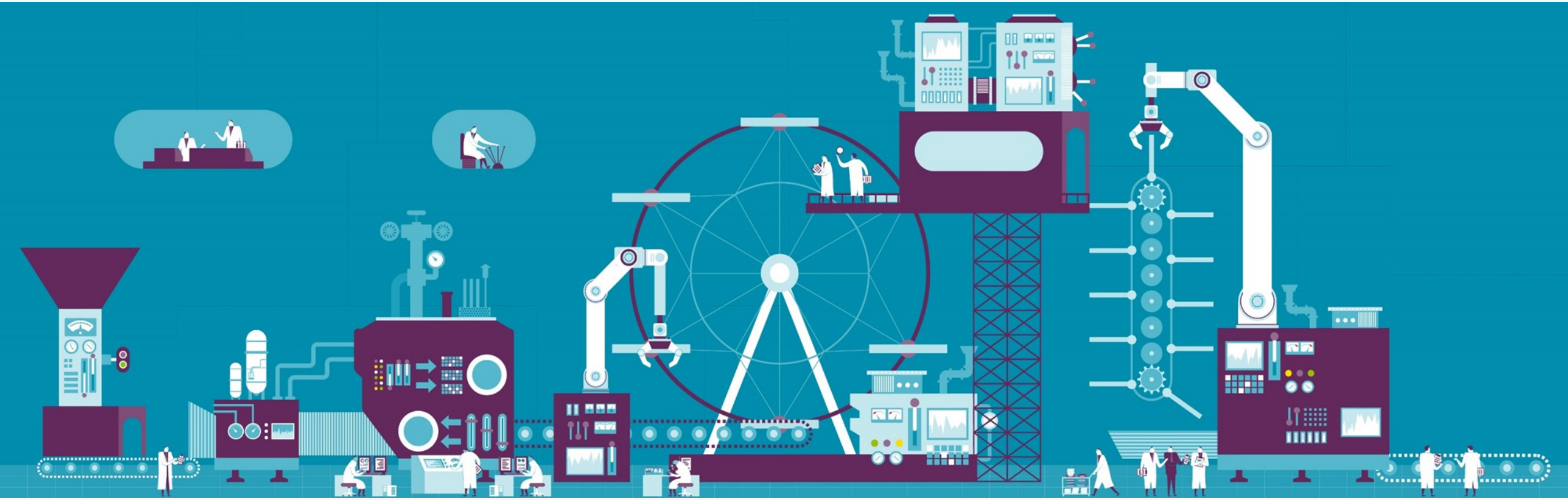
How to Evaluate a Machine Learning Model

Timo Klerx
pmOne Analytics GmbH



Overview

- Task Definition
- Performance Metrics
- Model Estimation
- Hyperparameter Tuning
- Model Fitting

Task Definition

Why do we build a machine learning model?

- Predict class / value for new sample based on training data

- Find a good model

  to solve a specific task

  for unknown data

- How to find this model?
- When is it good? Criterion?

Classification vs. Regression

Classification

- Predict a class value (from a finite set)
  - e.g. “High” vs. “Average” vs. “Low”
- Usual Case: Binary (“High” vs. “Low”)

Regression

- Predict a real-valued number
  - e.g. “Number of swimming pool visitors”

Performance Metrics

Classification

- Accuracy
- Precision
- Recall
- F-Measure
- …

Regression

- (Root) Mean-squared Error ((R)MSE)
- Mean Absolute Error (MAE)
- …

Choosing the right metric is very problem-dependent !!!

Usual Workflow

Questions:

- How to split data?
- Which model to select?
- How to set hyper parameters?
- How to calculate test error?

Example

```
data = load_breast_cancer(return_X_y=True)
logistic = linear_model.LogisticRegression()
X_train, X_test, y_train, y_test =  sklearn.model_selection.train_test_split(X,y)
logistic.fit(X_train, y_train)
y_pred = logistic.predict(X_test)
```

When is a model good?

```
print ('Accuracy:', metrics.accuracy_score(y_test, y_pred))
print ('F1 score:', metrics.f1_score(y_test, y_pred))
print ('Precision:', metrics.precision_score(y_test, y_pred))
print ('Recall:', metrics.recall_score(y_test, y_pred))
```

```
Accuracy: 0.780701754386
F1 score: 0.848484848485
Precision: 0.736842105263
Recall: 1.0
```

- Did I find a good model?

```
print ('F1 score:', metrics.f1_score(y_test, y_pred,pos_label=1))
print ('\nF1 score:', metrics.f1_score(y_test, y_pred,pos_label=0))
print ('Precision:', metrics.precision_score(y_test, y_pred, pos_label=0))
print ('Recall:', metrics.recall_score(y_test, y_pred, pos_label=0))
```

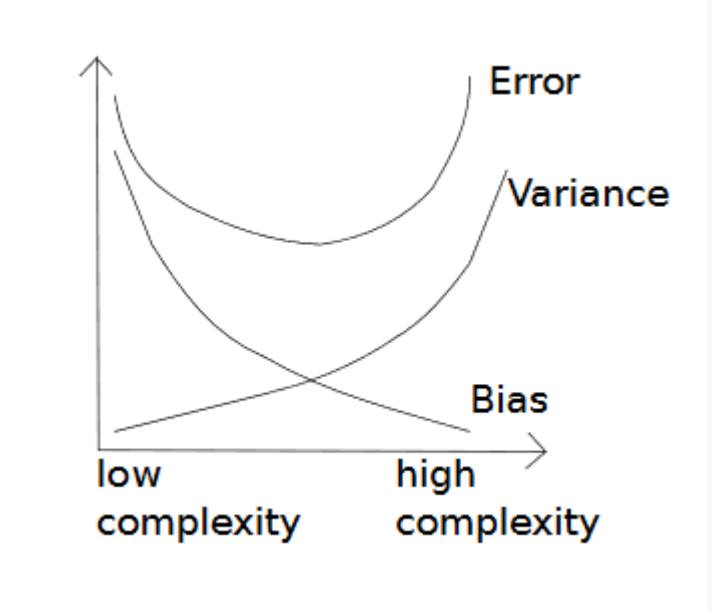
```
F1 score: 0.848484848485

F1 score: 0.603174603175
Precision: 1.0
Recall: 0.431818181818
```

Bias-Variance Tradeoff (I)

Error consists of two parts:

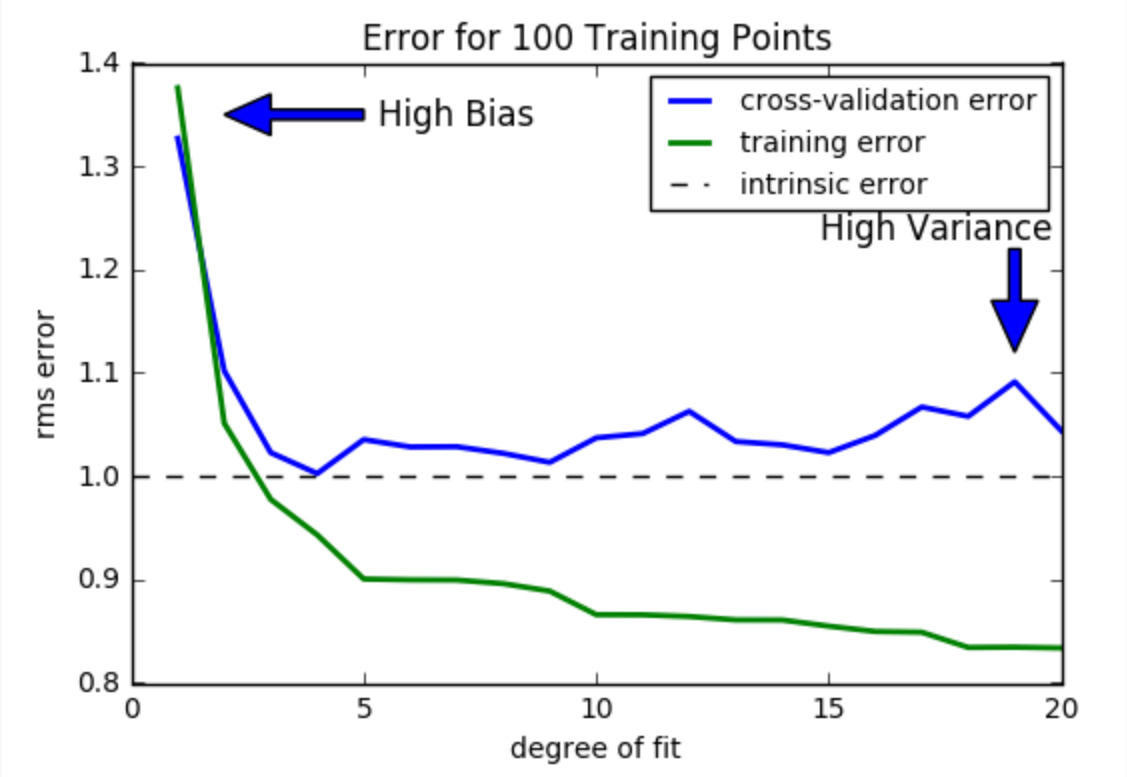
- Bias: How well model representation fits the data
  - Model too simple/complex to fit the data well?

- Variance: Fluctuation for repeated learning
  - What happens if you retrain with other training data?

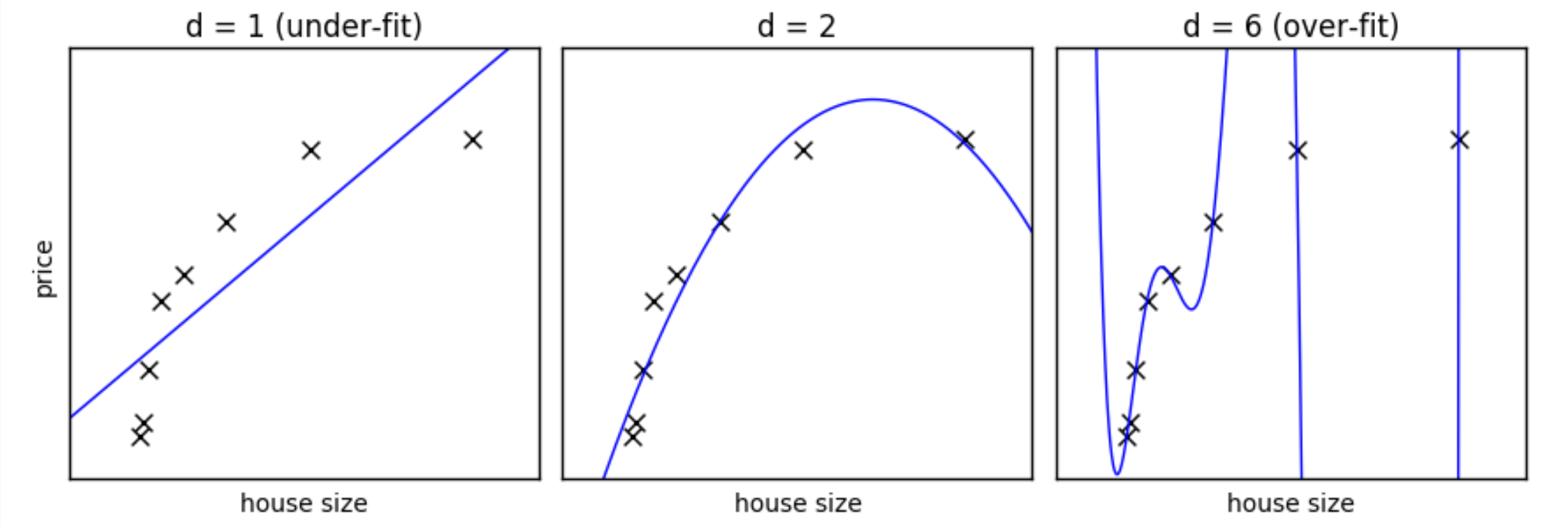


Bias-Variance Tradeoff (II)

- Fit a polynomial
- Which degree to choose?



Overfitting



Error lower for d=6

Probably does not generalize well

Train/Test Split

```
X_train, X_test, y_train, y_test =  sklearn.model_selection.train_test_split(X,y, test_size=0.1)
logistic.fit(X_train, y_train)
y_pred = logistic.predict(X_test)
print ('Accuracy:', metrics.accuracy_score(y_test, y_pred))
print ('F1 score:', metrics.f1_score(y_test, y_pred))
print ('Recall:', metrics.recall_score(y_test, y_pred))
print ('Precision:', metrics.precision_score(y_test, y_pred))
```

```
Accuracy: 1.0
F1 score: 1.0
Recall: 1.0
Precision: 1.0
```

Did I find a perfect model?

```
X_train, X_test, y_train, y_test =  sklearn.model_selection.train_test_split(X,y, test_size=0.1)
logistic.fit(X_train, y_train)
y_pred = logistic.predict(X_test)
print ('Accuracy:', metrics.accuracy_score(y_test, y_pred))
print ('F1 score:', metrics.f1_score(y_test, y_pred))
print ('Recall:', metrics.recall_score(y_test, y_pred))
print ('Precision:', metrics.precision_score(y_test, y_pred))
```

```
Accuracy: 0.912280701754
F1 score: 0.920634920635
Recall: 0.966666666667
Precision: 0.878787878788
```

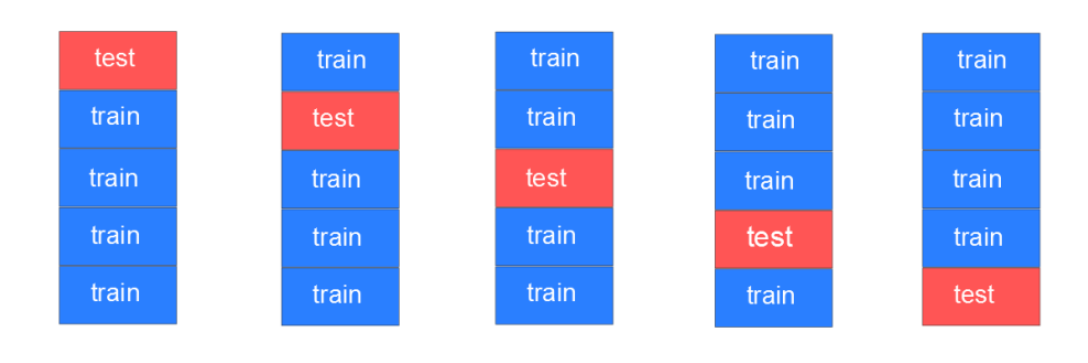
Why the difference?

Single train/test split

- Another split may yield other results
- Train/Test split has high variance
- No good estimation of the model for unknown data

Other approaches

- *k*-fold Cross validation (CV)
  - Do train/test *k* times
  - Special cases:
    - 2 fold
    - Leave One Out (LOO)
- Bootstrapping
  - Random sampling



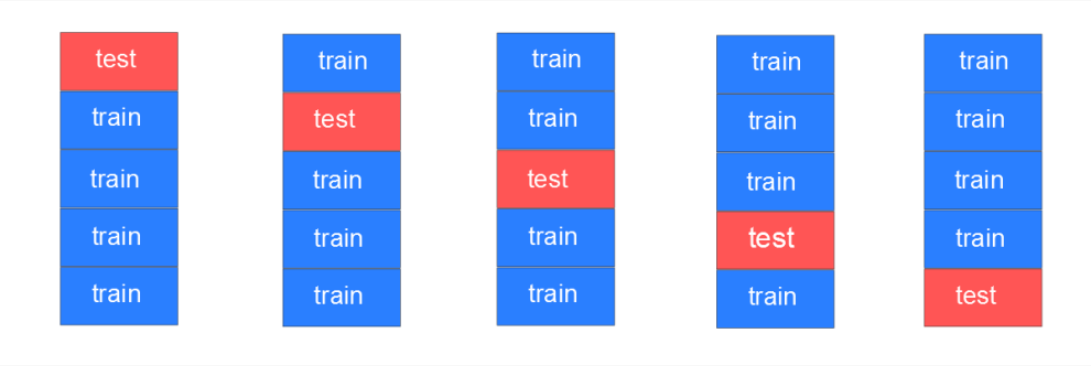
So we’re done?

- Not yet…

- Many models exist
- Find the right model (class)
- Tune its hyperparameters (e.g. “Learning rate”, etc…)

Cross Validation

- For every parameter configuration
  - CV (split train data)
    - Fit model
    - Calculate error
  - Compute score of inner CV (e.g. mean)
- Compute score of parameter config (mean)



So we’re done?

- Not yet…

- Many models exist
- Find the right model (class)
- Tune its hyperparameters (e.g. “Learning rate”, etc…)

- Add another Cross validation loop (Nested CV)

Nested Cross Validation

- Outer CV (split data → train/test)
  - For every parameter configuration
    - Inner CV (split train data again)
      - Fit model
      - Calculate error
    - Compute score of inner CV (e.g. mean)
  - Compute score of parameter config (mean)
- Compute final estimate (mean)

- So which model & config to “deploy”?

Model Fitting

Omit the outer loop

Using more data for training is usually better

- For every parameter configuration
  - CV (split data)
    - Fit model
    - Calculate error
  - Compute score of CV
- Compute score of parameter config
- Use config with best score

Summary

- Choosing right metric is important
- Optimize model for chosen metric
- Choose right model estimation
- Fit final model on more data
- Add seed for recomputation
- Check for unbalanced data
- Be careful with time series

TIMO KLERX
Data Scientist

timo.klerx@pmOne.com
linkedin.com/in/timoklerx