Evaluation Metrics &

Model Selection

Table of Content

What will We Learn Today?

1. Train test split
2. Evaluation Metrics
3. Hyperparameter Tuning

There are two questions:

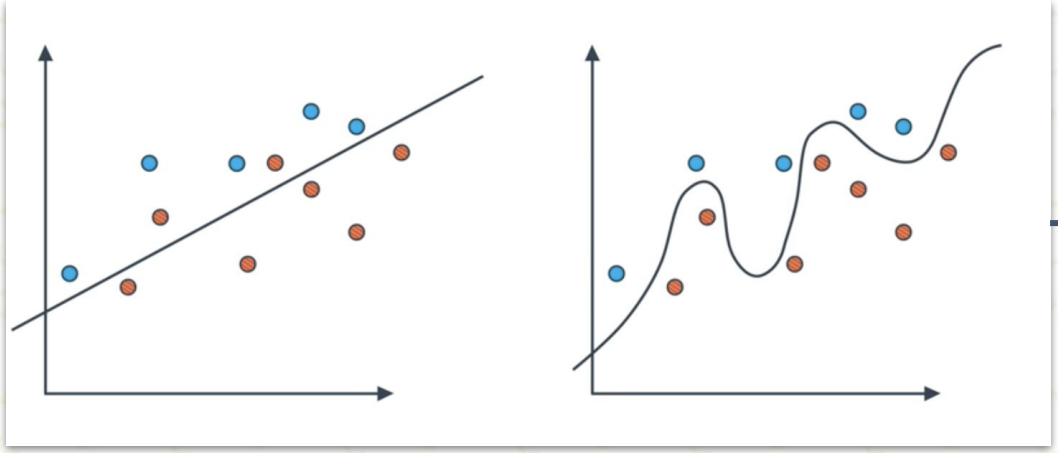
1. how well is my model doing?, is this model good or not?
2. how do we improve it based on these metrics?

DigitalSkola
Uncover The World Of Digital Skills With Us

Train test split

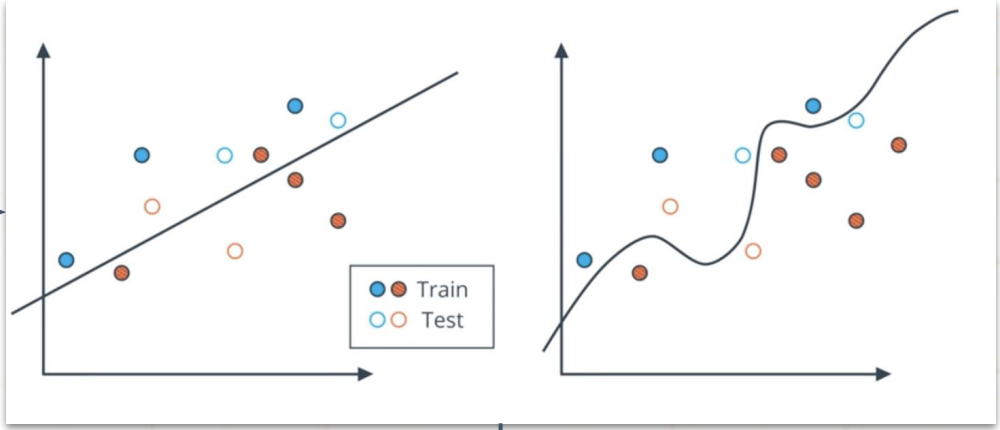
Train test split

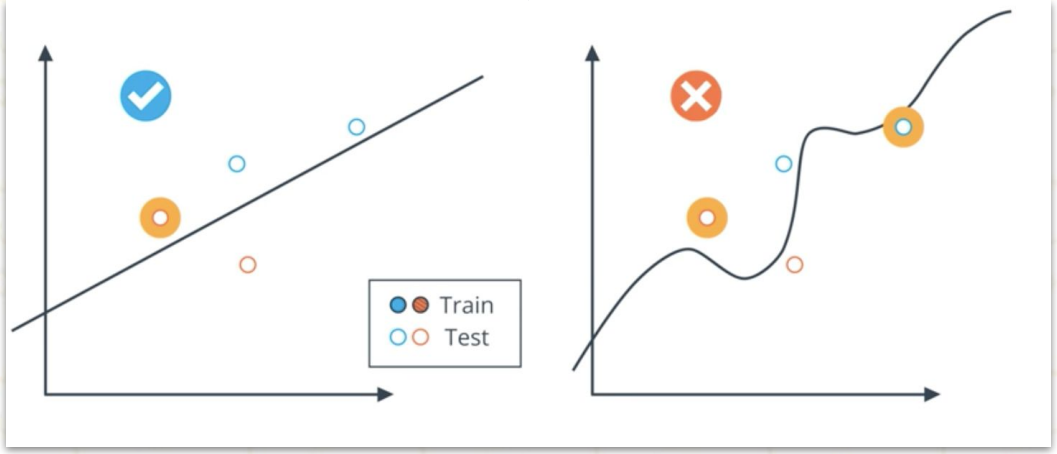
Which model is better?



source: udacity machine learning nanodegree

Split them into training and testing






```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test =
train_test_split(X, y, test_size = 0.25)
```

DigitalSkola
Uncover The World Of Digital Skills With Us

Evaluation Metrics

How well is my model doing? How do we improve the model based on metrics?

Confusion Matrix

The confusion matrix is a simple table that stores these **four values**

10, 000 PATIENTS

DIAGNOSIS

PATIENTS

	Diagnosed Sick	Diagnosed Healthy
Sick	1000 True Positives	200 False Negatives
Healthy	800 False Positives	8000 True Negatives

source: udacity machine learning nanodegree

Two types of error:

1. **Type 1 error** (false positive), this is when we misdiagnose a healthy patient as sick
2. **Type 2 error** (false negative), this is when we misdiagnose a sick patient as healthy

Accuracy

Out of all target data, how many did we classify correctly?

10, 000 PATIENTS

DIAGNOSIS

PATIENTS

	Diagnosed Sick	Diagnosed Healthy
Sick	1000 True Positives	200 False Negatives
Healthy	800 False Positives	8000 True Negatives

$$Accuracy = \frac{(TruePositive + TrueNegative)}{(TP + FP + FN + TN)}$$

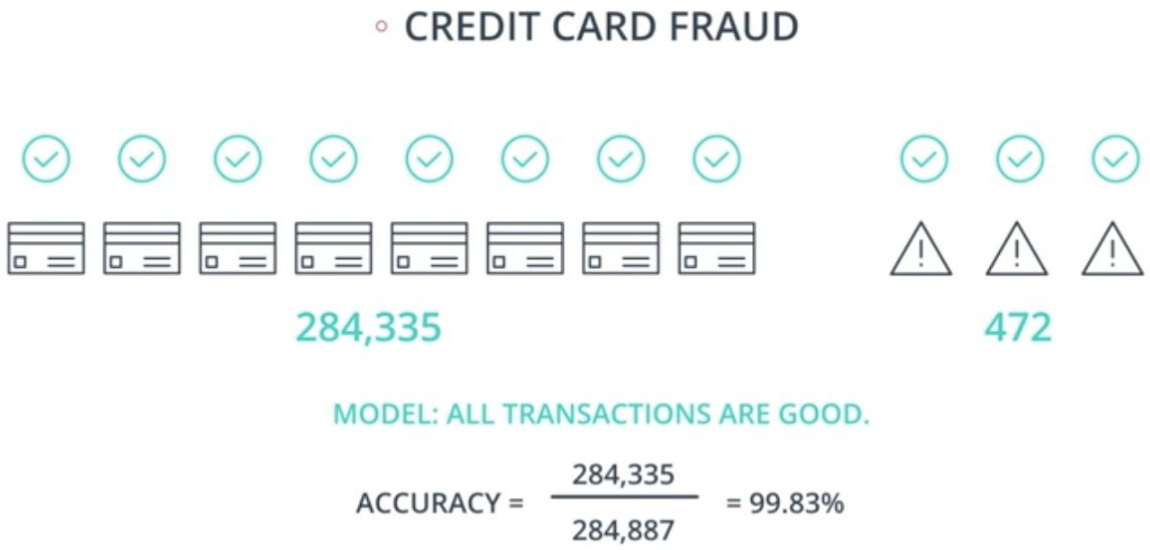
$$= \frac{1.000 + 8.000}{10.000}$$

$$= 90\%$$

source: udacity machine learning nanodegree

```
from sklearn.metrics import accuracy_score
accuracy_score (y_true, y_pred)
```

Accuracy may not be the best metric



	Guessed Positive	Guessed Negative
Positive	True Positives	False Negatives
Negative	False Positives	True Negatives

source: udacity machine learning nanodegree

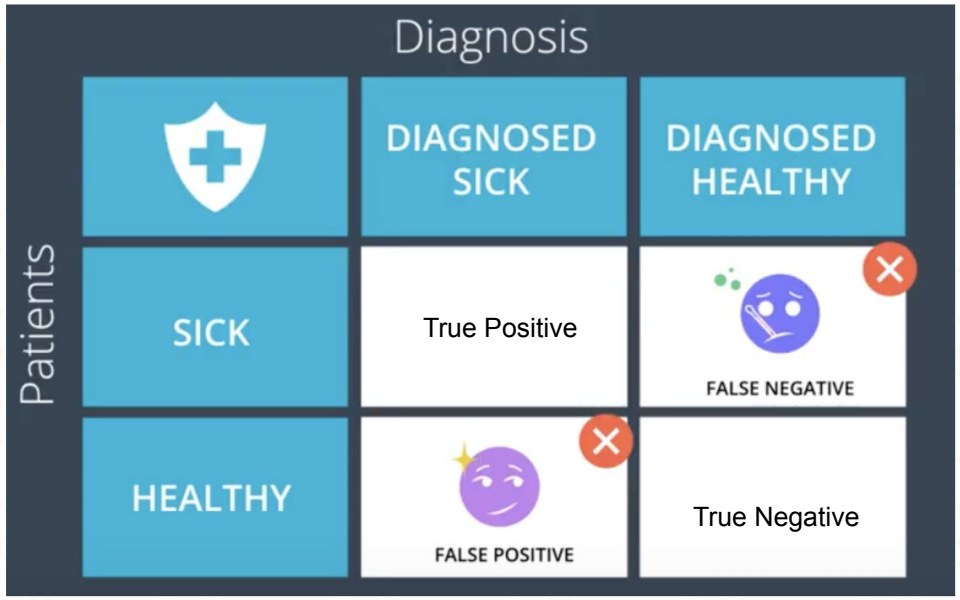
$$Accuracy = \frac{(TruePositive + TrueNegative)}{(TP + FP + FN + TN)}$$

So, what's the next?

Precision, Recall & F1 Score

False positive: when the patient is healthy, and you diagnose them as sick

False negative: when the patient is sick, and you diagnose them as healthy



source: udacity machine learning nanodegree

Which one you choose?

$$Precision = \frac{(TruePositives)}{(TP+FP)}$$

Out of all the points predicted to be positive, how many of them were actually positive?

$$Recall = \frac{(TruePositive)}{(TP+FN)}$$

Out of the points that are labeled positive, how many of them were correctly predicted as positive?

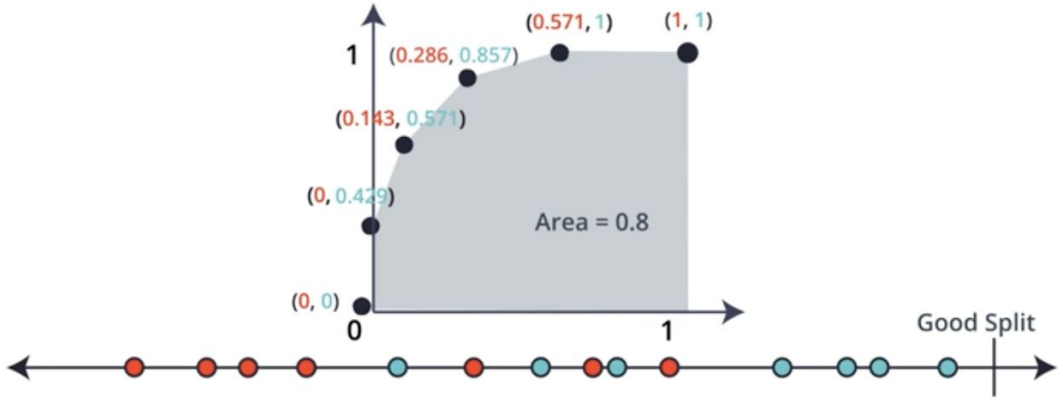
F1 score: the average between precision and recall

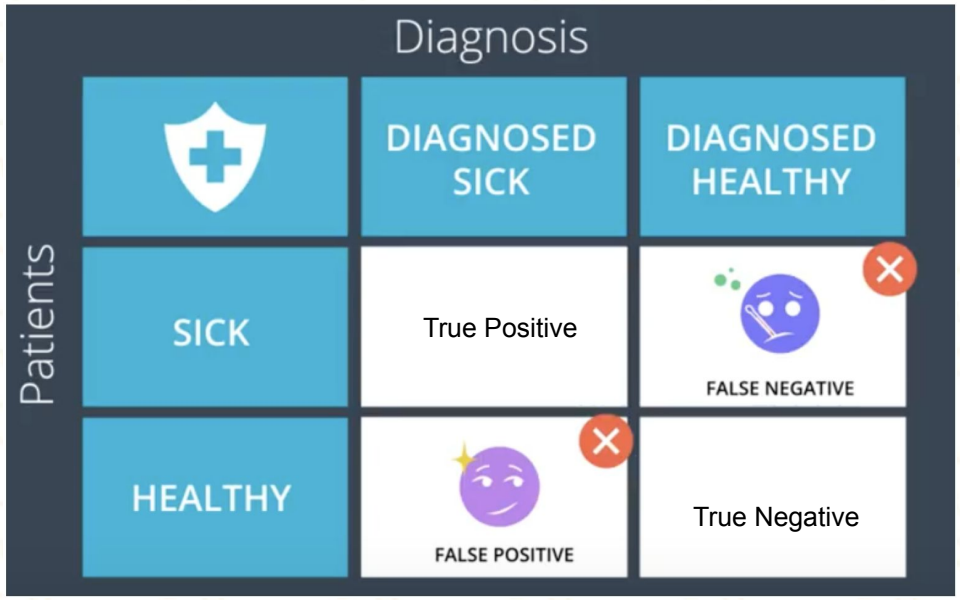
ROC Curve

Receiver Operator Characteristic curve

$$\text{True Positive Rate} = \frac{\text{TRUE POSITIVES}}{\text{ALL POSITIVES}} \quad \textit{Sensitivity}$$

$$\text{False Positive Rate} = \frac{\text{FALSE POSITIVES}}{\text{ALL NEGATIVES}} \quad \textit{Specificity}$$





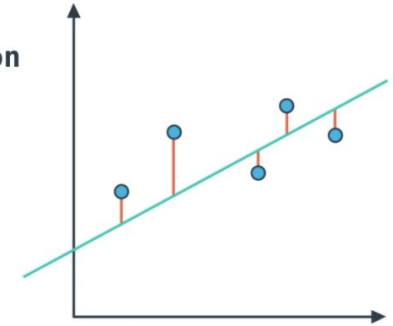
source: udacity machine learning nanodegree

Regression Case

There are three metrics to see how well our model in the regression case:

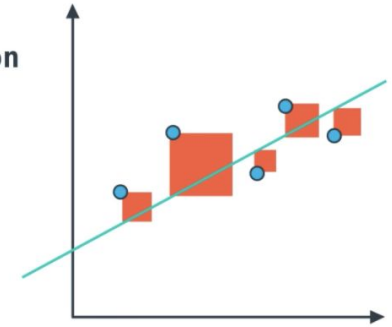
1. Mean absolute error, is just calculated with the mean absolute error function

```
from sklearn.metrics import mean_absolute_error
from sklearn.linear_model import LinearRegression

classifier = LinearRegression()
classifier.fit(X,y)

guesses = classifier.predict(X)

error = mean_absolute_error(y, guesses)
```



2. Mean squared error add the squares of the distance between the points and the line

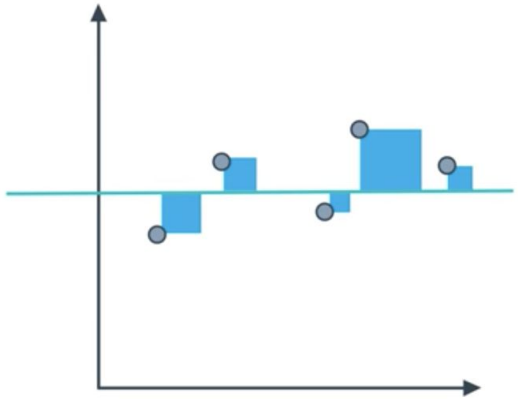
```
from sklearn.metrics import mean_squared_error
from sklearn.linear_model import LinearRegression

classifier = LinearRegression()
classifier.fit(X,y)

guesses = classifier.predict(X)

error = mean_squared_error(y, guesses)
```



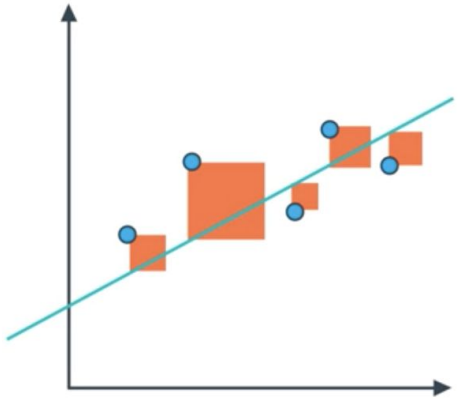
Regression Case

There are three metrics to see how well our model in the regression case:

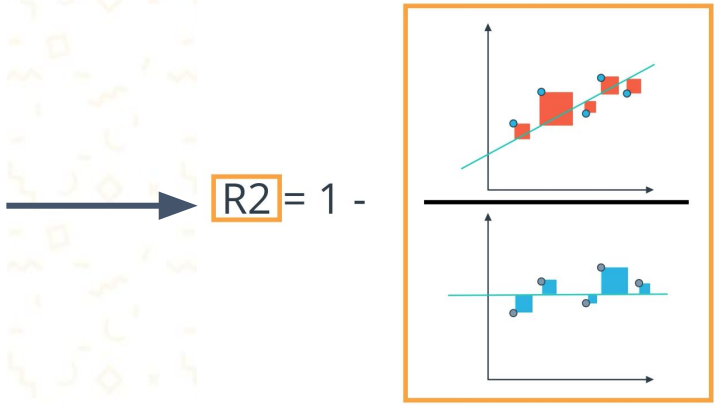
3. R2 score is a metric based on comparing our model to the simplest possible model



Simple model with MSE



Our model with its error




```
from sklearn.metrics import r2_score

r2_score(y_true, y_pred)
```

There are two results:

1. **Bad Model,** the error should be similar, R2 score should be close to 0
2. **Good Model**, the mean squared error from our model should be smaller than the simple model, R2 score should be close to 1

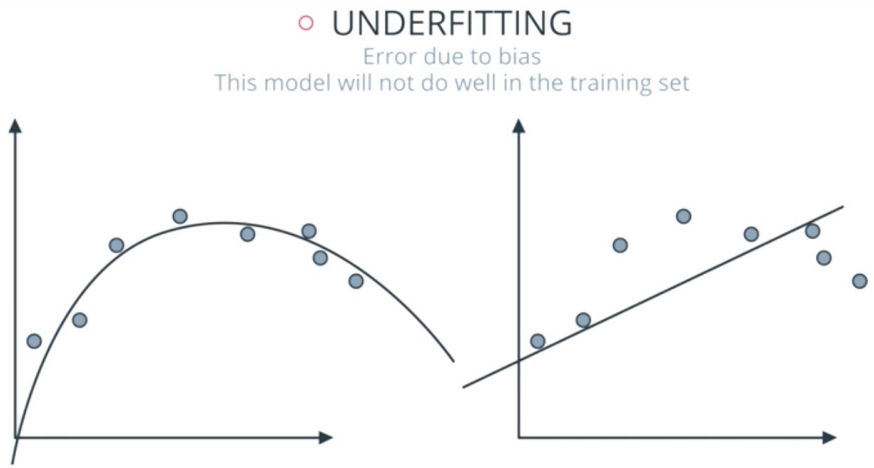
DigitalSkola
Uncover The World Of Digital Skills With Us

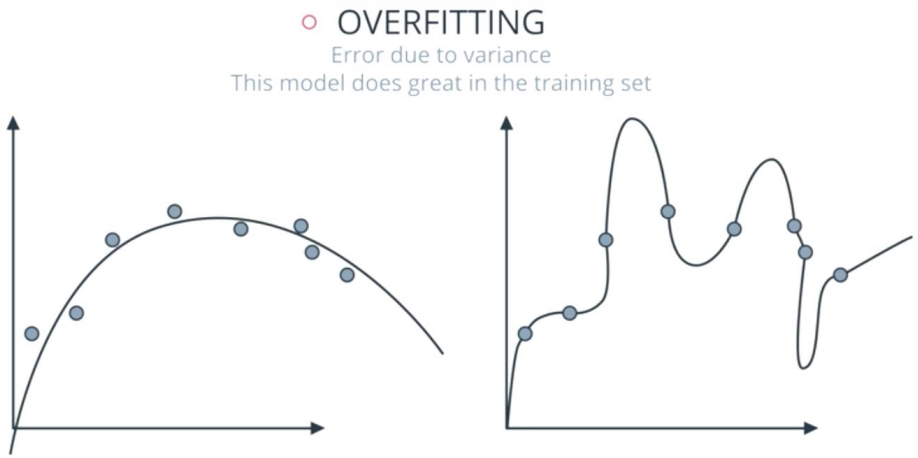
Hyperparameter Tuning

Types of Error

There are two types of error:

1. **Underfitting,** when we oversimplify the problem
2. **Overfitting,** when we overcomplicate the problem

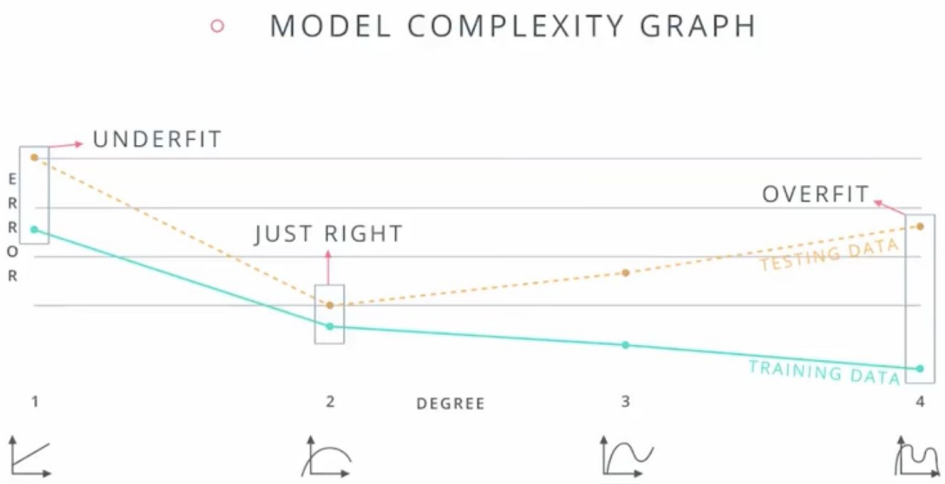




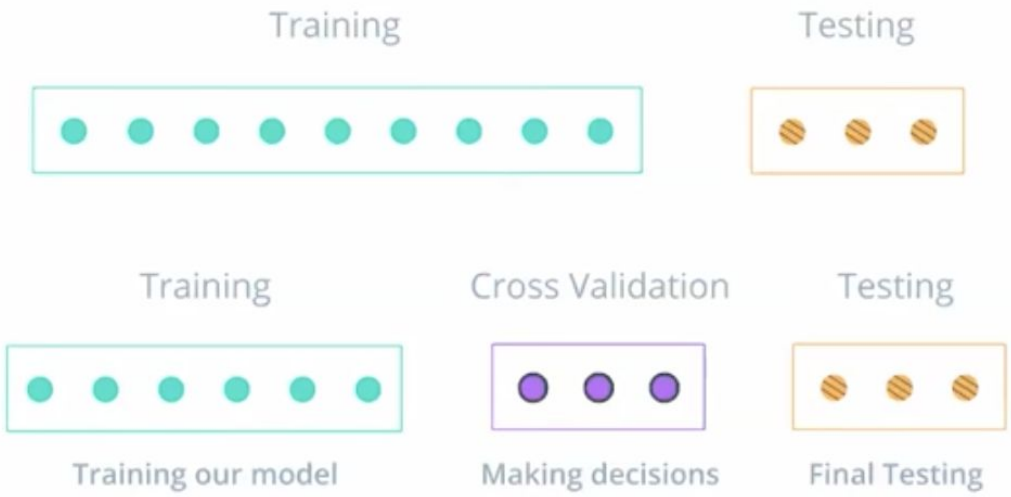
source: udacity machine learning nanodegree

Cross Validation

Instead of having training and testing set, we can use cross validation that will be used for **making decision**



source: udacity machine learning nanodegree



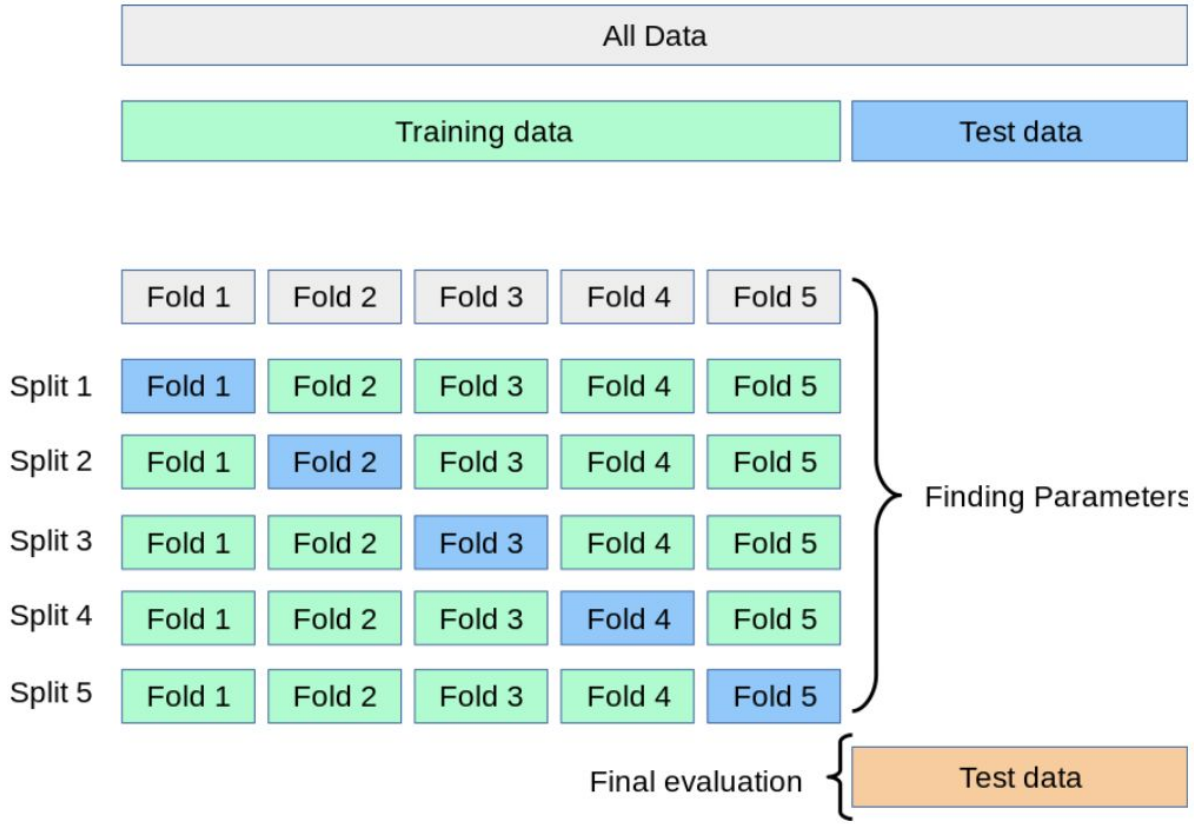
Never use your testing set for training

K-Fold - Cross Validation

Even if we have separated the data into three sets, it is better not to throw away the data that can be useful for training our algorithm.

with K-Fold Cross Validation

- Break our data into K buckets.
- Then, average the results to get a final model

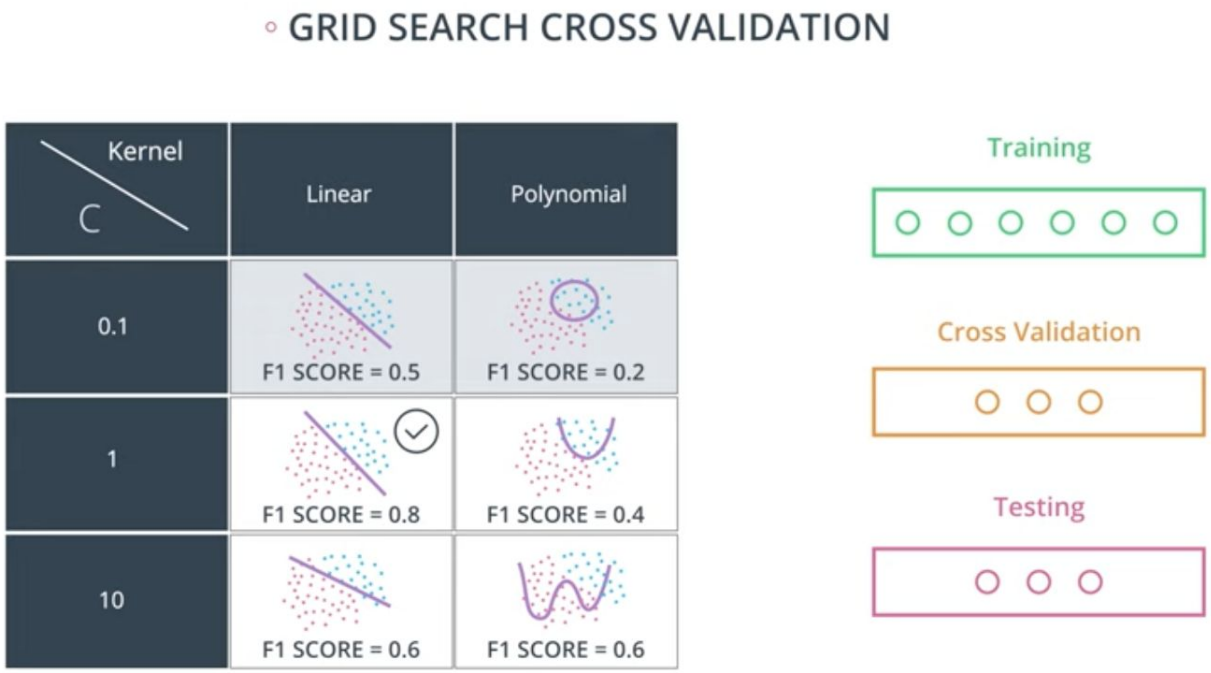


source: scikit-learn.org

```
from sklearn.model_selection import KFold

X = np.array([[1, 2], [3, 4], [1, 2], [3, 4]])
y = np.array([1, 2, 3, 4])

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

Grid Search CV

Finding the best parameter to use it as our model.



source: udacity machine learning nanodegree

```
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import make_scorer
from sklearn.metrics import f1_score

scorer = make_scorer(f1_score)

parameters = {‘max_depth’:[2,4,6,8],
‘min_samples_leaf’:[2,4,6,8]}

grid_obj = GridSearchCV(clf, parameters,
scoring=scorer)
grid_fit = grid_obj.fit(X, y)
```

DigitalSkola

Uncover The World Of Digital Skills With Us

Thank YOU