BEST PRACTICES & COMMON PIT FALLS

Practical Lectures

by Chris Emmery (MSc)

@_cmry • @cmry

THIS LECTURE

- Evaluation in Practice
- Setting up for Generalization
  - Data
  - Target
  - Algorithms
- Real Example

EVALUATION IN PRACTICE

```
Epoch 10/15
25000/25000 [==============================] - 13s - loss: 0.0789 - acc: 0.9724 - val_loss: 0.6907 - val_acc: 0.804
Epoch 11/15
25000/25000 [==============================] - 13s - loss: 0.0654 - acc: 0.9766 - val_loss: 0.7954 - val_acc: 0.806
Epoch 12/15
25000/25000 [==============================] - 13s - loss: 0.0540 - acc: 0.9814 - val_loss: 0.9849 - val_acc: 0.798
Epoch 13/15
25000/25000 [==============================] - 13s - loss: 0.0636 - acc: 0.9792 - val_loss: 0.8487 - val_acc: 0.803
Epoch 14/15
25000/25000 [==============================] - 13s - loss: 0.0436 - acc: 0.9851 - val_loss: 0.8941 - val_acc: 0.804
Epoch 15/15
25000/25000 [==============================] - 13s - loss: 0.0354 - acc: 0.9891 - val_loss: 0.9715 - val_acc: 0.801
25000/25000 [==============================] - 2s

Test score: 0.971525938511
Test accuracy: 0.801799985886
```

WHY EVALUATION?

$$f(X) \to Y$$

- Data: $X = [[0.25, 0.13, 0.90], \ldots, [0.56, 0.94, 0.72]]$
- True Labels: $y = \{0, 1, 0, 0, 1, \ldots, 1\}$
- Model: $k\text{-NN}(X; k = 3) \to \hat{y}$
- Predictions: $\hat{y} = \{1, 1, 0, 0, 0, \ldots, 1\}$

How good is our model?
Model = algo + params + data

STATS VS. ML

- **Stats**: hypothesis → data → test → significance + effect size → conclusions
- **ML**: hypothesis → data → model → prediction + baseline metrics → conclusions

Did our model learn something? Are we doing better than x (baseline, previous model iteration, related work, state-of-the-art)?

FACILITATE FORMAL COMPARISON

- True Labels: $y = \{0, 1, 0, 0, 1, \ldots, 1\}$
- Our Predictions: $\hat{y} = \{1, 1, 0, 0, 0, \ldots, 1\}$
- ModelX Predictions: $\hat{m} = \{0, 1, 0, 0, 0, \ldots, 1\}$
- Baseline Predictions: $\hat{b} = \{1, 1, 1, 1, 1, \ldots, 1\}$

	Baseline	ModelX	Our model
accuracy	0.5	0.85	0.90

EVALUATION: THE RIGHT METRIC FOR THE JOB

- Regression:
  - R^2, RMSE, MAE, and more.
- Classification:
  - Accuracy, Precision, Recall, F_1-score.
  - Losses, Rankings, ROC / AUC, and more.

REGRESSION: THE RIGHT METRIC?

- R^2 → how well does my model fit the data?
  - Low $R^2 \neq$ useless model.
  - High $R^2 \neq$ useful model: overfitting, incorrect predictions.
- MAE → how far do my predictions deviate from the actual values?
  RMSE → how far is the model off, and how bad is it (bigger is worse)?
  - MAE / RMSE - choices depend on the problem.
  - MAE / RMSE - interpretation depends on the problem.

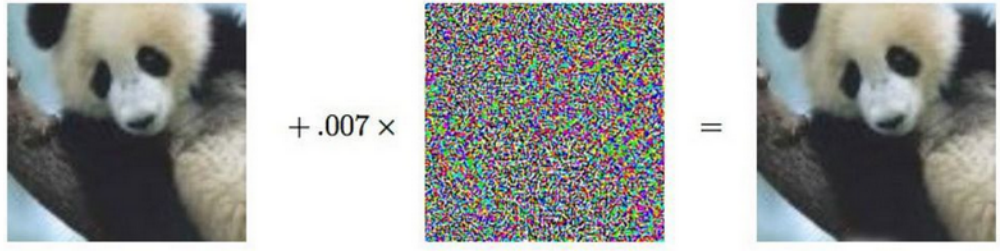
CLASSIFICATION: THE RIGHT METRIC?

- Accuracy: assumes balanced classes, or multiple labels.
- P / R / F_1: good to assess certain characteristics of the predictions, but more classes makes interpretation tricky; (show average over classes, weight averages by class frequency, etc.).

ARE HIGH SCORES EVERYTHING?

spoiler: no

SETTING UP FOR GENERALIZATION



Source: Explaining and Harnassing Adverserial Examples - Goodfellow et al. (2015)

WHY IS GENERALIZATION SO IMPORTANT?

BEFORE ANYTHING: KNOW YOUR DATA!

- How was it collected? Is there a lot of noise?
  - Are there any anomalies in my data / features?
  - How many outliers / missing values?
  - Is standardization / normalization required?
- How was it labelled? What was the agreement?
  - Information leakage (pollution, contamination)?
- What are useful features, which are less useful?

TARGET

- Do you have a target?
- Is the task realistic given the data?
- Are you doing what you want to be doing?
- Are you measuring the performance correctly?

Data + Target: Your data is a sample, is it a good sample (size / distribution)?

ALGORITHMS

- Do I need Machine Learning?
- Which kind of algorithm fits my problem?
- What are the properties of an algorithm / what are the assumptions / how is it expected to behave?
- How interpretable is the algorithm?
- What do the hyper-parameters do? Will it affect x?
- How is it performing; bias / variance / underfitting / overfitting?

MOST IMPORTANT: FAIR EVALUATION

- Choose a evaluation scheme that fits with your data (more later).
- Always make sure you have a baseline:
  - Majority class for classification.
  - Mean target value for regression.
  - Even better: make your own!
  - Take some standard parameter 'simple' algorithm to compare.
- Leave your test side aside **until the very end**!

EVALUATION SCHEMES

- Hold-out
- k-fold Cross Validation
    - k-fold Cross Validation (nested)
- Leave-One-Out

When to use what?

REAL EXAMPLE

DISCLAIMER: RULES ARE NOT CLEAR-CUT IN PRACTICE

- Good advice $\neq$ always applicable advice.
- Many factors influence how to tackle certain prediction tasks.

DATASET

```
import pandas as pd

df = pd.DataFrame.from_csv('kc_house_data.csv')
df.head()
```

id	date	price	bedrooms	bathrooms	s
7129300520	20141013T000000	221900.0	3	1.00	1
6414100192	20141209T000000	538000.0	3	2.25	2
5631500400	20150225T000000	180000.0	2	1.00	1

```
del df['date']
```

PREPPING THE DATA

```
y = df.pop('price')
X = df.as_matrix()
```

```
from sklearn.preprocessing import StandardScaler

X = StandardScaler().fit_transform(X)
```

```
from sklearn.model_selection import train_test_split

X, X_hidden, y, y_hidden = \
  train_test_split(X, y, test_size=0.5)
```

```
from sklearn.preprocessing import PolynomialFeatures

pl = PolynomialFeatures()
X = pl.fit_transform(X)
X_hidden = pl.transform(X)
```

OUR BASIC MODEL

```
from sklearn.linear_model import LinearRegression

lr = LinearRegression(n_jobs=-1)
lr.fit(X, y)
```

```
from sklearn.metrics import mean_absolute_error

ŷ_lr = lr.predict(X)
mean_absolute_error(y, ŷ_lr)
```

```
126761.69464928517
```

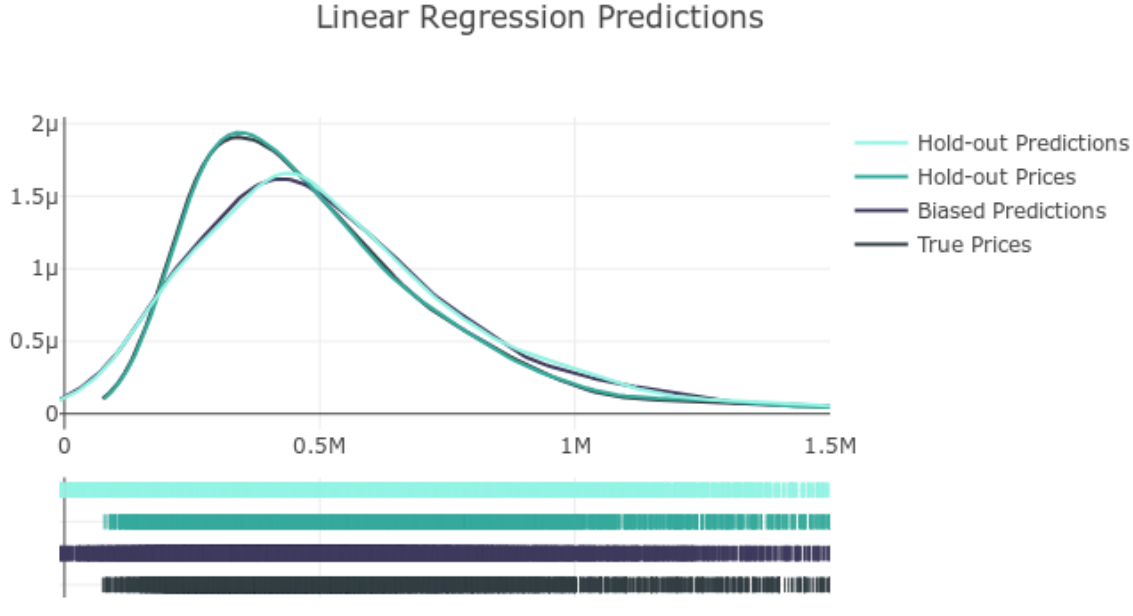
```
mean_absolute_error(y, [y.mean()] * len(y))
```

```
236047.97702255406
```

HIDDEN PERFORMANCE

```
ŷ_lr_hidden = lr.predict(X_hidden)
mean_absolute_error(y_hidden, ŷ_lr_hidden)
```

```
126889.91221034051
```



DIFFERENT MODEL

```
from sklearn.kernel_ridge import KernelRidge

svr = KernelRidge(kernel='rbf')
svr.fit(X, y)
```

```
ŷ_rf = svr.predict(X)
mean_absolute_error(y, ŷ_rf)
```

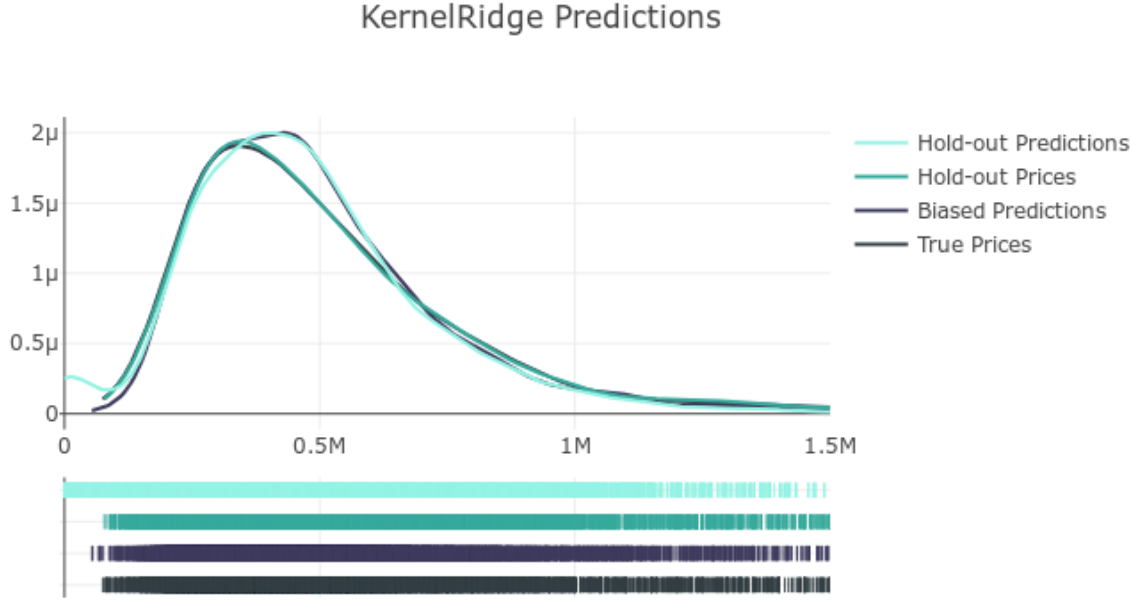
```
129596.0710315921
```

	BL	LR	KR
MAE	236047	126889	129596

HIDDEN PERFORMANCE

```
ŷ_rf_hidden = svr.predict(X_hidden)
mean_absolute_error(y_hidden, ŷ_rf_hidden)
```

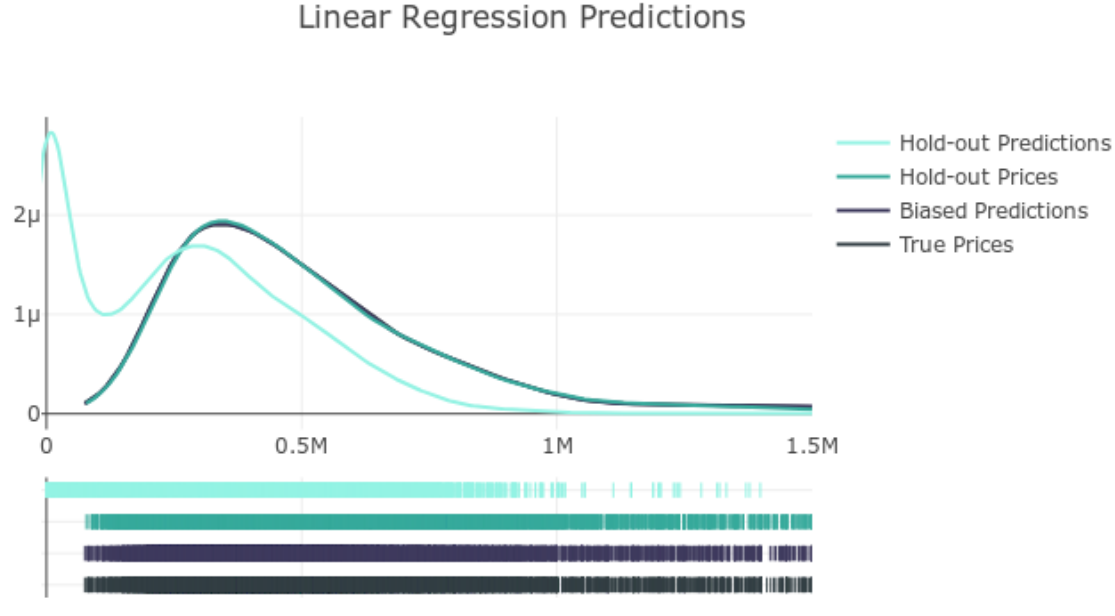
```
129596.0710315921
```



DON'T TUNE ON TRAINING

```
KernelRidge(alpha=alpha, gamma=0.1, kernel='rbf')
```

```
...
MAE for alpha=0.1 : 33690.2981898
MAE for alpha=0.01 : 5535.04330513
MAE for alpha=0.001 : 1423.82136117
```



USING CROSS-VALIDATION

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

```
from sklearn.model_selection import KFold, GridSearchCV

p_grid = {"alpha": [10, 1, 0.1, 0.01, 0.001, 0.0001]}
inner_cv = KFold(n_splits=10, shuffle=True, random_state=42)

clf = GridSearchCV(estimator=KernelRidge(kernel='rbf'),
                   param_grid=p_grid, cv=inner_cv, n_jobs=-1)
clf.fit(X_train, y_train)
```

	BL	LR	KR	KRF	KR CV	k-NN CV
MAE	236047	126889	129596	297466	124109	**93482**

FINAL RESULTS