Introduction to click-through rates

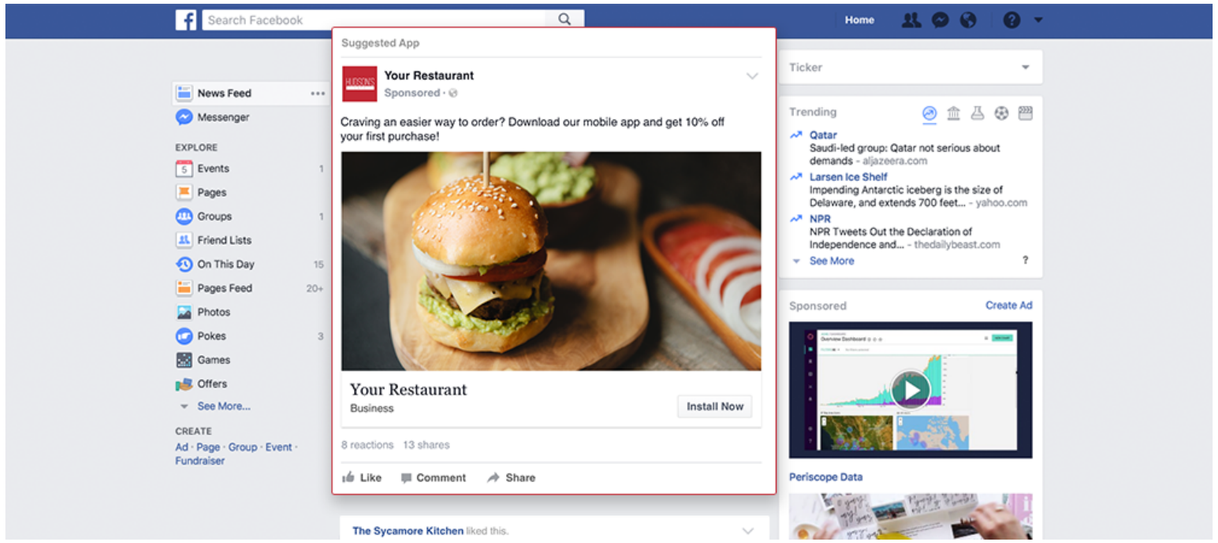
PREDICTING CTR WITH MACHINE LEARNING IN PYTHON

Kevin Huo
Instructor

datacamp

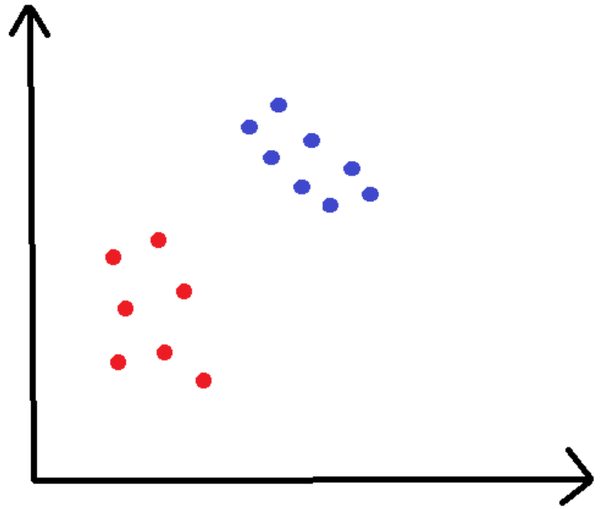
Click-through rates

- Click-through rate: # of clicks on ads / # of views of ads
- Companies and marketers serving ads want to maximize click-through rate
- Prediction of click-through rates is critical for companies and marketers



A classification lens

- Classification: assigning categories to observations
- Classifiers use training data and are evaluated on testing data
- Target: a binary variable, 0/1 for non-click or click
- Feature: any variable used to help predict the target



A brief look sample data

click	hour	banner_pos	device_type	device_conn_type
0	14102100	0	1	2
0	14102100	0	1	0
0	14102100	0	1	0
0	14102100	0	1	0
0	14102100	1	1	0

- Each row represents a particular outcome of click or not click for a given user for a given ad
- Filtering for columns can be done through `.isin()` : `df.columns.isin(['device'])]`
- Assuming `y` is a column of clicks, CTR can be found by: `y.sum()/len(y)`

Analyzing features

```
print(df.device_type.value_counts())
```

```
1    45902
0    2947
```

```
print(df.groupby('device_type')['click'].sum())
```

```
0     633
1    7890
```

Let's practice!

PREDICTING CTR WITH MACHINE LEARNING IN PYTHON

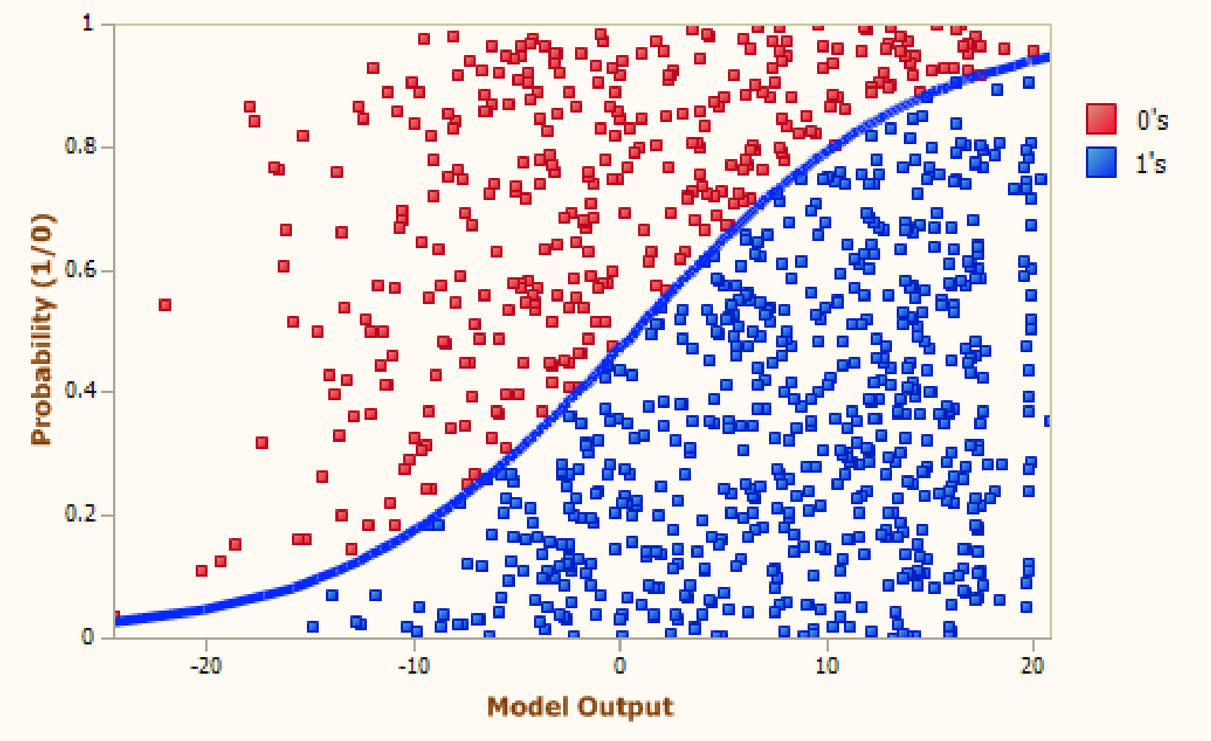
Overview of machine learning models

PREDICTING CTR WITH MACHINE LEARNING IN PYTHON

Kevin Huo
Instructor

datacamp

Logistic regression



- Logistic regression: linear classifier between dependent variable and independent variables

Training the model

- Can create the model via: `clf = LogisticRegression()`
- Each classifier has a `fit()` method which takes in an `X_train, y_train`:
  `clf.fit(X_train, y_train)`
- `X_train` is the vector of training features, `y_train` is the vector of training targets
- Classifier should only see training data to avoid "seeing answers beforehand"

Testing the model

- Each classifier has a `predict()` method which takes in an `X_test` to generate a `y_test` as follows:

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

- `predict_proba()` method produces probability scores

```
array([0.2, 0.8], [0.4, 0.6] ..., [0.1, 0.9] [0.3, 0.7]])
```

- Score reflects probability of a particular ad being clicked by particular user

Evaluating the model

- Accuracy: the percentage of test targets correctly identified
- `accuracy_score(y_test, y_pred)`
- Should not be the only metric to evaluate model, particularly in imbalanced datasets
- CTR prediction is an example where classes are imbalanced

Let's practice!

PREDICTING CTR WITH MACHINE LEARNING IN PYTHON

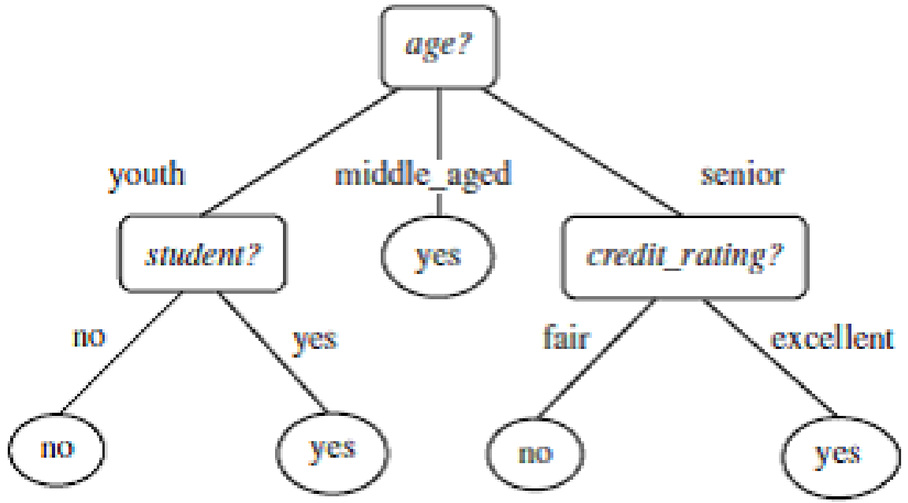
CTR prediction using decision trees

PREDICTING CTR WITH MACHINE LEARNING IN PYTHON

Kevin Huo
Instructor

datacamp

Decision trees



- Nodes represent the features
- Branches represent the decisions based on features

- Sample outcomes are shown in table below:
- First split is based on age of application
- For youth group, second split is based on student status
- Model provides heuristics for understanding

	is_student	loan
middle_aged		1
youth	no	0
youth	yes	1

Training and testing the model

- Create via: `clf = DecisionTreeClassifier()`
- Similar to logistic regression, a decision tree also involves `clf.fit(X_train, y_train)` for training data and `clf.predict(X_test)` for testing labels:

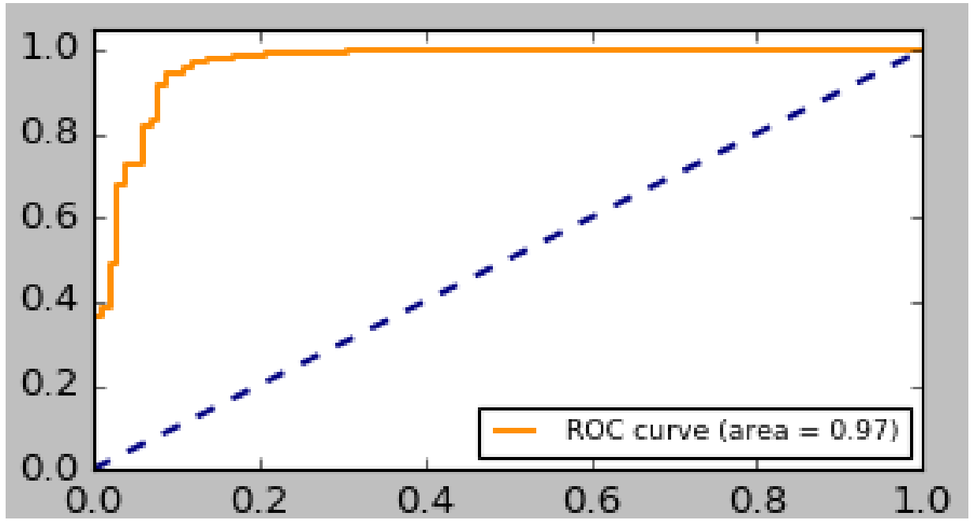
```
array([0, 1, 1, ..., 1, 0, 1])
```

- `clf.predict_proba(X_test)` for probability scores:

```
array([0.2, 0.8], [0.4, 0.6] ..., [0.1, 0.9] [0.3, 0.7]])
```

- Example for randomly splitting training and testing data, where testing data is 30% of total sample size: `train_test_split(X, y, test_size = .3, random_state = 0)`

Evaluation with ROC curve



- True positive rate (Y-axis) = #(classifier predicts positive, actually positive) / #(positives)
- False positive rate (X-axis) = #(classifier predicts positive, actually negative) / #(negatives)
- Dotted blue line: baseline AUC of 0.5
- Want orange line (AUC) to be as close to 1 as possible

AUC of ROC curve

```
Y_score = clf.predict_proba(X_test)
```

```
fpr, tpr, thresholds = roc_curve(Y_test, Y_score[:, 1])
```

- `roc_curve()` inputs: test and score arrays

```
roc_auc = auc(fpr, tpr)
```

- `auc()` input: false-positive and true-positive arrays
- If model is accurate and CTR is low, you may want to reassess how the ad message is relayed and what audience it is targeted for

Let's practice!

PREDICTING CTR WITH MACHINE LEARNING IN PYTHON