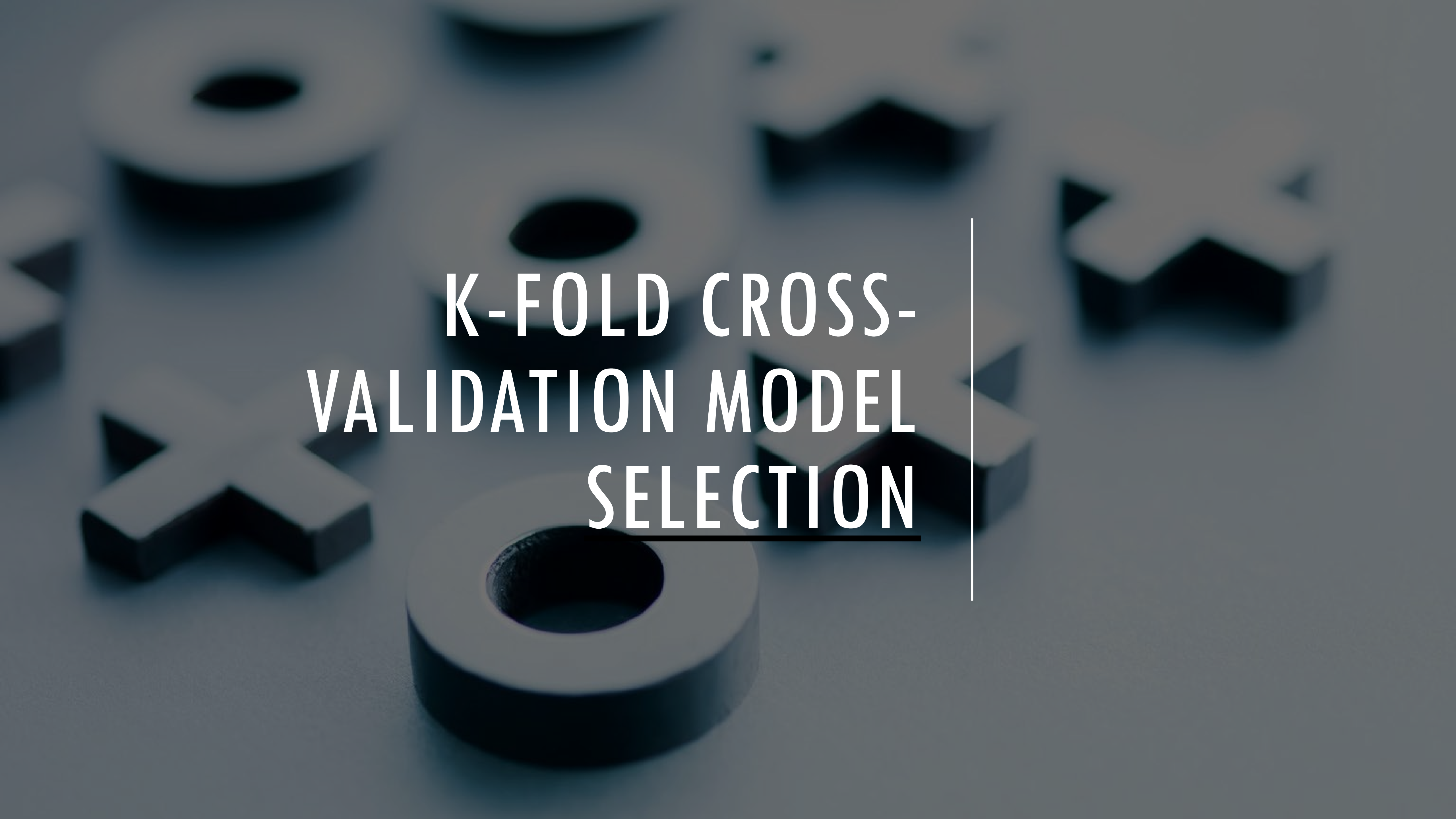
# Import Data

```
import pandas as pd

df_iris = pd.read_csv('iris.csv')
df_iris.head()
```

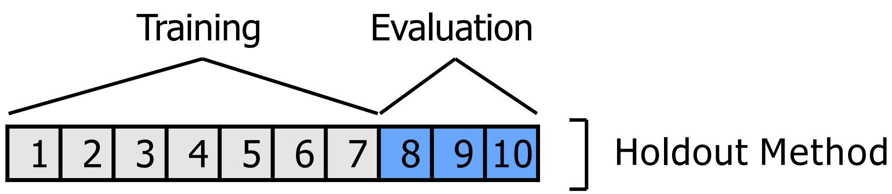
[56] ✓ 0.0s Open 'df_iris' in Data Wrangler Python

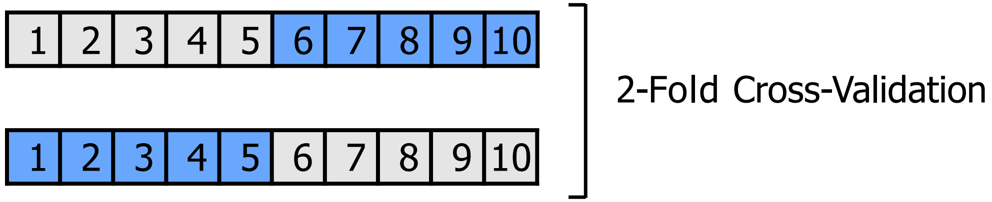
| | Id | SepalLength[cm] | SepalWidth[cm] | PetalLength[cm] | PetalWidth[cm] | Species |
|---|---|---|---|---|---|---|
| 0 | 1 | 5.1 | 3.5 | 1.4 | 0.2 | Iris-setosa |
| 1 | 2 | 4.9 | 3.0 | 1.4 | 0.2 | Iris-setosa |
| 2 | 3 | 4.7 | 3.2 | 1.3 | 0.2 | Iris-setosa |
| 3 | 4 | 4.6 | 3.1 | 1.5 | 0.2 | Iris-setosa |
| 4 | 5 | 5.0 | 3.6 | 1.4 | 0.2 | Iris-setosa |



# K-FOLD CROSS-VALIDATION MODEL SELECTION

# K-FOLD CROSS VALIDATION




Note that k-fold cross-validation is to evaluate the model design, not a particular training. Because you re-trained the model of the same design with different training sets.

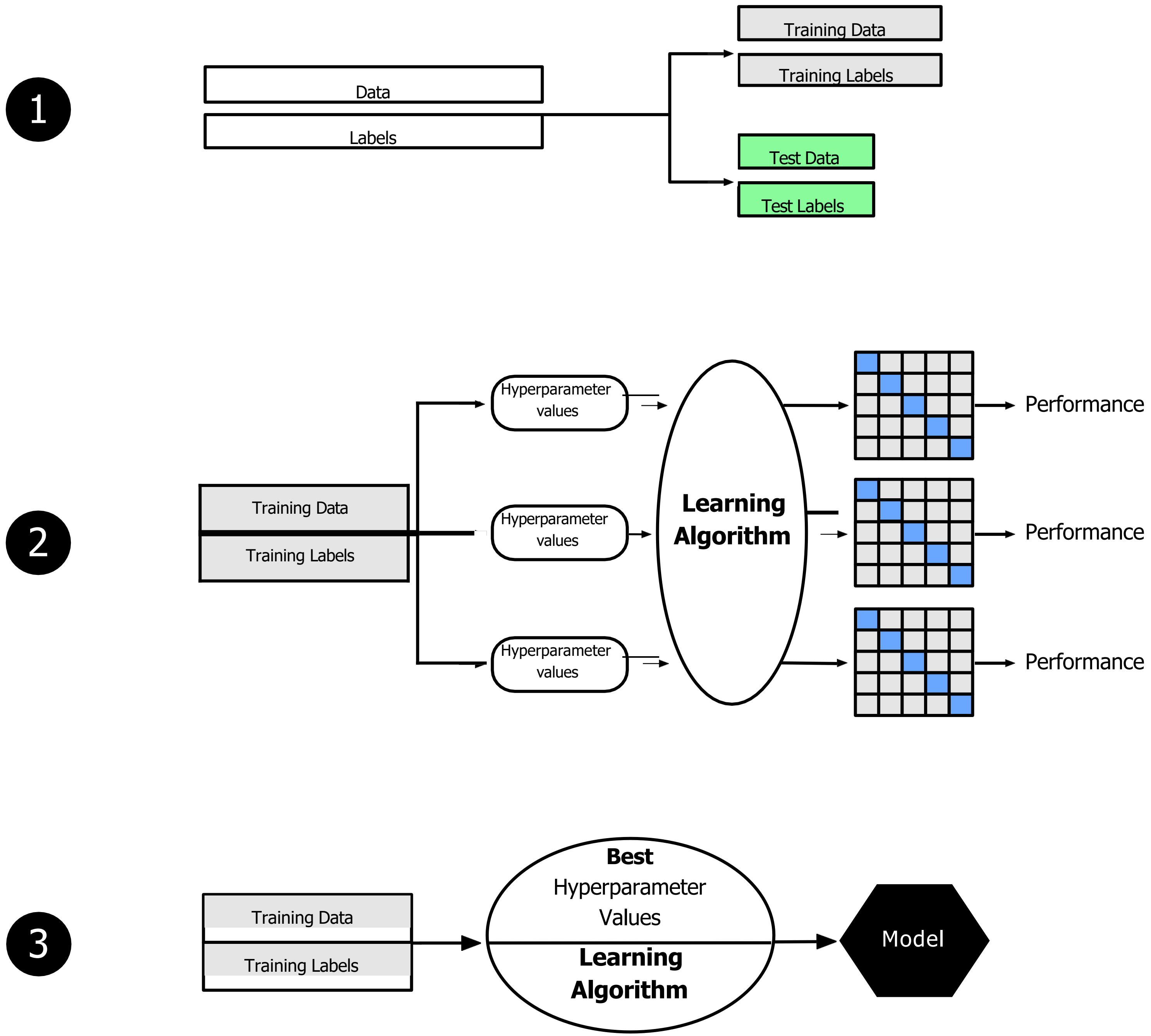
The general procedure is as follows:

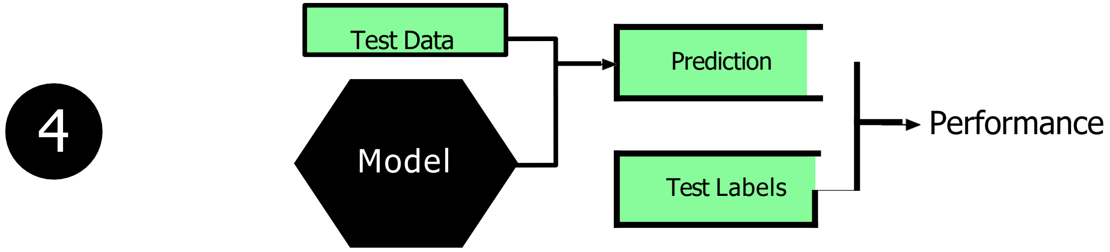
1. Shuffle the dataset randomly.
2. Split the dataset into k groups
3. For each unique group:
    - Take the group as a hold out or test data set
    - Take the remaining groups as a training data set
    - Fit a model on the training set and evaluate it on the test set
    - Retain the evaluation score and discard the model
4. Summarize the skill of the model using the sample of model evaluation scores

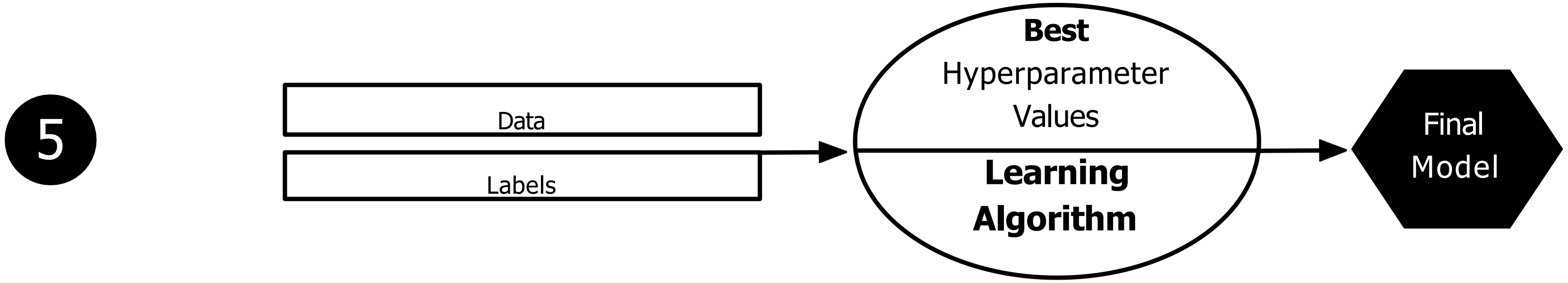
Imagine we have a data sample with 6 observations:

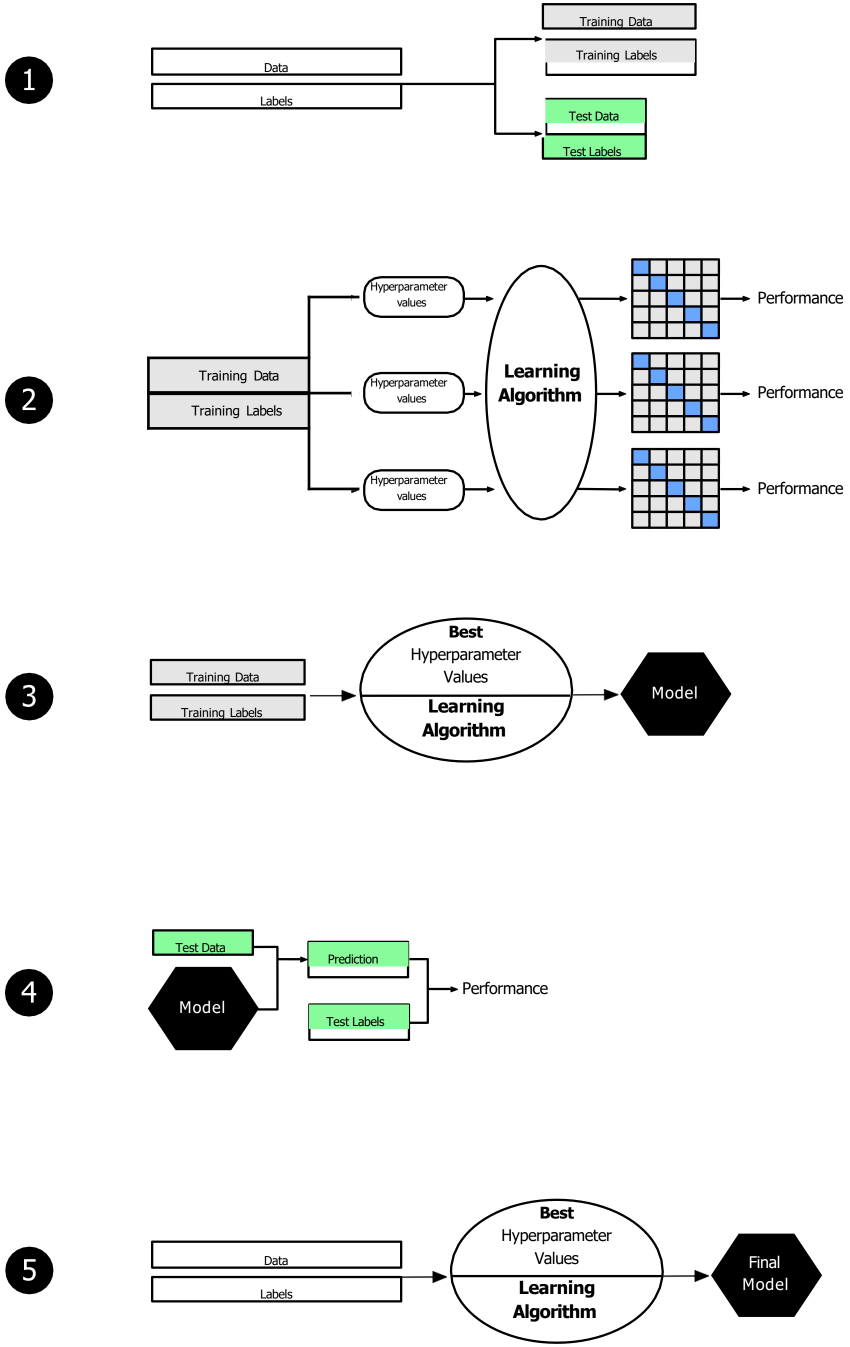
[0.1, 0.2, 0.3, 0.4, 0.5, 0.6]

The first step is to pick a value for k in order to determine the number of folds used to split the data. Here, we will use a value of k=3. That means we will shuffle the data and then split the data into 3 groups. Because we have 6 observations, each group will have an equal number of 2 observations.


1
Data
Labels
Training Data
Training Labels
Test Data
Test Labels
2
Training Data
Training Labels
Hyperparameter values
Hyperparameter values
Hyperparameter values
Learning Algorithm
Performance
Performance
Performance
3
Training Data
Training Labels
Best Hyperparameter Values
Learning Algorithm
Model


4
Test Data
Model
Prediction
Test Labels
Performance

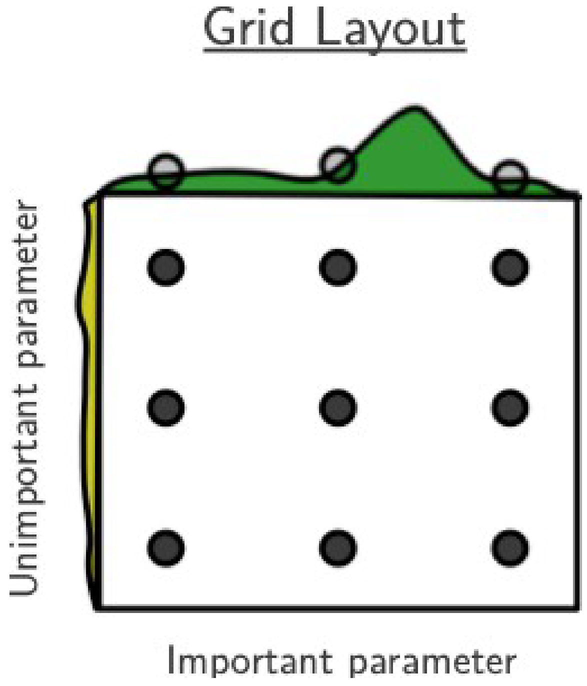

5
Data
Labels
Best
Hyperparameter
Values
Learning
Algorithm
Final
Model


1
Data
Labels
Training Data
Training Labels
Test Data
Test Labels
2
Training Data
Training Labels
Hyperparameter values
Hyperparameter values
Hyperparameter values
Learning Algorithm
Performance
Performance
Performance
3
Training Data
Training Labels
Best Hyperparameter Values
Learning Algorithm
Model
4
Test Data
Model
Prediction
Test Labels
Performance
5
Data
Labels
Best Hyperparameter Values
Learning Algorithm
Final Model

# GRID SEARCH

GridSearchCV

- Exhaustive search
- Thorough but expensive
- Specify grid for parameter search
- Can be run in parallel
- Can suffer from poor coverage
- Often run with multiple resolutions




Bergstra, J., & Bengio, Y. (2012). Random search for hyper-parameter optimization. *The Journal of Machine Learning Research*, *13*(1), 281-305.

https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html

# RANDOMIZED SEARCH

RandomizedSearchCV

- Search based on a time budget
- Preferred if there are many hyperparameters (e.g. > 3 distinct ones)
- specify distribution for parameter search
- can be run in parallel

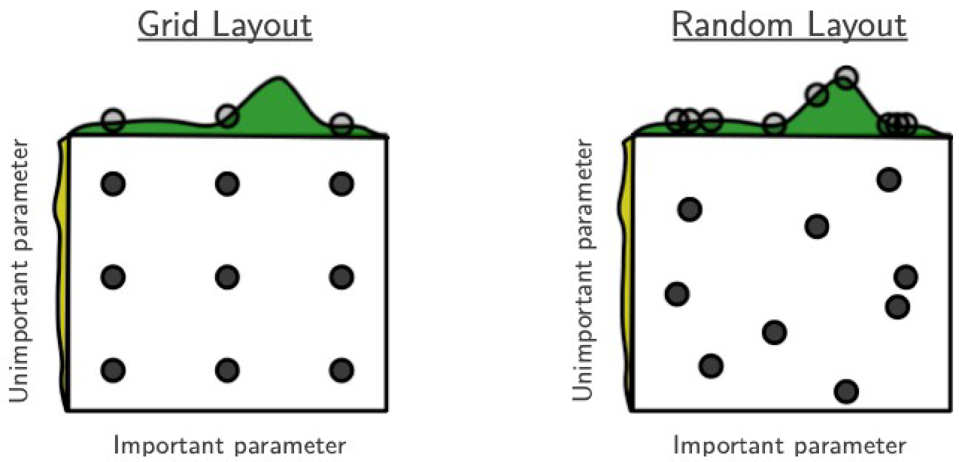



Figure 1: Grid and random search of nine trials for optimizing a function $f(x,y) = g(x) + h(y) \approx g(x)$ with low effective dimensionality. Above each square $g(x)$ is shown in green, and left of each square $h(y)$ is shown in yellow. With grid search, nine trials only test $g(x)$ in three distinct places. With random search, all nine trials explore distinct values of $g$. This failure of grid search is the rule rather than the exception in high dimensional hyper-parameter optimization.

Bergstra, J., & Bengio, Y. (2012). Random search for hyper-parameter optimization. *The Journal of Machine Learning Research*, *13*(1), 281-305.

EM 538-001: Practical Machine Learning for Enginering Analystics (Spring 2025)
Instructor: Fred Livingston (fjliving@ncsu.edu)

# Load and Prepare Datasets

```python
from sklearn.model_selection import train_test_split
import pandas as pd

df_iris = pd.read_csv('iris.csv')

X = df_iris[['PetalLength[cm]', 'PetalWidth[cm]']]
y = df_iris['Species']

X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                    test_size=0.33,
                                                    random_state=123,
                                                    shuffle=True, stratify=y)
```

[ ] Python

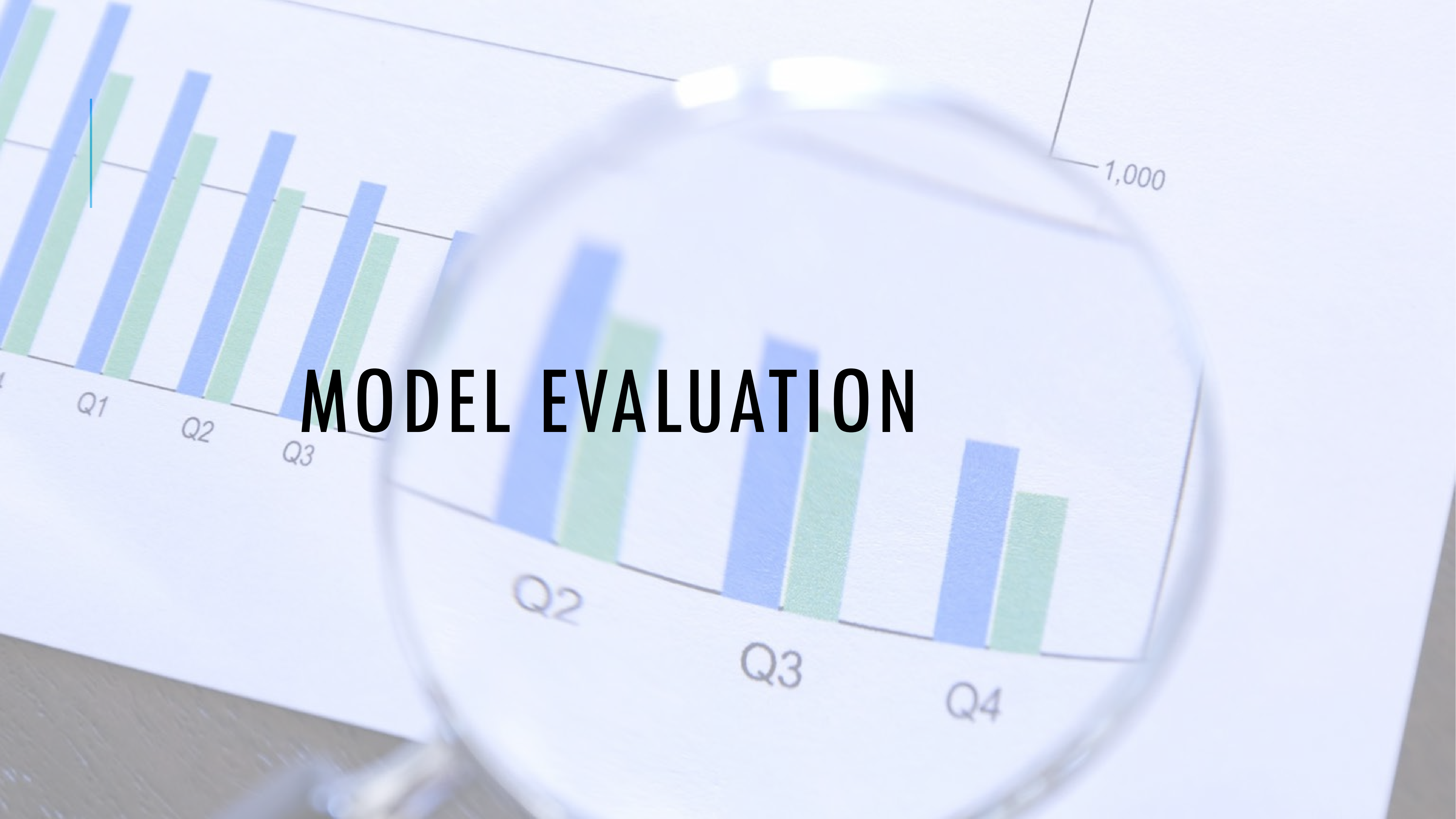
```python
X_train.shape
```

[ ] Python

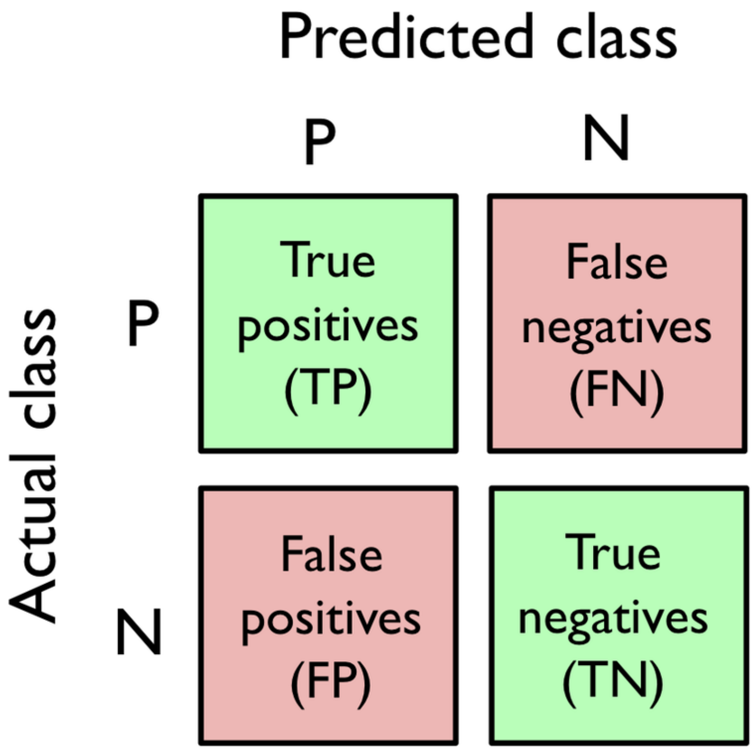
```python
X_test.shape
```

[ ] Python

# MODEL EVALUATION



# 2X2 CONFUSION




$$ERR = \frac{FP + FN}{FP + FN + TP + TN} = 1 - ACC$$

$$ACC = \frac{TP + TN}{FP + FN + TP + TN} = 1 - ERR$$

# COMPUTE CONFUSION MATRIX

```python
pipe.fit(X_train, y_train)
pipe.predict(X_test)
```

[3] ✓ 0.0s Python

```
array([1, 0, 2, 2, 0, 0, 2, 2, 2, 0, 0, 2, 2, 1, 2, 1, 0, 0, 0, 0, 0, 2,
       2, 1, 2, 2, 1, 1, 1, 1])
```

```python
y_test
```

[5] ✓ 0.0s Python

```
array([1, 0, 2, 2, 0, 0, 2, 1, 2, 0, 0, 1, 2, 1, 2, 1, 0, 0, 0, 0, 0, 2,
       2, 1, 2, 2, 1, 1, 1, 1])
```

Predicted class

| Actual class | | P | N |
|---|---|---|---|
| | P | True positives (TP) | False negatives (FN) |
| | N | False positives (FP) | True negatives (TN) |

$$ERR = \frac{FP + FN}{FP + FN + TP + TN} = 1 - ACC$$

$$ACC = \frac{TP + TN}{FP + FN + TP + TN} = 1 - ERR$$

# False Positive Rate and False Negative Rate

$$\mathrm{TPR}^{*} = \frac{\mathrm{TP}}{\mathrm{P}} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} = 1 - \mathrm{FNR}$$

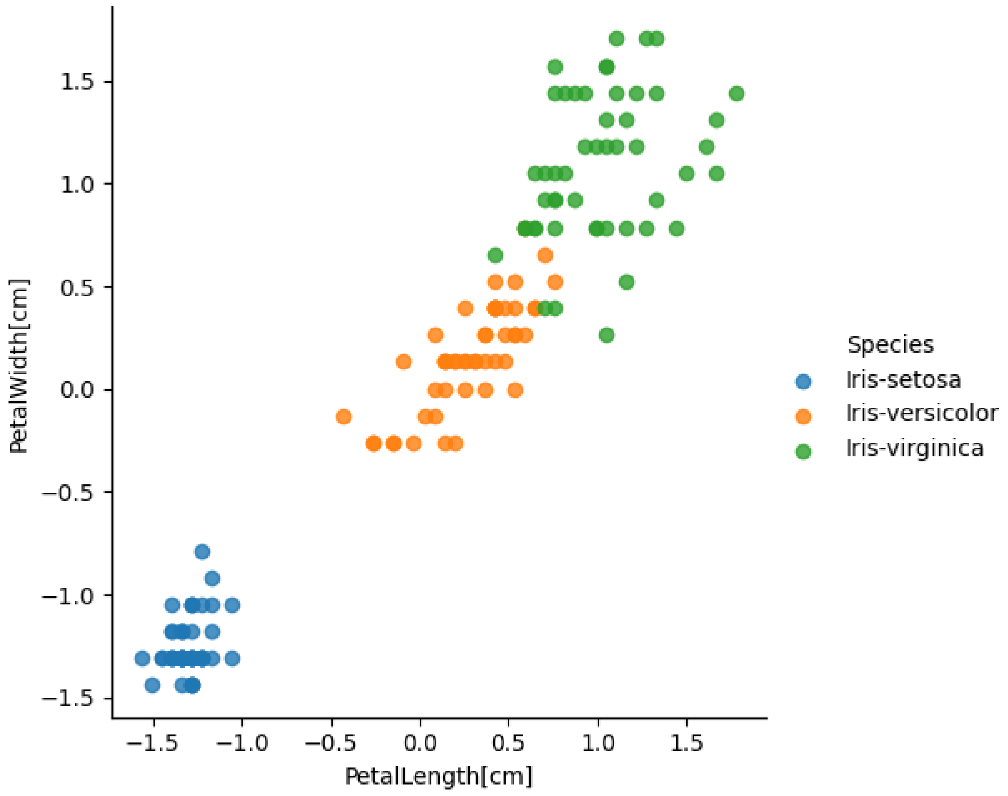
* Relevant later for ROC

$$\mathrm{FPR}^{*} = \frac{\mathrm{FP}}{\mathrm{N}} = \frac{\mathrm{FP}}{\mathrm{FP} + \mathrm{TN}} = 1 - \mathrm{TNR}$$

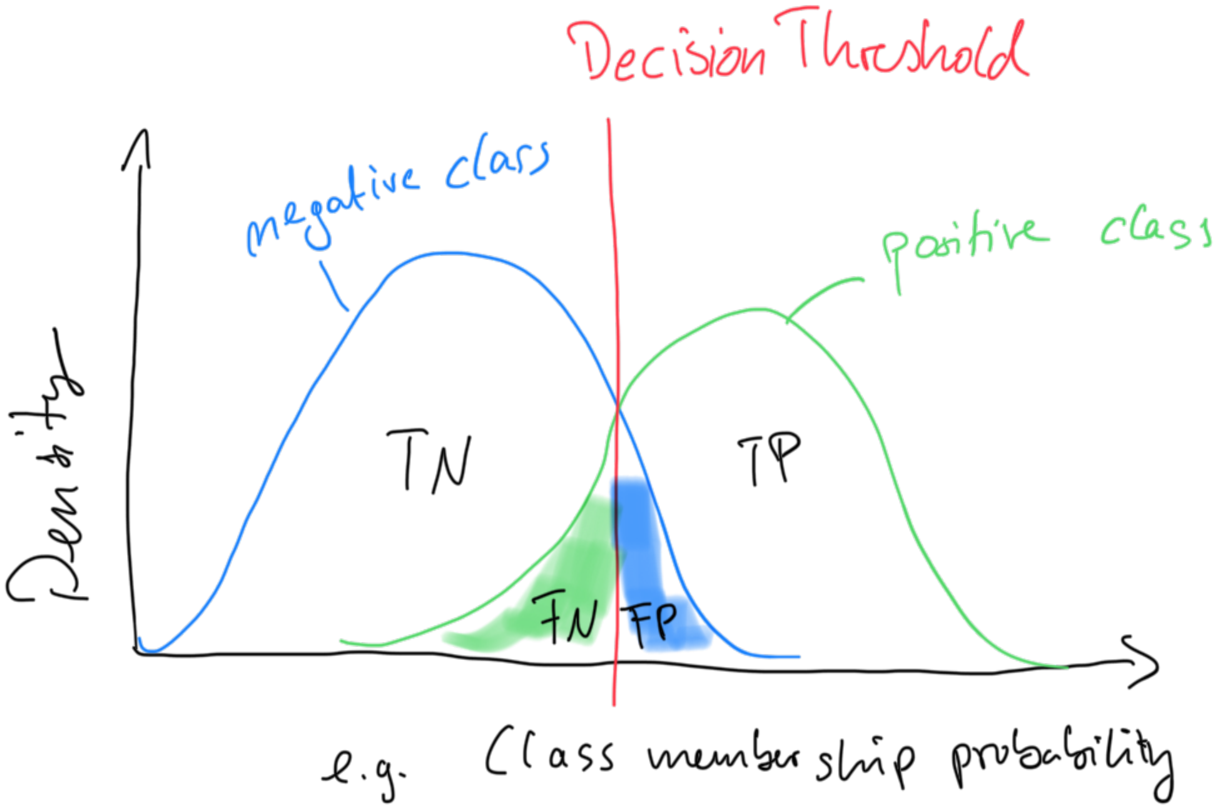
$$\mathrm{FNR} = \frac{\mathrm{FN}}{\mathrm{P}} = \frac{\mathrm{FN}}{\mathrm{FN} + \mathrm{TP}} = 1 - \mathrm{TPR}$$

$$\mathrm{TNR} = \frac{\mathrm{TN}}{\mathrm{N}} = \frac{\mathrm{TN}}{\mathrm{TN} + \mathrm{FP}} = 1 - \mathrm{FPR}$$

Think of it in a spam classification problem (what are true positives, and if you had to pick one at the expense of the other: would you rather decrease the FPR or increase the TPR?)



Species
Iris-setosa
Iris-versicolor
Iris-virginica
PetalWidth[cm]
PetalLength[cm]

Decision Threshold
negative class
positive class
Density
TN
TP
FN
FP
e.g. Class membership probability

# PRECISION, RECALL, AND F1 SCORE

$$PRE = \frac{TP}{TP + FP}$$

$$REC = TPR = \frac{TP}{P} = \frac{TP}{FN + TP}$$

$$F_1 = 2 \cdot \frac{PRE \cdot REC}{PRE + REC}$$

- Terms that are more popular in Information Technology
- Recall is actually just another term for True Positive Rate (or "sensitivity")

# OTHERS: MATTHEW'S CORRELATION COEfficIENT

- Matthews correlation coefficient (MCC) was first formulated by Brian W. Matthews [1] in 1975 to assess the performance of protein secondary structure predictions
- The MCC can be understood as a specific case of a linear correlation coefficient (Pearson r) for a binary classification setting
- Considered as especially useful in unbalanced class settings
- The previous metrics take values in the range between 0 (worst) and 1 (best)
- The MCC is bounded between the range 1 (perfect correlation between ground truth and predicted outcome) and -1 (inverse or negative correlation) — a value of 0 denotes a random prediction.

$$MCC = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \qquad (10)$$

[1] Brian W Matthews. Comparison of the predicted and observed secondary structure of T4 phage lysozyme. Biochimica et Biophysica Acta (BBA)- Protein Structure, 405(2):442–451, 1975.

# CONFUSION MATRIX FOR MULTI-CLASS SETTINGS

Predicted Labels

True Labels

| | Class 0 | Class 1 | Class 2 |
|---|---|---|---|
| Class 0 | T(0,0) | | |
| Class 1 | | T(1,1) | |
| Class 2 | | | T(2,2) |

$$ACC = \frac{T}{n}$$

Confusions readily matrices are traditionally for binary class problems but we can be generalized it to multi-class settings

$$PRE = \frac{TP}{TP + FP}$$

$$ACC = \frac{T}{n}$$

Predicted Labels

| True Labels | Class 0 | Neg Class |
|---|---|---|
| Class 0 | | |
| Neg Class | | |

Predicted Labels

| True Labels | Class 1 | Neg Class |
|---|---|---|
| Class 1 | | |
| Neg Class | | |

Predicted Labels

| True Labels | Class 2 | Neg Class |
|---|---|---|
| Class 2 | | |
| Neg Class | | |

```
[3]  ✓ 0.0s
...  array([1, 0, 2, 2, 0, 0, 2, 2, 2, 0, 0, 2, 2, 1, 2, 1, 0, 0, 0, 0, 0, 2,
            2, 1, 2, 2, 1, 1, 1, 1])

     y_test
[5]  ✓ 0.0s
...  array([1, 0, 2, 2, 0, 0, 2, 1, 2, 0, 0, 1, 2, 1, 2, 1, 0, 0, 0, 0, 0, 2,
            2, 1, 2, 2, 1, 1, 1, 1])
```

Predicted Labels

| True Labels | Class 0 | Class 1 | Class 2 |
|---|---|---|---|
| Class 0 | | | |
| Class 1 | | | |
| Class 2 | | | |

# CONFUSION MATRIX FOR KNN

2-knn_hyperparmeters.ipynb | 3-knn_metrics.ipynb

My Drive > Academics > Machine Learning > Spring 2025 > Lectures > Lecture 006 > lecture6_code_examples > 3-knn_metrics.ipynb > Load and Prepare Datasets > from sklearn.model_selection import train_te

Generate | + Code | + Markdown | Run All | Restart | Clear All Outputs | View data | Jupyter Variables | Outline | ... | pyml (Python 3.11.9)

## Confusion Matrix

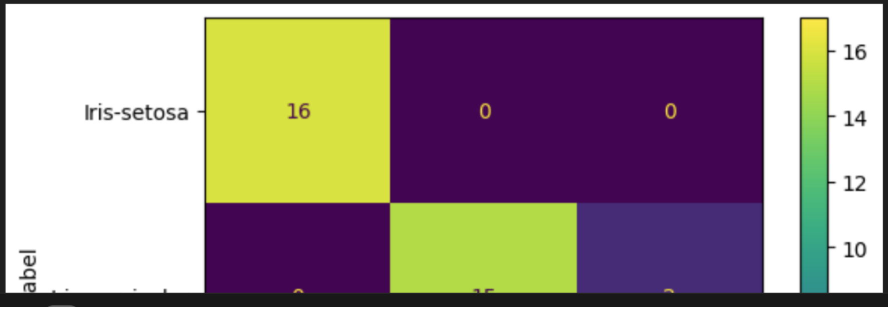
```python
from sklearn.metrics import confusion_matrix
confmat = confusion_matrix(y_test, y_pred)
print(confmat)
```

[5] ✓ 0.0s Open 'confmat' in Data Wrangler — Python

```
[[16  0  0]
 [ 0 15  2]
 [ 0  0 17]]
```

```python
import matplotlib.pyplot as plt
from sklearn.metrics import ConfusionMatrixDisplay
disp = ConfusionMatrixDisplay(confusion_matrix=confmat,
                              display_labels=pipe.classes_)
disp.plot()
plt.show()
```

[6] ✓ 0.9s — Python



# Q/A?