# Practical Predictive Modeling in Python

Robert Dempsey
robertwdempsey.com

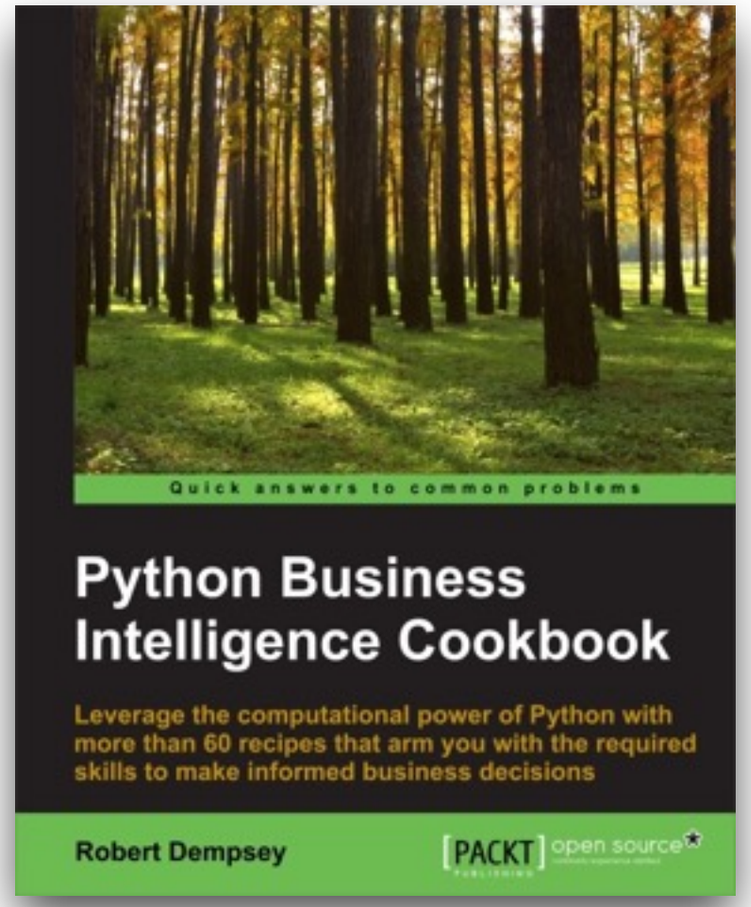
# Robert Dempsey

robertwdempsey.com

robertwdempsey

rdempsey

rdempsey



# District Data Labs

Blog Posts | Webinars | Workshops | Incubator | Research Lab

Learning Continuum

“Those who drank coffee instead of alcohol began the day alert and stimulated, rather than relaxed and mildly inebriated, and the quality and quantity of their work improved. … Western Europe began to emerge from an alcoholic haze that had lasted for centuries.”

–Steven Johnson, *The Invention of Air*

# Doing All Things In SQL

Makes Panda sad and confused

# Each New Thing You Learn

Leads to another new thing to learn, and another, and…

# So Many Things

1. Which predictive modeling technique to use
2. How to get the data into a format for modeling
3. How to ensure the “right” data is being used
4. How to feed the data into the model
5. How to validate the model results
6. How to save the model to use in production
7. How to implement the model in production and apply it to new observations
8. How to save the new predictions
9. How to ensure, over time, that the model is correctly predicting outcomes
10. How to later update the model with new training data

I'm definitely going to need coffee for this shit.

# Choose Your Model

# Model Selection

- How much data do you have?
- Are you predicting a category? A quantity?
- Do you have labeled data?
- Do you know the number of categories?
- How much data do you have?

# Regression

- Used for estimating the relationships among variables

- Use when:
  - Predicting a quantity
  - More than 50 samples

# Classification

- Used to answer “what is this object”
- Use when:
  - Predicting a category
  - Have labeled data

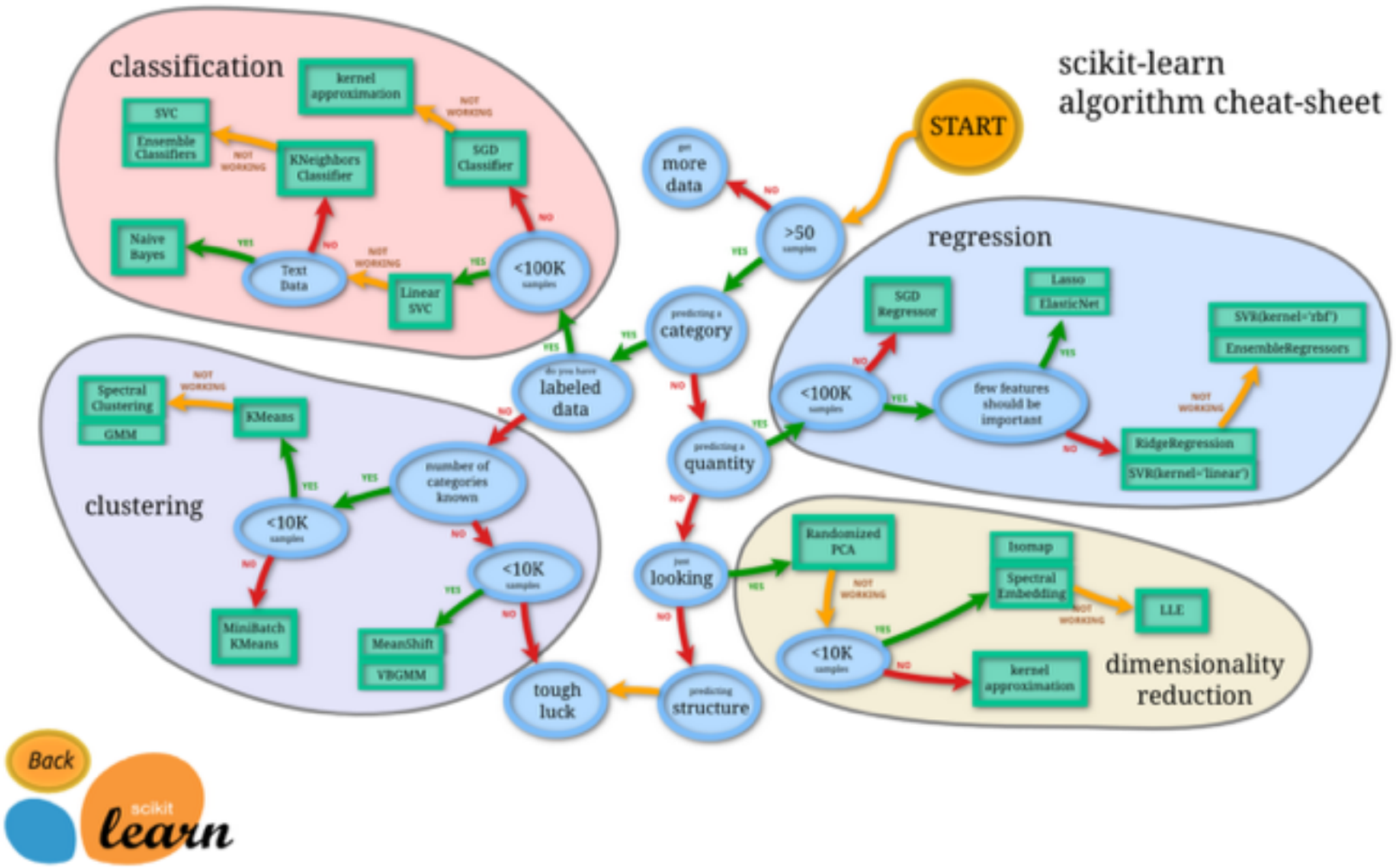
# Clustering

- Used to group similar objects
- Use when:
  - Predicting a category
  - Don’t have labeled data
  - Number of categories is known or unknown
  - Have more than 50 samples

# Dimensionality Reduction

- Process for reducing the number of random variables under consideration (feature selection and feature extraction)
- Use when:
  - Not predicting a category or a quantity
  - Just looking around

# Model Selection




http://scikit-learn.org/stable/tutorial/machine_learning_map/

# Format Thine Data

# Format The Data

- Pandas FTW!
- Use the map() function to convert any text to a number
- Fill in any missing values
- Split the data into features (the data) and targets (the outcome to predict) using .values on the DataFrame

# map()

```
def update_failure_explanations(type):
    if type == 'dob':
        return 0
    elif type == 'name':
        return 1
    elif type == 'ssn dob name':
        return 2
    elif type == 'ssn':
        return 3
    elif type == 'ssn name':
        return 4
    elif type == 'ssn dob':
        return 5
    elif type == 'dob name':
        return 6
```

# Fill In Missing Values

```
df.my_field.fillna('Missing', inplace=True)
```

```
df.fillna(0, inplace=True)
```

# Split the Data

1. Create a matrix of values

```
t_data = raw_data.iloc[:,0:22].values
```

2. Create a matrix of targets

```
t_targets = raw_data['verified'].values
```

# Get the (Right) Data

# Get The Right Data

- This is called “Feature selection”
- Univariate feature selection
  - SelectKBest removes all but the k highest scoring features
  - SelectPercentile removes all but a user-specified highest scoring percentage of features using common univariate statistical tests for each feature: false positive rate
  - SelectFpr, false discovery rate SelectFdr, or family wise error SelectFwe.
  - GenericUnivariateSelect allows to perform univariate feature selection with a configurable strategy.

http://scikit-learn.org/stable/modules/feature_selection.html

# Feed Your Model

# Build the Model

```
from sklearn import linear_model
logClassifier = linear_model.LogisticRegression(C=1,
                                                random_state=111)
```

# Model Parameters

C: inverse of regularization strength; must be a positive float. The smaller the C the simpler the model (it will try to use the strongest features).*

random_state: pseudo-random number generator state used for random sampling.

* You don't always want a simpler model

# Train the Model

```
from sklearn import cross_validation
X_train, X_test, y_train, y_test = cross_validation.train_test_split(the_data,
                                                                     the_targets,
                                                                     cv=12,
                                                                     test_size=0.20,
                                                                     random_state=111)

logClassifier.fit(X_train, y_train)
```

http://scikit-learn.org/stable/modules/cross_validation.html

# Cross Validation

A model validation technique for assessing how the results of a statistical analysis will generalize to an independent data set.

Helps avoid **overfitting**: occurs when a statistical model describes random error or noise instead of the underlying relationship.

https://en.wikipedia.org/wiki/Cross-validation_(statistics)

https://en.wikipedia.org/wiki/Overfitting

# Cross Validation

X_train: 80% of the features

Y_train: 80% of the labels

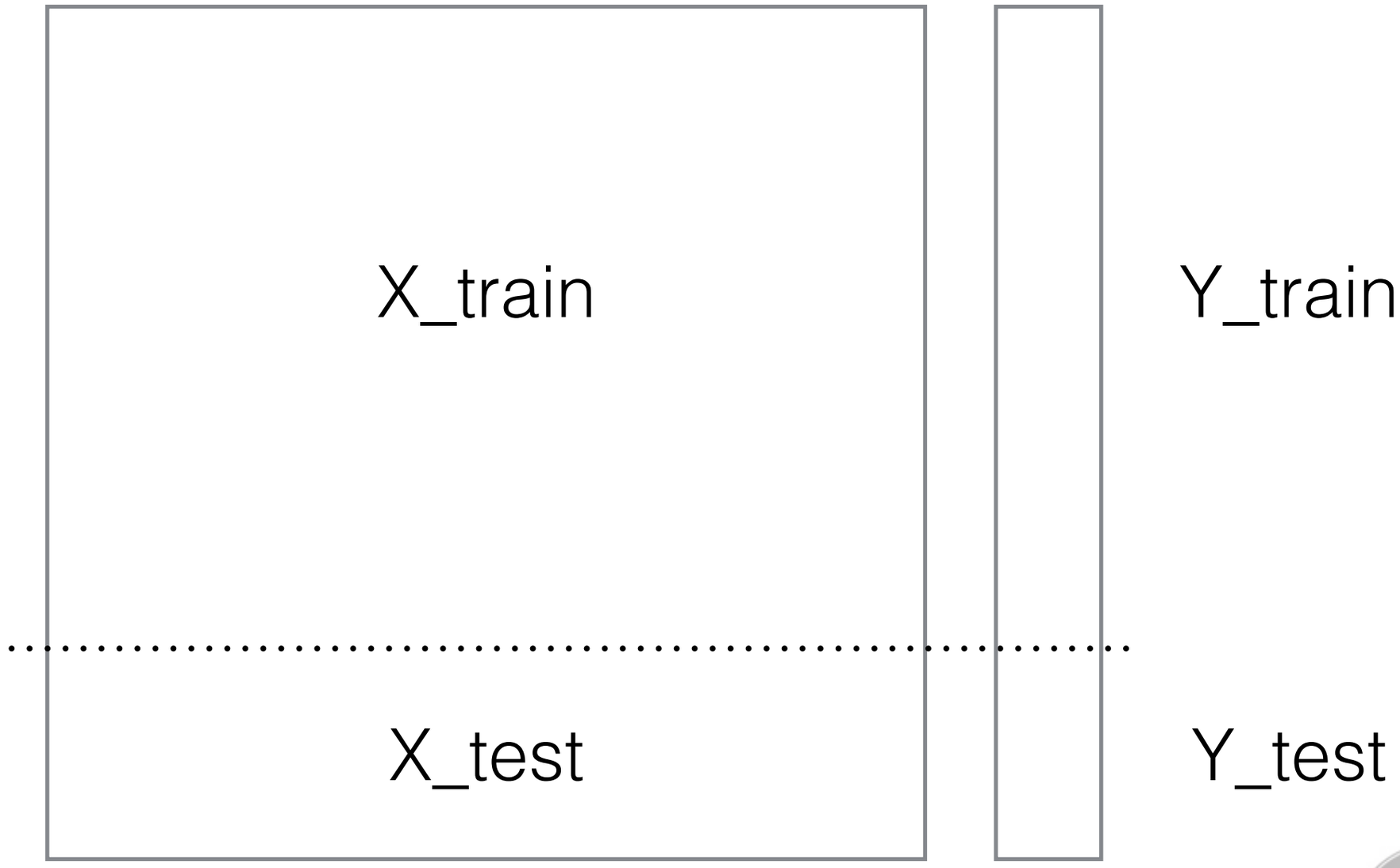
X_test: 20% of the features

Y_test: 20% of the labels

https://en.wikipedia.org/wiki/Cross-validation_(statistics)

https://en.wikipedia.org/wiki/Overfitting

# Cross Validation



# Train the Model

```
from sklearn import cross_validation
X_train, X_test, y_train, y_test = cross_validation.train_test_split(the_data,
                                                                     the_targets,
                                                                     cv=12,
                                                                     test_size=0.20,
                                                                     random_state=111)

logClassifier.fit(X_train, y_train)
```

http://scikit-learn.org/stable/modules/cross_validation.html

# train_test_split

Split arrays or matrices into random train and test subsets.

http://scikit-learn.org/stable/modules/cross_validation.html

# train_test_split Params

arrays: sequence of indexables with same length / shape[0]

cv: 12-fold cross validation

test_size: the proportion of the dataset to include in the test split

random_state: pseudo-random number generator state used for random sampling

http://scikit-learn.org/stable/modules/cross_validation.html

# Fit the Model

```
from sklearn import cross_validation
X_train, X_test, y_train, y_test = cross_validation.train_test_split(the_data,
                                                                     the_targets,
                                                                     cv=12,
                                                                     test_size=0.20,
                                                                     random_state=111)
```

```
logClassifier.fit(X_train, y_train)
```

http://scikit-learn.org/stable/modules/cross_validation.html

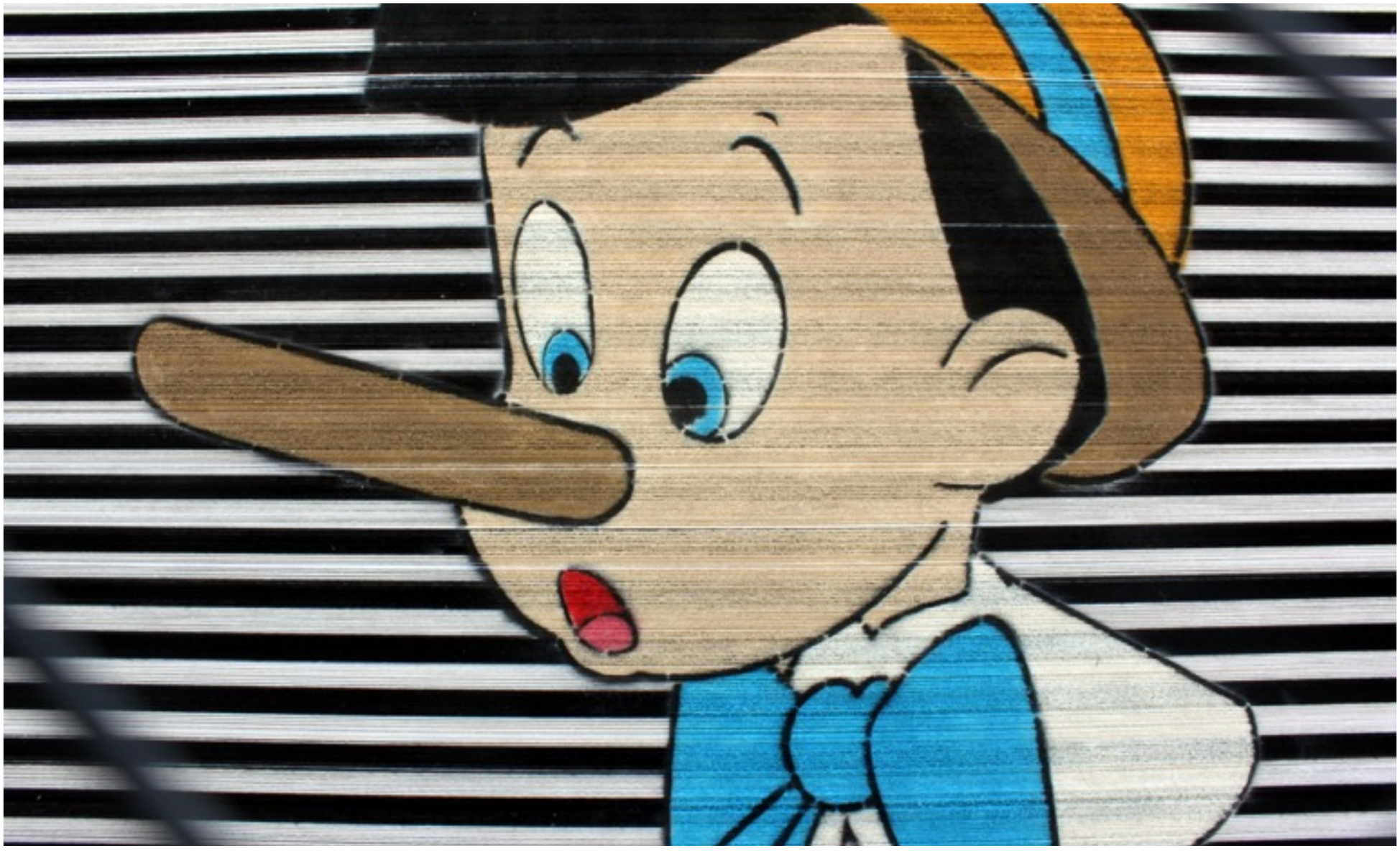
# fit

Performs the training of the model.

X: training data

y: target values

http://scikit-learn.org/stable/modules/cross_validation.html



# Validate That!

# Validation

1. Accuracy Score

```
from sklearn import metrics
metrics.accuracy_score(y_test, predicted)
```

2. Confusion Matrix

```
metrics.confusion_matrix(y_test, predicted)
```

http://scikit-learn.org/stable/modules/cross_validation.html


# Save Your Model

# Save the Model

Pickle it!

```
import pickle
model_file = "/lr_classifier_09.29.15.dat"
pickle.dump(logClassifier, open(model_file, "wb"))
```

Did it work?

```
logClassifier2 = pickle.load(open(model, "rb"))
print(logClassifier2)
```

https://docs.python.org/3/library/pickle.html

# Ship It

# Implement in Production

- Clean the data the same way you did for the model
  - Feature mappings
  - Column re-ordering
- Create a function that returns the prediction
  - Deserialize the model from the file you created
  - Feed the model the data in the same order
  - Call .predict() and get your answer

# Example

```
def verify_record(record_scores):

    # Reload the trained model

    tif = "models/t_lr_classifier_07.28.15.dat"
    log_classifier = pickle.load(open(tcf, "rb"))


    # Return the prediction

    return log_classifier.predict(record_scores)[0]
```

# DEMO!

# Save The Predictions

# Save Your Predictions

As you would any other piece of data

# (Keep) Getting it Right

# Unleash the minion army!

… or get more creative


AN UPDATE IS AVAILABLE FOR YOUR COMPUTER
COOL, MORE FREE STUFF!
NOT AGAIN!
OOH, ONLY $99!
linux
windows
mac
stickycomics.com

# Update It

# Be Smart

Train it again, but with validated predictions

# Review

# Step Review

1. Select a predictive modeling technique to use
2. Get the data into a format for modeling
3. Ensure the “right” data is being used
4. Feed the data into the model
5. Validate the model results

# Step Review

6. Save the model to use in production

7. Implement the model in production and apply it to new observations

8. Save the new predictions

9. Ensure the model is correctly predicting outcomes over time

10. Update the model with new training data

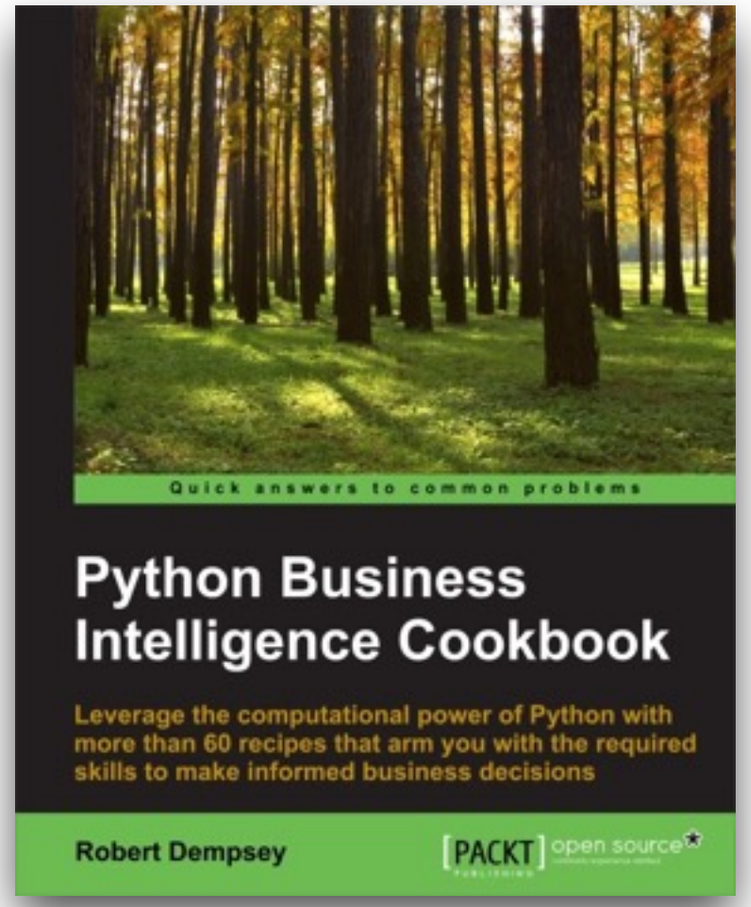
# Robert Dempsey

robertwdempsey.com

robertwdempsey

rdempsey

rdempsey



# Image Credits

- Format: https://www.flickr.com/photos/zaqography/3835692243/
- Get right data: https://www.flickr.com/photos/encouragement/14759554777/
- Feed: https://www.flickr.com/photos/glutnix/4291194/
- Validate: https://www.flickr.com/photos/lord-jim/16827236591/
- Save: http://www.cnn.com/2015/09/13/living/candice-swanepoel-victorias-secret-model-falls-feat/
- Ship It: https://www.flickr.com/photos/oneeighteen/15492277272/
- Save Predictions: https://www.flickr.com/photos/eelssej_/486414113/
- Get it right: https://www.flickr.com/photos/clickflashphotos/3402287993/
- Update it: https://www.flickr.com/photos/dullhunk/5497202855/
- Review: https://www.flickr.com/photos/pluggedmind/10714537023/