Practical Machine Learning

Goals

High Level:

- Overview of a workflow
- Be aware of important Do's and Don'ts
- Some useful algorithms

In Practice:

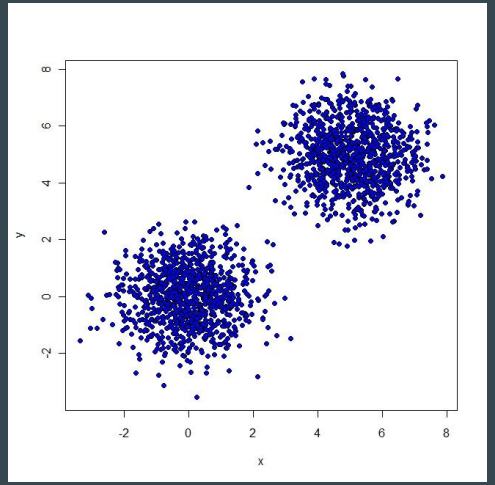
- Where to start
- What works, what doesn't
- Stuff you can google / talk more about

Types of ML Tasks

Supervised

- Majority of ML projects
- Build model to predict Value or Label based on some known features
- Logistic Regression, Neural Nets, Dec Trees

Unsupervised



- Algorithmically exploring data to find underlying groups/clusters
- Harder to validate
- DBSCAN, K-Means

A Workflow Overview

- Data Exploration
- Feature engineering
- Modeling
- Evaluation

Grabbing Data and Exploring

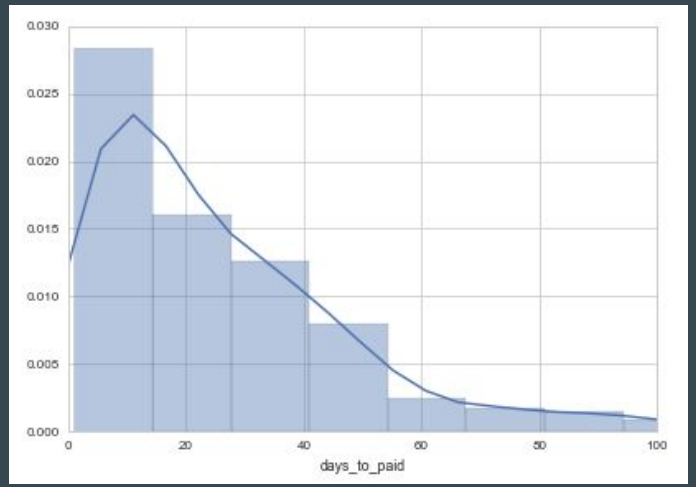
```
query = """
SELECT DISTINCT ON (administrations.admin_id)
projects.*, administrations.admin_id, administrations.project_id,
administrations.created_at AS admin_created_at
FROM projects
JOIN administrations
ON projects.id = administrations.project_id
ORDER BY administrations.admin_id, projects.created_at, projects.id
"""

proj_admin = pd.read_sql_query(query, conn)
proj_admin = proj_admin[proj_admin.project_id != 14] # this is a test entry in the DB
```

```
# get days to set up fee
projects['days_to_setup_fee'] = projects.loc[projects['set_up_fee_paid_at'].notnull(),
                                             'set_up_fee_paid_at'].sub(projects['created_at'], axis=0)
projects['days_to_setup_fee'] = projects.days_to_setup_fee.apply(get_days)

# pledged amount as function of goal
projects['pledged_perc'] = (projects.pledged_amount / projects.goal) * 100

projects = projects[projects.ended_at.notnull()]
projects = projects[projects.project_id != 2812]
```

Sanity Checks

Is my data weird?

- Histograms
- Impossible values
- Unique values
- IsNan, IsNA
- Missing data



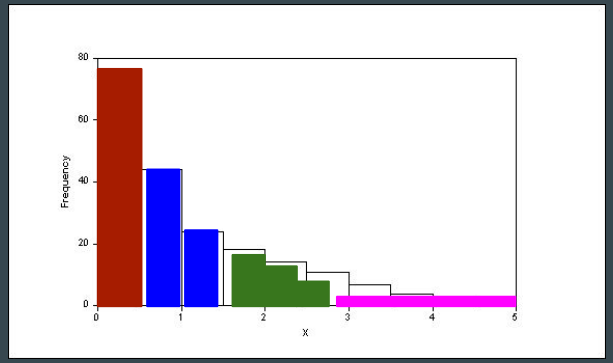
Pre-model Hypotheses

- Plotting data by outcome variable
- Scatterplot matrix
- Are values missing at random?
  - How can I replace these intelligently?
  - Do I need to replace these?

Feature Engineering

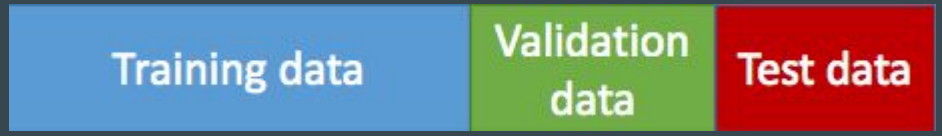
Multilevel categorical features:

Very Skewed data:



- Clustering Data
- Feature reduction
  - PCA
- Correlated Features?
- Treat IsNA as a category
- Feature Selection
  - Forward Selection
  - Backward Elimination
- NORMALIZATION

Split Data!



OR
Cross Validation

Test data should never be used for model selection!

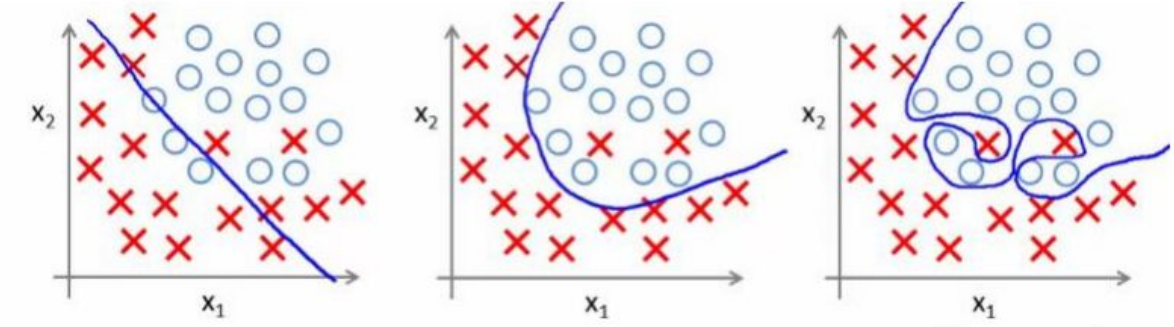
Splitting Considerations and Tips

- Stratification?
  - Is one class over-represented?
- Random Sampling
  - But use a random seed!
- 80/20 splits
- Leave-one out cross validation

```
# get positive and negative cases for stratification
paying = projects.loc[projects.is_paying, :]
notpaying = projects.loc[~projects.is_paying, :]
print(len(paying.index))
print(len(notpaying.index))

samp = np.random.rand(len(paying)) <= 0.8
payTrain = paying[samp]
payTest = paying[~samp]

samp = np.random.rand(len(notpaying)) <= 0.8
notpayTrain = notpaying[samp]
notpayTest = notpaying[~samp]
```

The Bias-Variance Trade Off

Underfitting
(high bias)



Overfitting
(high variance)

Modeling in SKLearn

```
from sklearn.ensemble import RandomForestClassifier
```

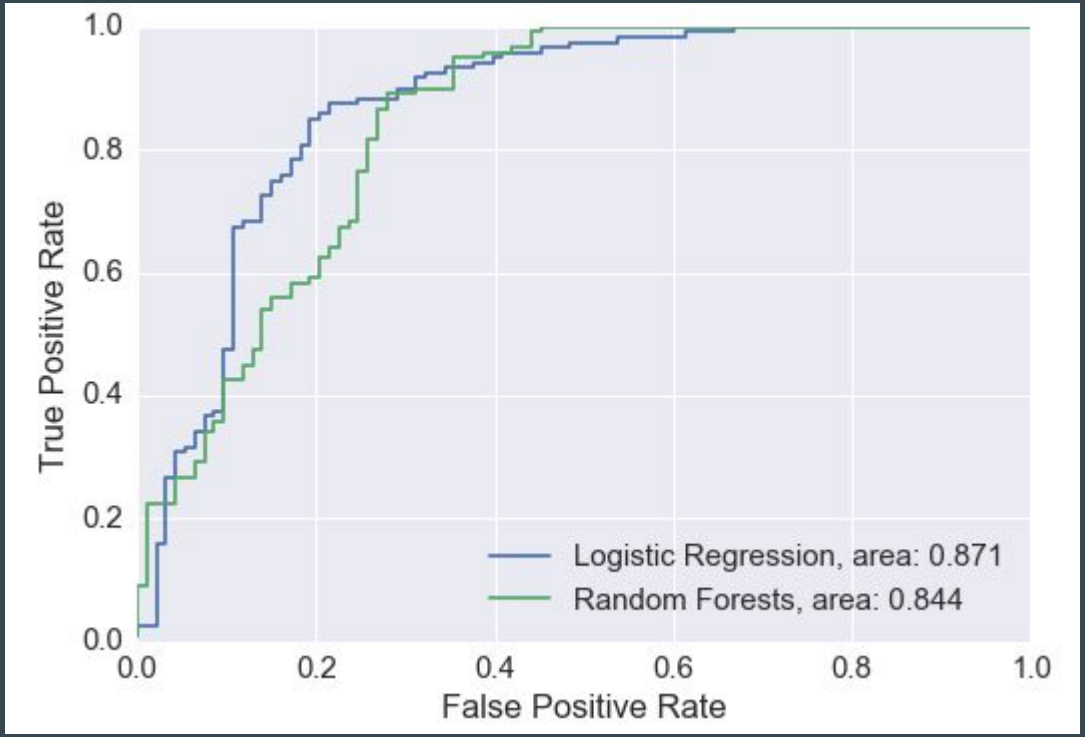
```
rfc = RandomForestClassifier(n_estimators=100, min_samples_leaf=1, random_state=0)
rfc.fit(train_X, train_Y)

pred = rfc.predict(test_X)
print("accuracy: " + str(sum(pred == test_Y)/float(len(test_Y))))

(tp,fp,tn,fn) = get_pn_rates(pred, test_Y)
print("precision: " + str(float(tp)/(tp+fp)))
print("recall: " + str(float(tp)/(tp+fn)))
```

```
accuracy: 0.804222648752
precision: 0.434367541766
recall: 0.85046728972
```

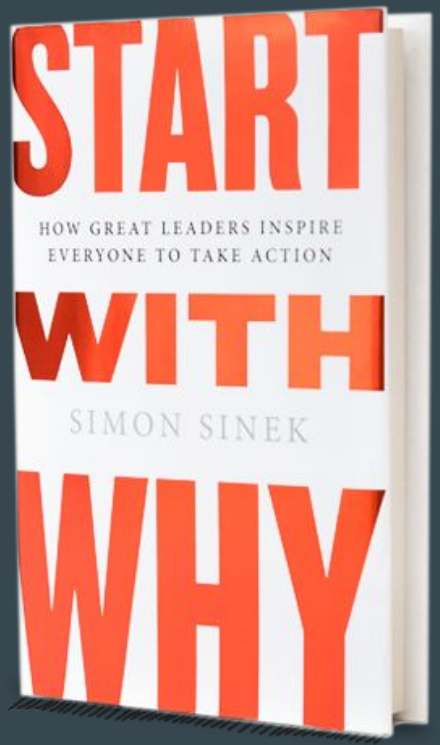
Evaluation

- Know WHY you're evaluating with a particular metric
- Accuracy (correct # classifications, mean squared error)
- Precision: TP / (TP+FP)
  - Is the model just blindly labeling things as true?
  - Very deceptive in unbalanced datasets
- Recall: TP / (TP+FN)
  - Is the model capturing positive cases?
- Would you want to maximize Precision or Recall when making a classifier for identifying biopsy samples as cancerous?

Receiver-Operating Characteristic



Some Practical Advice



- Know why you're using a model before you try it
  - Strengths? Weaknesses? Be able to speak to tradeoffs
  - Does the model take a long time to train? To classify? Memory requirements?
- Try several models, and understand why they're performing differently
- Start very simple, and build up
  - This will help you start with the why
- Finding unique tweaks on simple algorithms can be as impressive as making a complicated algorithm work
- Speak about models in plain english. Practice explaining on each other by teaching!

Quick Brainstorming Exercise

- Let's name some different algorithms, and talk about their strengths and weaknesses