Applications of metric evaluation

PREDICTING CTR WITH MACHINE LEARNING IN PYTHON

Kevin Huo
Instructor

Four categories of outcomes

	Actual Positives	Actual Negatives
Positive Predictions	True Positives (TP)	False Positives (FP)
Negative Predictions	False Negatives (FN)	True Negatives (TN)

- First part of category (true/false) represents whether model was correct or not
- Second part of the category (positive/negative) represents the target label the model applied

Interpretations of four categories

- If model predicts there is a click, then there is a bid for that impression which costs money
- If no click predicted, no bidding and hence no cost
- True positives (TP): money gained (impressions paid for that were clicked on).
- False positives (FP): money lost (impressions that were paid for, but not clicked).
- True negatives (TN): money saved (no click predicted so no impressions bought).
- False negatives (FN): money lost out on (no click predicted, but would have been actual click in reality).

Confusion matrix

```
print(confusion_matrix(y_test, y_pred))
```

```
[[8163  166]
 [1517  154]]
```

```
# Order: tn, fp, fn, tp
print(confusion_matrix(y_test, y_pred).ravel())
```

```
[8163, 166, 1517, 154]
```

ROI analysis

- Assume: some cost `c` and return `r` per X number of impressions

```
total_return = tp * r
```

```
total_cost = (tp + fp) * c
```

```
tp * r > (tp + fp) * c
```

```
roi = total_return / total_spent
```

Let's practice!

Model evaluation

PREDICTING CTR WITH MACHINE LEARNING IN PYTHON

Kevin Huo
Instructor

Precision and recall

- Precision: proportion of clicks relative to total number of impressions, `TP / (TP + FP)`
  - Higher precision means higher ROI on ad spend
- Recall: the proportion of clicks gotten of all clicks available, `TP / (TP + FN)`
  - Higher recall means better targeting of relevant audience

Calculating precision and recall

```
print(precision_score(
  y_test, y_pred, average = 'weighted'))
```

```
0.73
```

```
print(recall_score(
  y_test, y_pred, average = 'weighted'))
```

```
0.75
```

Baseline classifiers

- It is important to evaluate classifiers relative to an appropriate baseline
  - The baseline here, due to imbalanced nature of click data, is a classifier that always predicts no click

```
y_pred = np.asarray([0 for x in range(len(X_test))])
```

```
[[0]
 [0] ...]
```

Implications on ROI analysis

- For the baseline classifier, `tp` and `fp` will be zero
- Therefore total return and total spend will be zero, and ROI undefined
- Confusion matrix via `confusion_matrix()` along with `ravel()` to get the four categories of outcomes

```
total_return = tp * r
total_spent = (tp + fp) * cost
roi = total_return / total_spent
```

Let's practice!

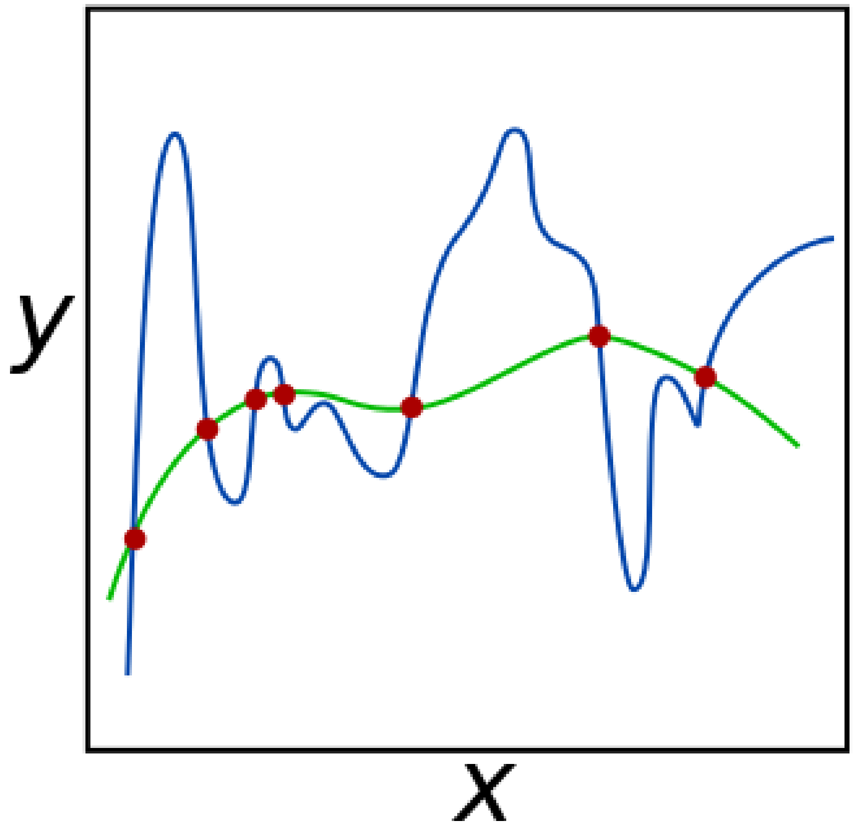
PREDICTING CTR WITH MACHINE LEARNING IN PYTHON

Tuning models

PREDICTING CTR WITH MACHINE LEARNING IN PYTHON

Kevin Huo
Instructor

Regularization

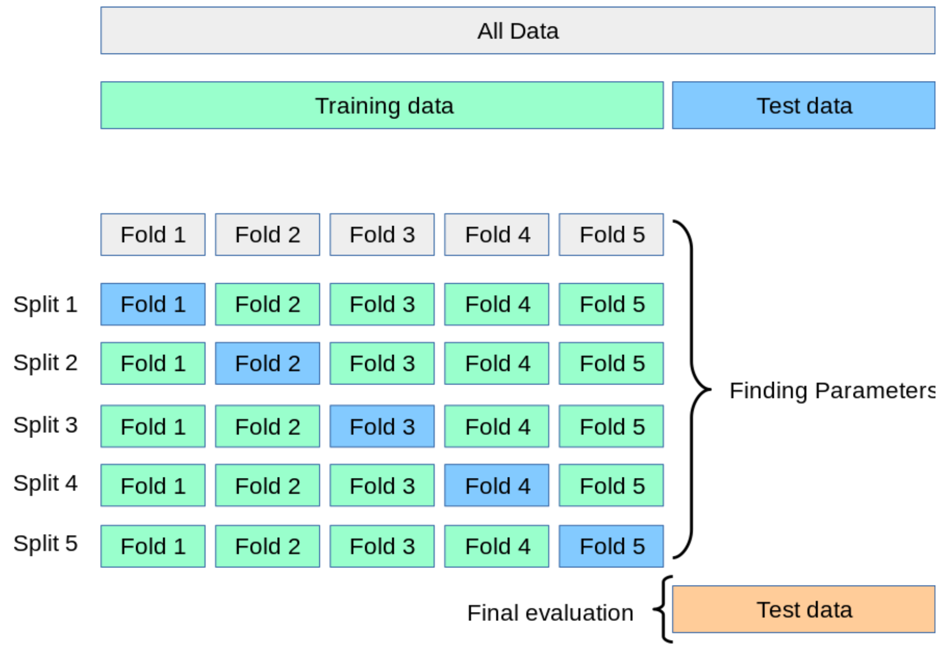


- Regularization: addressing overfitting by altering the magnitude of coefficients of parameters within a model
- Regularization can increase performance metrics and hence ROI on ad spend

Examples of regularization

- Logistic Regression: the `C` parameter is the inverse of the regularization strength.
- From least to most complex: `C=0.05 < C=0.5 < C=1`
- Decision Tree: the `max_depth` parameter controls how many layers deep the tree can grow.
- From least to most complex: `max_depth=3 < max_depth=5 < max_depth=10`

Cross validation



- For each of the `k` folds, that fold will be used as a testing set (for validation) while other `k-1` are used as training.
- Therefore, you have `k` evaluations of model performance.
- Note you still have the separate evaluation testing set.

Examples of cross validation

```
k_fold = KFold(n_splits = 4, random_state = 0)
```

```
for i in [3, 5, 10]:
  clf = DecisionTreeClassifier(max_depth = i)
  cv_precision = cross_val_score(
    clf, X_train, y_train, cv = k_fold,
    scoring = 'precision_weighted')
```

- Scoring strings: `precision_weighted, recall_weighted, roc_auc`

Let's practice!

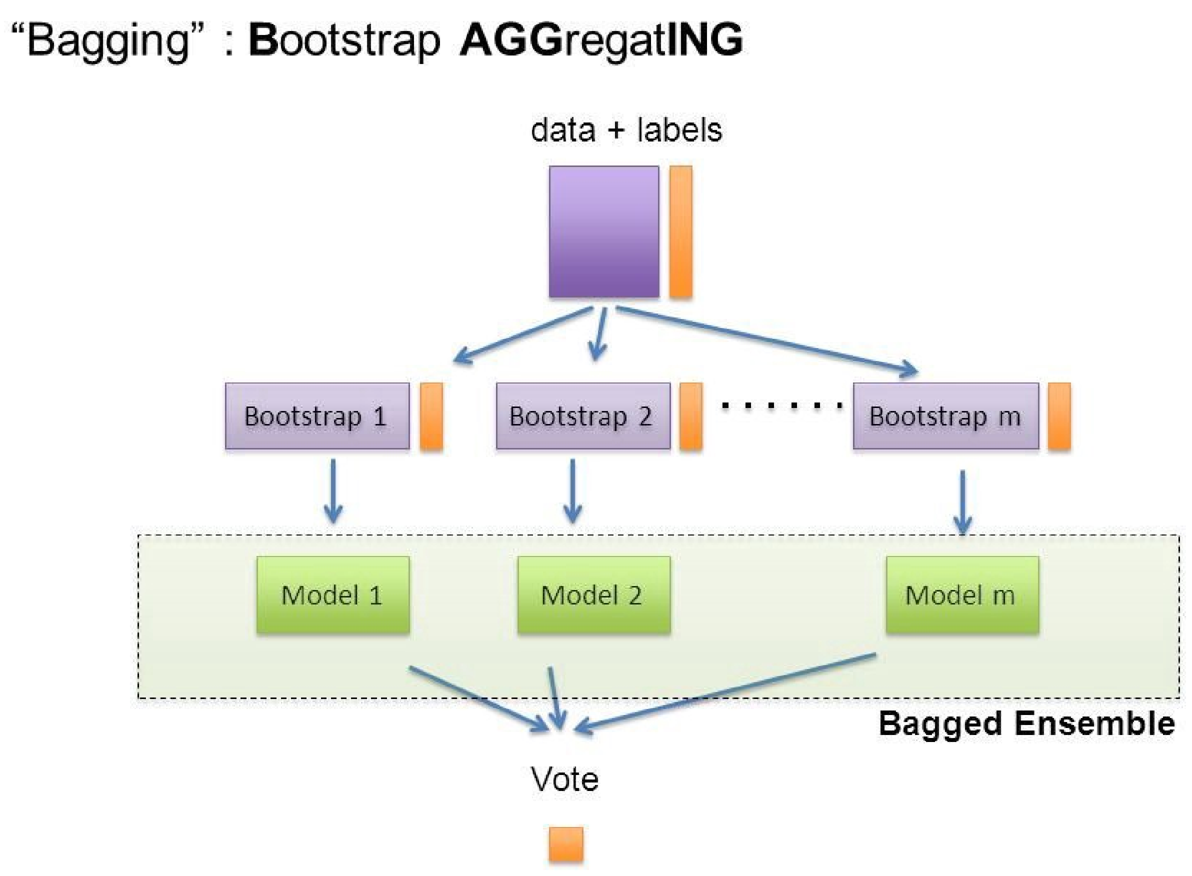
PREDICTING CTR WITH MACHINE LEARNING IN PYTHON

Ensembles and hyperparameter tuning

PREDICTING CTR WITH MACHINE LEARNING IN PYTHON

Kevin Huo
Instructor

Ensemble methods



- Bagging: random samples selected for different models, then models are individually trained and combined.

Random forests

```
clf = RandomForestClassifier()
print(clf)
```

```
RandomForestClassifier(
  bootstrap=True,
  ...
  max_depth = 10,
  ...
  n_estimators = 100,
  ...)
```

Hyperparameter tuning

- Hyperparameter: parameters configured before training, and external to a model
- Examples of parameters but NOT hyperparameters: slope coefficient in linear regression, weights in logistic regression, etc.
- Examples of hyperparameters: `max_depth` , `n_estimators` , etc.

Grid search

```
param_grid = {'n_estimators': n_estimators,
              'max_depth': max_depth}
```

```
clf = GridSearchCV(estimator = model,
                   param_grid = param_grid,
                   scoring = 'roc_auc')
print(clf.best_score_)
print(clf.best_estimator_)
```

```
0.6777
RandomForestClassifier(max_depth = 100, ...)
```

Let's practice!

PREDICTING CTR WITH MACHINE LEARNING IN PYTHON