Machine learning pipelines and an end to end project

[1]Church and Duncan Group,

[2]Information School, UC Berkeley

[3]School of Informatics, Computing and Engineering, Indiana University

EMAIL: James_DOT_Shanahan_AT_gmail_DOT_com

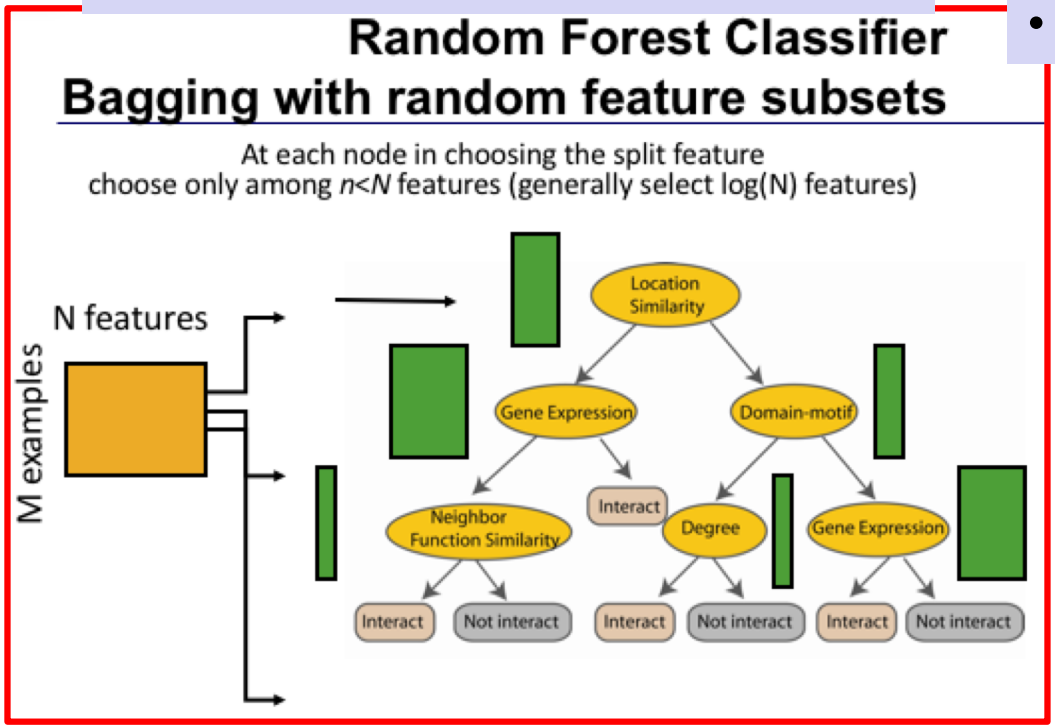
Outline

- **Introduction**
- **End to end ML project in SKLearn**
  - Problem definition and data
  - EDA
  - SKLearn Pipelines for prepping data
  - Full learning pipeline
  - Finetune model:
    - GridSearch versus Random
- **Which model is better? Significance Tests**
- **Summary**

Fine-tune your model

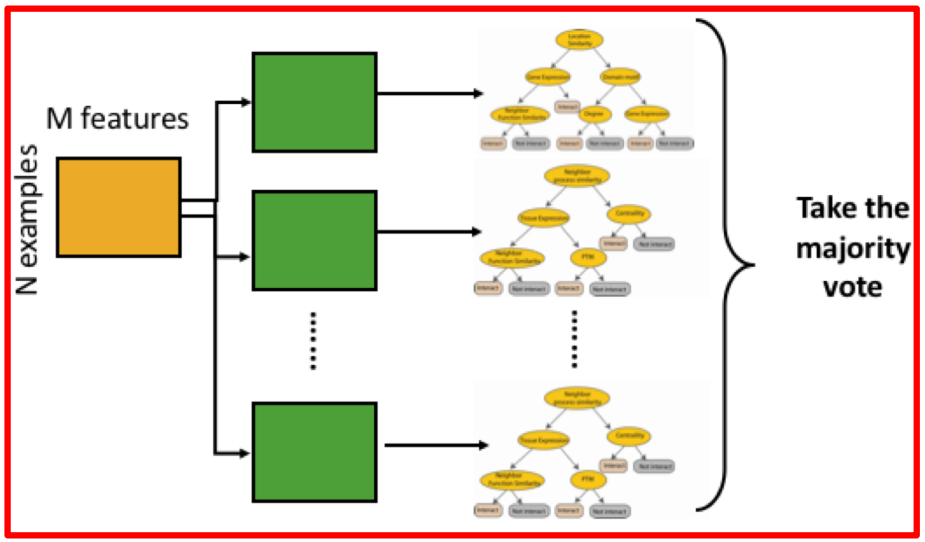
- **GridSearchCV**
- **RandomSearchCV**

Learn one decision tree in the forest



Prediction using the forest of Decision Tree

- Use majority class for classification
- Use average prediction for regression



Random forest of decision trees

A random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and use averaging to improve the predictive accuracy and control over-fitting. The sub-sample size is always the same as the original input sample size but the samples are drawn with replacement if bootstrap=True (default).

Read more in the User Guide.

Parameters:

n_estimators : integer, optional (default=10)

The number of trees in the forest.

criterion : string, optional (default="gini")

The function to measure the quality of a split. Supported criteria are "gini" for the Gini impurity and "entropy" for the information gain. Note: this parameter is tree-specific.

max_features : int, float, string or None, optional (default="auto")

The number of features to consider when looking for the best split:

- If int, then consider max_features features at each split.
- If float, then max_features is a percentage and int(max_features * n_features) features are considered at each split.
- If "auto", then max_features=sqrt(n_features).
- If "sqrt", then max_features=sqrt(n_features) (same as "auto").
- If "log2", then max_features=log2(n_features).
- If None, then max_features=n_features.

Note: the search for a split does not stop until at least one valid partition of the node samples is found, even if it requires to effectively inspect more than `max_features` features.

max_depth : integer or None, optional (default=None)

The maximum depth of the tree. If None, then nodes are expanded until all leaves are pure or until all leaves contain less than min_samples_split samples.

Random Forest: Hyperparameters

In the following the `param_grid` tells Scikit-Learn to first evaluate all 3 × 4 = 12 combinations of `n_estimators` and `max_features` hyperparameter values specified in the first dict (don't worry about what these hyperparameters mean for now; they will be explained in Chapter 7), then try all 2 × 3 = 6 combinations of hyperparameter values in the second dict, but this time with the `bootstrap` hyperparameter set to `False` instead of `True` (which is the default value for this hyperparameter).

All in all, the grid search will explore 12 + 6 = 18 combinations of `RandomForestRegressor` hyperparameter values, and it will train each model five times (since we are using five-fold cross validation). In other words, all in all, there will be 18 × 5 = 90 rounds of training!

```
    # try 12 (3×4) combinations of hyperparameters
    {'n_estimators': [3, 10, 30], 'max_features': [2, 4, 6, 8]},
    # then try 6 (2×3) combinations with bootstrap set as False
    {'bootstrap': [False], 'n_estimators': [3, 10], 'max_features': [2, 3, 4]},
  ]
```

- **3, or 10, 30 trees in ensemble**

bootstrap : boolean, optional (default=True)

Whether bootstrap samples are used when building trees.

Tune RandomForrests:

```
from sklearn.model_selection import GridSearchCV

param_grid = [
    # try 12 (3×4) combinations of hyperparameters
    {'n_estimators': [3, 10, 30], 'max_features': [2, 4, 6, 8]},
    # then try 6 (2×3) combinations with bootstrap set as False
    {'bootstrap': [False], 'n_estimators': [3, 10], 'max_features': [2, 3, 4]},
  ]

forest_reg = RandomForestRegressor(random_state=42)
# train across 5 folds, that's a total of (12+6)*5=90 rounds of training
grid_search = GridSearchCV(forest_reg, param_grid, cv=5,
                           scoring='neg_mean_squared_error')
grid_search.fit(housing_prepared, housing_labels)
```

```
GridSearchCV(cv=5, error_score='raise',
       estimator=RandomForestRegressor(bootstrap=True, criterion='mse', max_depth=None
           max_features='auto', max_leaf_nodes=None,
           min_impurity_split=1e-07, min_samples_leaf=1,
           min_samples_split=2, min_weight_fraction_leaf=0.0,
           n_estimators=10, n_jobs=1, oob_score=False, random_state=42,
           verbose=0, warm_start=False),
       fit_params={}, iid=True, n_jobs=1,
       param_grid=[{'n_estimators': [3, 10, 30], 'max_features': [2, 4, 6, 8]}, {'boot
s': [3, 10], 'max_features': [2, 3, 4]}],
       pre_dispatch='2*n_jobs', refit=True, return_train_score=True,
       scoring='neg_mean_squared_error', verbose=0)
```

```
cvres = grid_search.cv_results_
for mean_score, params in zip(cvres["mean_test_score"], cvres["params"]):
    print(np.sqrt(-mean_score), params)
```

```
63825.0479302 {'max_features': 2, 'n_estimators': 3}
55643.8429091 {'max_features': 2, 'n_estimators': 10}
53380.6566859 {'max_features': 2, 'n_estimators': 30}
60959.1388585 {'max_features': 4, 'n_estimators': 3}
52740.5841667 {'max_features': 4, 'n_estimators': 10}
50374.1421461 {'max_features': 4, 'n_estimators': 30}
58661.2866462 {'max_features': 6, 'n_estimators': 3}
52009.9739798 {'max_features': 6, 'n_estimators': 10}
50154.1177737 {'max_features': 6, 'n_estimators': 30}
57865.3616801 {'max_features': 8, 'n_estimators': 3}
51730.0755087 {'max_features': 8, 'n_estimators': 10}
49694.8514333 {'max_features': 8, 'n_estimators': 30}
62874.4073931 {'bootstrap': False, 'max_features': 2, 'n_estimators': 3}
54561.9398157 {'bootstrap': False, 'max_features': 2, 'n_estimators': 10}
59416.6463145 {'bootstrap': False, 'max_features': 3, 'n_estimators': 3}
52660.245911 {'bootstrap': False, 'max_features': 3, 'n_estimators': 10}
57490.0168279 {'bootstrap': False, 'max_features': 4, 'n_estimators': 3}
51093.9059428 {'bootstrap': False, 'max_features': 4, 'n_estimators': 10}
```

Outline

- **Introduction**
- **End to end ML project in SKLearn**
  - Problem definition and data
  - EDA
  - SKLearn Pipelines for prepping data
  - Full learning pipeline
  - Finetune model:
    - GridSearch versus RandomSearch
- **Which model is better? Significance Tests**
- **Summary**

Stop experiment

Cross-validation strategy

I like to do **coarse -> fine** cross-validation in stages

First stage: only a few epochs to get rough idea of what params work
Second stage: longer running time, finer search
… (repeat as necessary)

Tip for detecting explosions in the solver:
If the cost is ever > 3 * original cost, break out early

65

Sample in log space (for 5 epochs)

For example: run coarse search for 5 epochs

```
max_count = 100
for count in xrange(max_count):
      reg = 10**uniform(-5, 5)
      lr = 10**uniform(-3, -6)

      trainer = ClassifierTrainer()
      model = init_two_layer_model(32*32*3, 50,
      trainer = ClassifierTrainer()
      best_model_local, stats = trainer.train(X_train, y_train, X_val, y_val,
                                              model, two_layer_net,
                                              num_epochs=5, reg=reg,
                                              update='momentum', learning_rate_decay=0.9,
                                              sample_batches = True, batch_size = 100,
                                              learning rate=lr, verbose=False)
```

note it's best to optimize in log space!

- 10**(-5:5) [1]
- 1e-05 1e-04 1e-03 1e-02 1e-01 1e+00 1e+01 1e+02 1e+03 1e+04 1e+05

Validation Accuracy	Learning Rate	Regularization coefficient
val_acc: 0.412000,	lr: 1.405206e-04,	reg: 4.793564e-01, (1 / 100)
val_acc: 0.214000,	lr: 7.231888e-06,	reg: 2.321281e-04, (2 / 100)
val_acc: 0.208000,	lr: 2.119571e-06,	reg: 8.011857e+01, (3 / 100)
val_acc: 0.196000,	lr: 1.551131e-05,	reg: 4.374936e-05, (4 / 100)
val_acc: 0.079000,	lr: 1.753300e-05,	reg: 1.200424e+03, (5 / 100)
val_acc: 0.223000,	lr: 4.215128e-05,	reg: 4.196174e+01, (6 / 100)
val_acc: 0.441000,	lr: 1.750259e-04,	reg: 2.110807e-04, (7 / 100)
val_acc: 0.241000,	lr: 6.749231e-05,	reg: 4.226413e+01, (8 / 100)
val_acc: 0.482000,	lr: 4.296863e-04,	reg: 6.642555e-01, (9 / 100)
val_acc: 0.079000,	lr: 5.401602e-06,	reg: 1.599828e+04, (10 / 100)
val_acc: 0.154000,	lr: 1.618508e-06,	reg: 4.925252e-01, (11 / 100)

nice

Multiplicative e interaction

Now run finer search...

```
max_count = 100
for count in xrange(max_count):
      reg = 10**uniform(-5, 5)
      lr = 10**uniform(-3, -6)

      trainer = ClassifierTrainer()
      model = init two layer model(3
```

adjust range

```
max_count = 100
for count in xrange(max_count):
      reg = 10**uniform(-4, 0)
      lr = 10**uniform(-3, -4)
```

Validation Accuracy | **Learning Rate** | **Regularization coefficient**

```
val_acc: 0.492000, lr: 2.279484e-04, reg: 9.991345e-04, (1 / 100)
val_acc: 0.512000, lr: 8.680827e-04, reg: 1.349727e-02, (2 / 100)
val_acc: 0.461000, lr: 1.028377e-04, reg: 1.220193e-02, (3 / 100)
val_acc: 0.460000, lr: 1.113730e-04, reg: 5.244309e-02, (4 / 100)
val_acc: 0.498000, lr: 9.477776e-04, reg: 2.001293e-03, (5 / 100)
val_acc: 0.469000, lr: 1.484369e-04, reg: 4.328313e-01, (6 / 100)
val_acc: 0.522000, lr: 5.586261e-04, reg: 2.312685e-04, (7 / 100)
val_acc: 0.530000, lr: 5.808183e-04, reg: 8.259964e-02, (8 / 100)
val_acc: 0.489000, lr: 1.979168e-04, reg: 1.010889e-04, (9 / 100)
val_acc: 0.490000, lr: 2.036031e-04, reg: 2.406271e-03, (10 / 100)
val_acc: 0.475000, lr: 2.021162e-04, reg: 2.287807e-01, (11 / 100)
val_acc: 0.460000, lr: 1.135527e-04, reg: 3.905040e-02, (12 / 100)
val_acc: 0.515000, lr: 6.947668e-04, reg: 1.562808e-02, (13 / 100)
val_acc: 0.531000, lr: 9.471549e-04, reg: 1.433895e-03, (14 / 100)
val_acc: 0.509000, lr: 3.140888e-04, reg: 2.857518e-01, (15 / 100)
val_acc: 0.514000, lr: 6.438349e-04, reg: 3.033781e-01, (16 / 100)
val_acc: 0.502000, lr: 3.921784e-04, reg: 2.707126e-04, (17 / 100)
val_acc: 0.509000, lr: 9.752279e-04, reg: 2.850865e-03, (18 / 100)
val_acc: 0.500000, lr: 2.412048e-04, reg: 4.997821e-04, (19 / 100)
val_acc: 0.466000, lr: 1.319314e-04, reg: 1.189915e-02, (20 / 100)
val_acc: 0.516000, lr: 8.039527e-04, reg: 1.528291e-02, (21 / 100)
```

53% - relatively good for a 2-layer neural net with 50 hidden neurons.

Sample randomly

Now run finer search...

```
max_count = 100
for count in xrange(max_count):
      reg = 10**uniform(-5, 5)
      lr = 10**uniform(-3, -6)
```

adjust range

```
max_count = 100
for count in xrange(max_count):
      reg = 10**uniform(-4, 0)
      lr = 10**uniform(-3, -4)
```

```
val_acc: 0.527000, lr: 5.340517e-04, reg: 4.097824e-01, (0 / 100)
val_acc: 0.492000, lr: 2.279484e-04, reg: 9.991345e-04, (1 / 100)
val_acc: 0.512000, lr: 8.680827e-04, reg: 1.349727e-02, (2 / 100)
val_acc: 0.461000, lr: 1.028377e-04, reg: 1.220193e-02, (3 / 100)
val_acc: 0.460000, lr: 1.113730e-04, reg: 5.244309e-02, (4 / 100)
val_acc: 0.498000, lr: 9.477776e-04, reg: 2.001293e-03, (5 / 100)
val_acc: 0.469000, lr: 1.484369e-04, reg: 4.328313e-01, (6 / 100)
val_acc: 0.522000, lr: 5.586261e-04, reg: 2.312685e-04, (7 / 100)
val_acc: 0.530000, lr: 5.808183e-04, reg: 8.259964e-02, (8 / 100)
val_acc: 0.489000, lr: 1.979168e-04, reg: 1.010889e-04, (9 / 100)
val_acc: 0.490000, lr: 2.036031e-04, reg: 2.406271e-03, (10 / 100)
val_acc: 0.475000, lr: 2.021162e-04, reg: 2.287807e-01, (11 / 100)
val_acc: 0.460000, lr: 1.135527e-04, reg: 3.905040e-02, (12 / 100)
val_acc: 0.515000, lr: 6.947668e-04, reg: 1.562808e-02, (13 / 100)
val_acc: 0.531000, lr: 9.471549e-04, reg: 1.433895e-03, (14 / 100)
val_acc: 0.509000, lr: 3.140888e-04, reg: 2.857518e-01, (15 / 100)
val_acc: 0.514000, lr: 6.438349e-04, reg: 3.033781e-01, (16 / 100)
val_acc: 0.502000, lr: 3.921784e-04, reg: 2.707126e-04, (17 / 100)
val_acc: 0.509000, lr: 9.752279e-04, reg: 2.850865e-03, (18 / 100)
val_acc: 0.500000, lr: 2.412048e-04, reg: 4.997821e-04, (19 / 100)
val_acc: 0.466000, lr: 1.319314e-04, reg: 1.189915e-02, (20 / 100)
val_acc: 0.516000, lr: 8.039527e-04, reg: 1.528291e-02, (21 / 100)
```

53% - relatively good for a 2-layer neural net with 50 hidden neurons.

But this best cross-validation result is worrying.

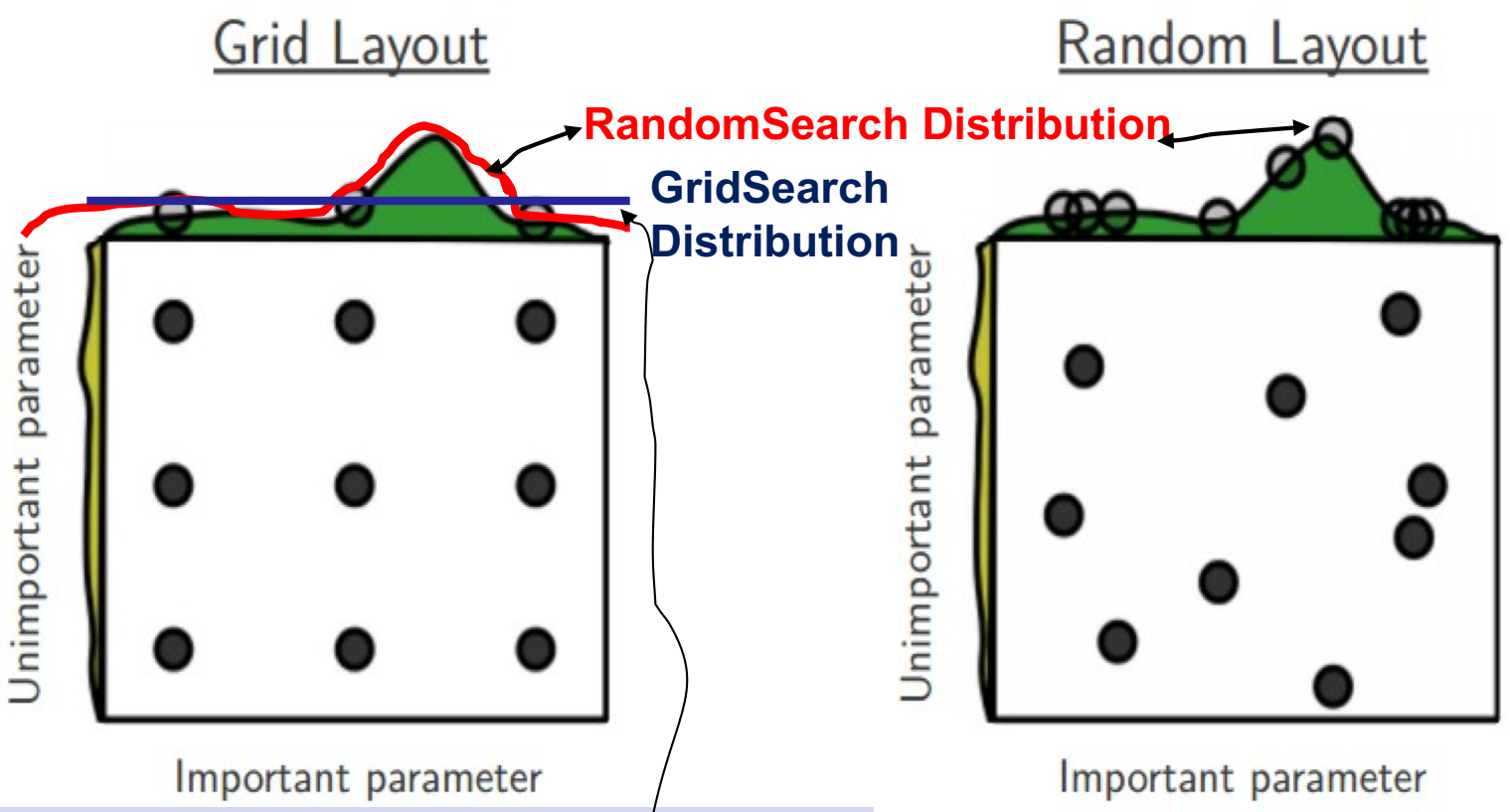
But this best cross-validation result is worrying. Why?

At the boundary of grid search [lr=0.0009, reg=0.001]
Lr in [1e-6, 1e-3] Reg in [1e-5, 1e5]

Our random sampling is better than grid search

Random Search vs. Grid Search

Performance distribution curve: E.g., Accuracy



Notice how grid search only does 9 experiments and misses the important hyperparameter zones

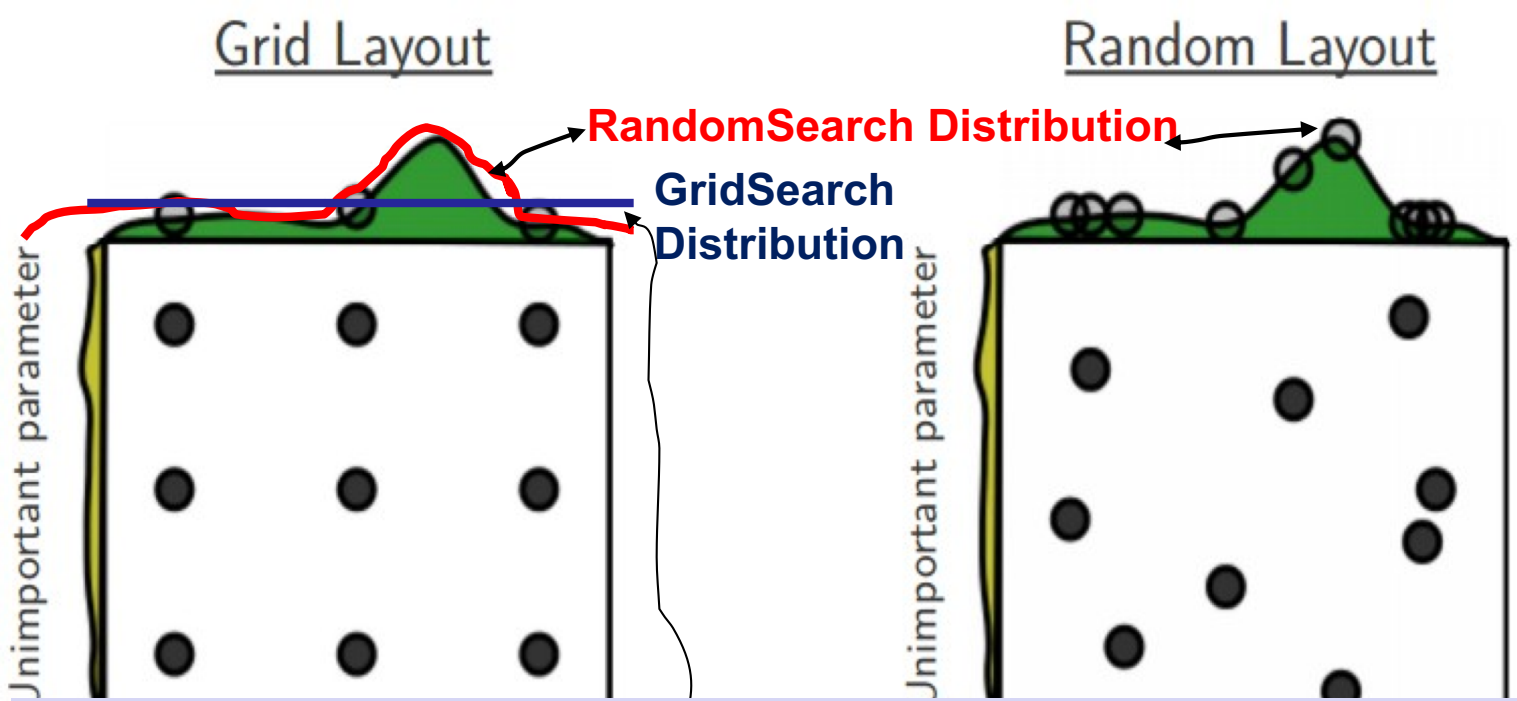
Performance of loss function is more affected by the X Dimension here

[See Random Search for Hyper-Parameter Optimization Bergstra and Bengio, 2012]

Our random sampling is better than grid search

Random Search vs. Grid Search

Performance distribution curve: E.g., Accuracy



Random gives more bang for the buck and approximates the performance distribution curve more closely. Get more samples across the important dimension (as opposed to 3 in Grid search)

important hyperparameter zones

[See Random Search for Hyper-Parameter Optimization Bergstra and Bengio, 2012]

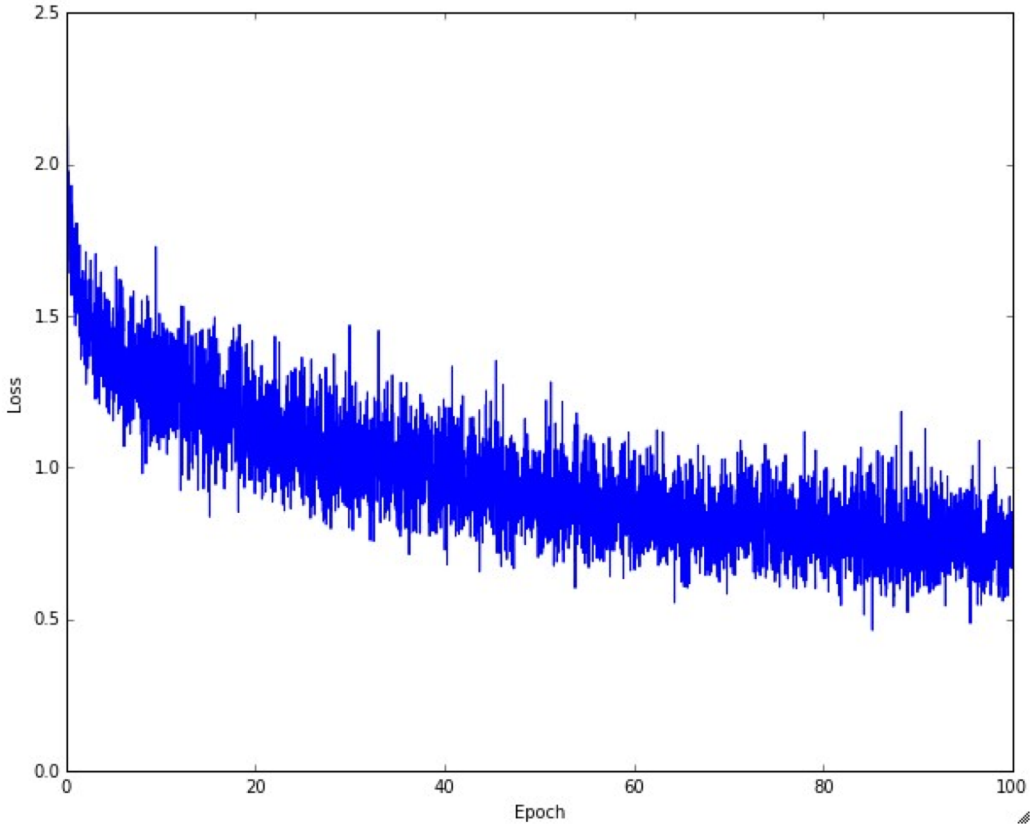
Loss, Acc curves provide insight

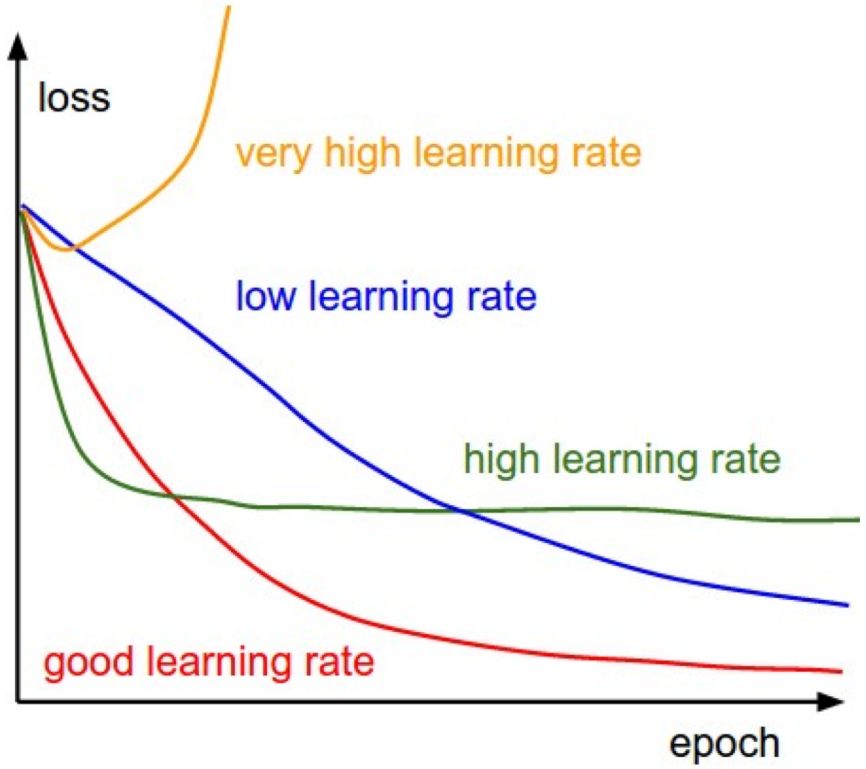
Hyperparameters to play with:

- model architecture
- learning rate, its decay schedule, update type
- regularization

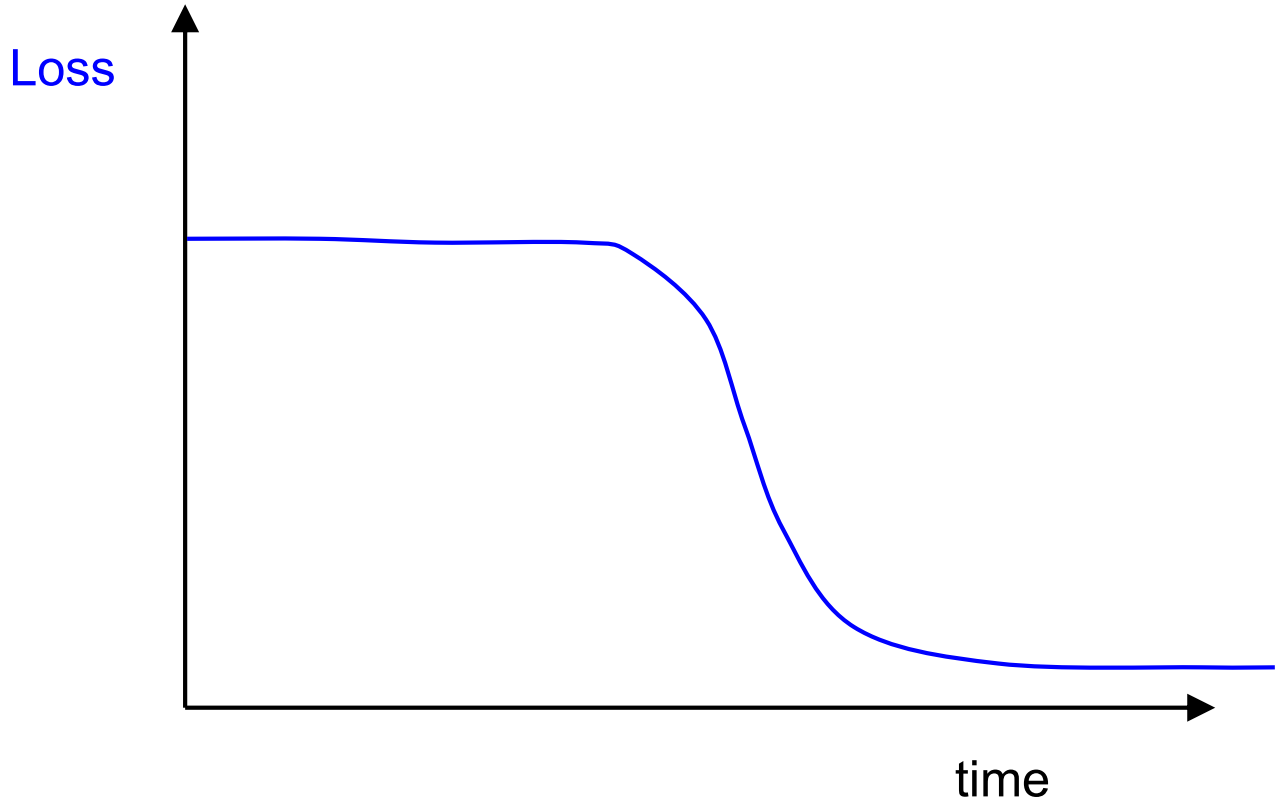
Monitor and visualize the loss curve

Monitor and visualize the loss curve

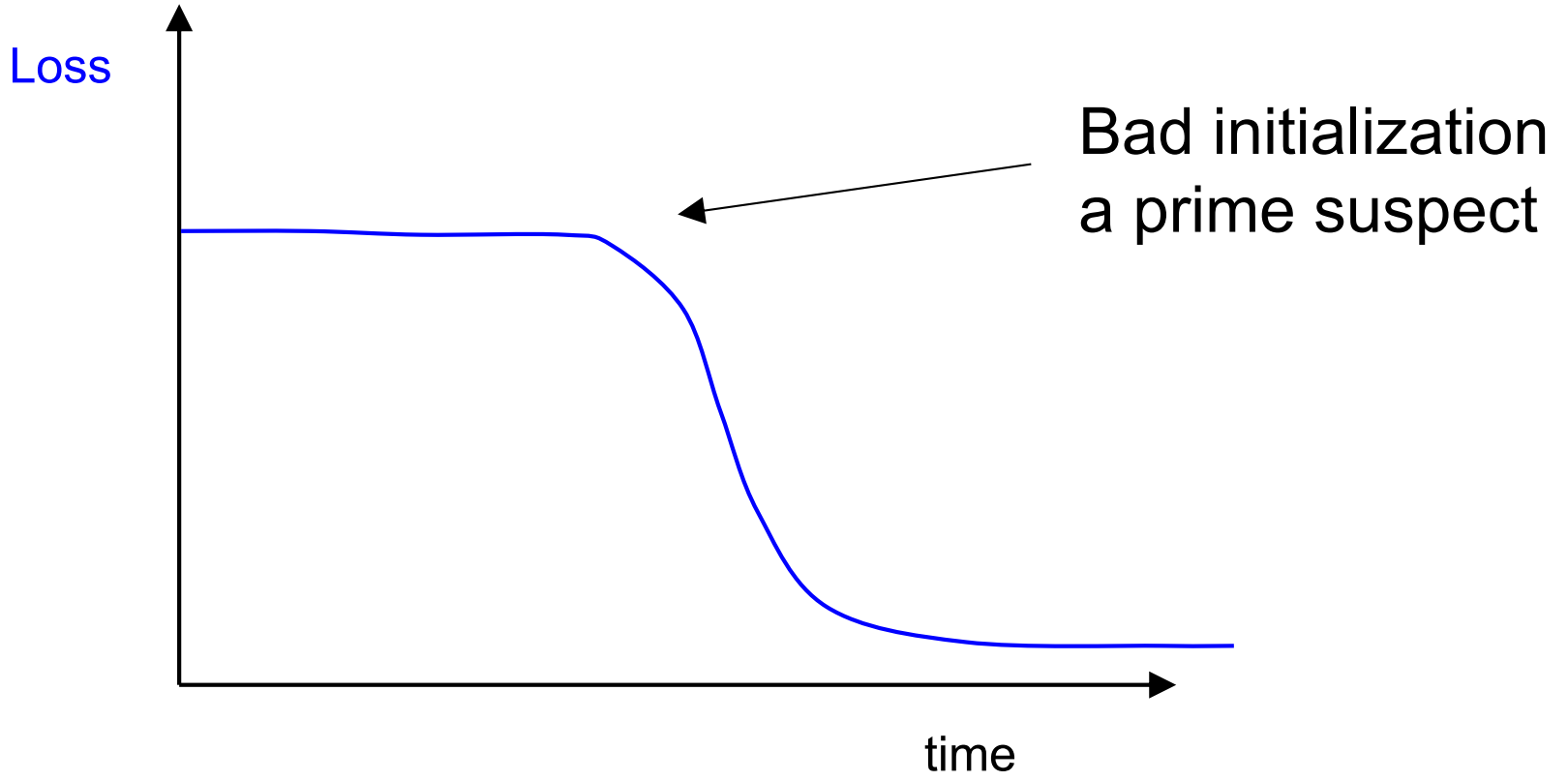




What's wrong here?

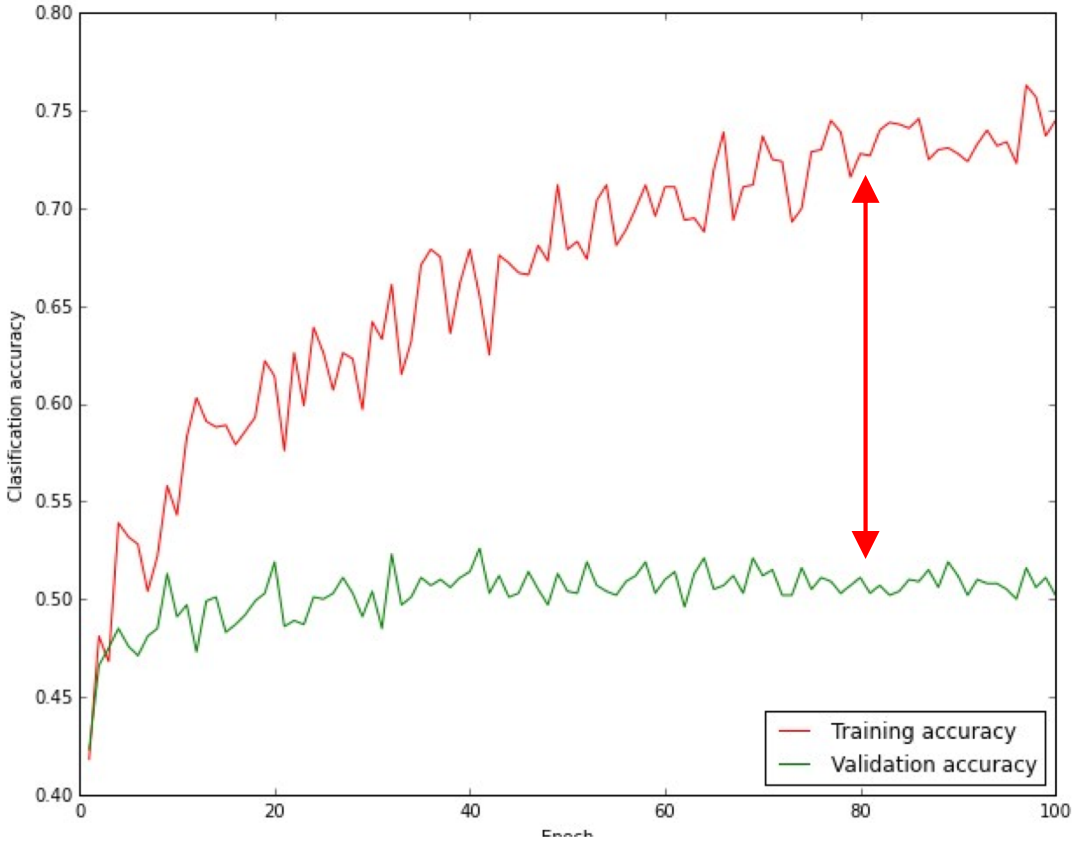


Bad initialization a prime suspect



Monitor underfitting vs. overfitting

Monitor and visualize the accuracy:



big gap = overfitting

=> increase regularization strength?

no gap

=> increase model capacity?

weight updates / weight magnitudes

Track the ratio of weight updates / weight magnitudes:

```
# assume parameter vector W and its gradient vector dW
param_scale = np.linalg.norm(W.ravel())
update = -learning_rate*dW # simple SGD update
update_scale = np.linalg.norm(update.ravel())
W += update # the actual update
print update_scale / param_scale # want ~1e-3
```

ratio between the values and updates: ~ 0.0002 / 0.02 = 0.01 (about okay)
want this to be somewhere around 0.001 or so

Weights are unit Gaussian so the updates should be small so

RandomizedSearchCV

- **When the hyperparameter search space is large, it is often preferable to use RandomizedSearchCV**
- **This class can be used in much the same way as the GridSearchCV class, but instead of trying out all possible combinations, it evaluates a given number of random combinations by selecting a random value for each hyperparameter at every iteration.**
- **This approach has two main benefits:**
  - If you let the randomized search run for, say, 1,000 iterations, this approach will explore 1,000 different values for each hyperparameter (instead of just a few values per hyperparameter with the grid search approach).
  - You have more control over the computing budget you want to allocate to hyperparameter search, simply by setting the number of iterations.

RandomSearchCV: 10 experiments

```
from sklearn.model_selection import RandomizedSearchCV
from scipy.stats import randint

param_distribs = {
        'n_estimators': randint(low=1, high=200),
        'max_features': randint(low=1, high=8),
    }

forest_reg = RandomForestRegressor(random_state=42)
rnd_search = RandomizedSearchCV(forest_reg, param_distributions=param_distribs,
                                n_iter=10, cv=5, scoring='neg_mean_squared_error',
rnd_search.fit(housing_prepared, housing_labels)
```

```
cvres = rnd_search.cv_results_
for mean_score, params in zip(cvres["mean_test_score"], cvres["params"]):
    print(np.sqrt(-mean_score), params)
```

```
49147.1524172 {'max_features': 7, 'n_estimators': 180}
51396.8768969 {'max_features': 5, 'n_estimators': 15}
50798.3025423 {'max_features': 3, 'n_estimators': 72}
50840.744514 {'max_features': 5, 'n_estimators': 21}
49276.1753033 {'max_features': 7, 'n_estimators': 122}
50776.7360494 {'max_features': 3, 'n_estimators': 75}
50682.7075546 {'max_features': 3, 'n_estimators': 88}
49612.1525305 {'max_features': 5, 'n_estimators': 100}
50472.6107336 {'max_features': 3, 'n_estimators': 150}
64458.2538503 {'max_features': 5, 'n_estimators': 2}
```

RandomSearchCV: 10 experiments

```
from sklearn.model_selection import RandomizedSearchCV
from scipy.stats import randint

param_distribs = {
        'n_estimators': randint(low=1, high=200),
        'max_features': randint(low=1, high=8),
    }

forest_reg = RandomForestRegressor(random_state=42)
rnd
rnd_search.fit(housing_prepared, housing_labels)
```

64458.2538503 {'max_features': 5, 'n_estimators': 2}

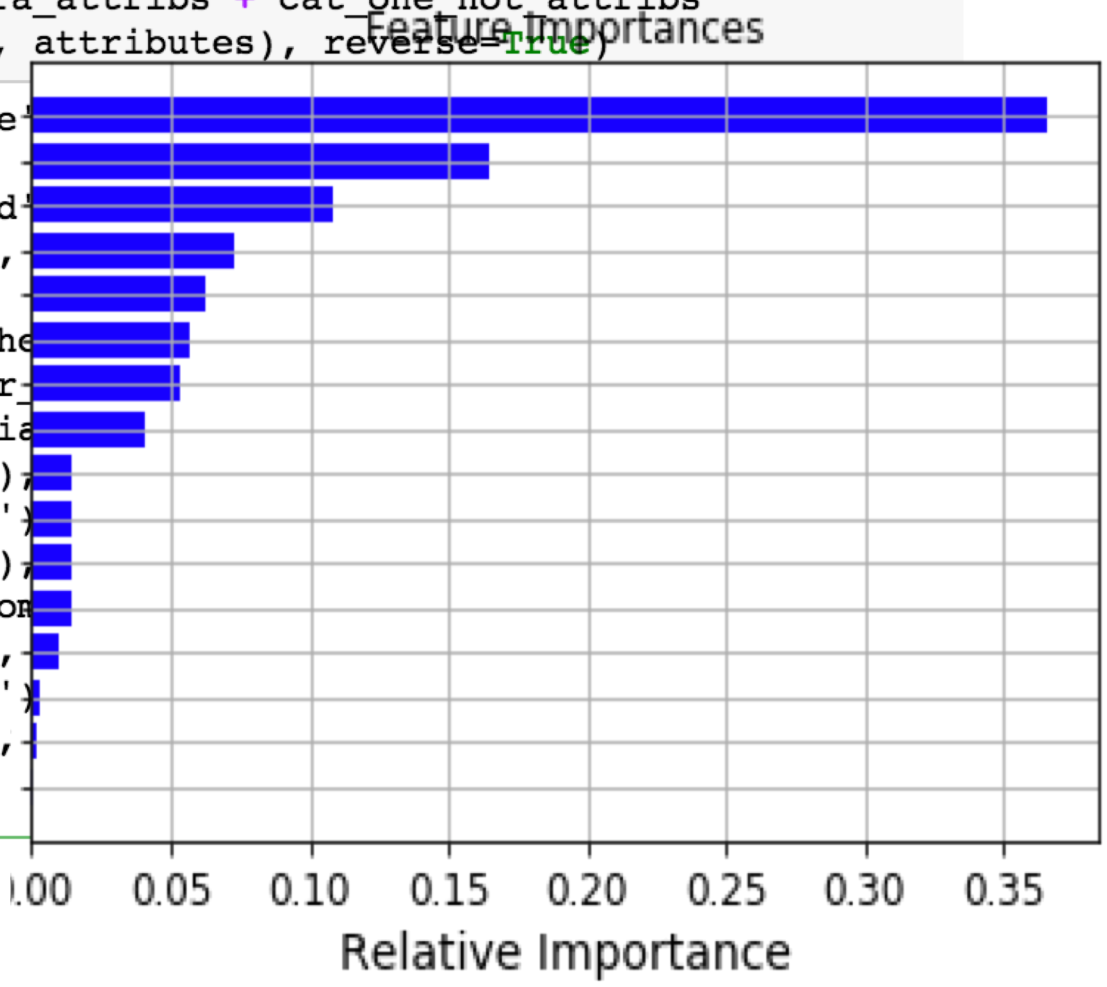
```
cvres = rnd_search.cv_results_
for mean_score, params in zip(cvres["mean_test_score"], cvres["params"]):
    print(np.sqrt(-mean_score), params)
```

```
49147.1524172 {'max_features': 7, 'n_estimators': 180}
51396.8768969 {'max_features': 5, 'n_estimators': 15}
50798.3025423 {'max_features': 3, 'n_estimators': 72}
50840.744514 {'max_features': 5, 'n_estimators': 21}
49276.1753033 {'max_features': 7, 'n_estimators': 122}
50776.7360494 {'max_features': 3, 'n_estimators': 75}
50682.7075546 {'max_features': 3, 'n_estimators': 88}
49612.1525305 {'max_features': 5, 'n_estimators': 100}
50472.6107336 {'max_features': 3, 'n_estimators': 150}
64458.2538503 {'max_features': 5, 'n_estimators': 2}
```

Analyze the Best Models and Their Errors

```
1 extra_attribs = ["rooms_per_hhold", "pop_per_hhold", "bedrooms_per_room"]
2 cat_one_hot_attribs = list(encoder.classes_)
3 attributes = num_attribs + extra_attribs + cat_one_hot_attribs
4 sorted(zip(feature_importances, attributes), reverse=True)
```

```
[(0.36615898061813418, 'median_income'
 (0.16478099356159051, 'INLAND'),
 (0.10879295677551573, 'pop_per_hhold'
 (0.073344235516012421, 'longitude'),
 (0.062909070482620302, 'latitude'),
 (0.056419179181954007, 'rooms_per_hhe
 (0.053351077347675809, 'bedrooms_per_
 (0.041143798478729635, 'housing_media
 (0.014874280890402767, 'population'),
 (0.014672685420543237, 'total_rooms'),
 (0.014257599323407807, 'households'),
 (0.014106483453584102, 'total_bedroom
 (0.010311488326303787, '<1H OCEAN'),
 (0.0028564746373201579, 'NEAR OCEAN'),
 (0.0019604155994780701, 'NEAR BAY'),
 (6.0280386727365991e-05, 'ISLAND')]
```



Outline

- **Introduction**
- **End to end ML project in SKLearn**
  - Problem definition and data
  - EDA
  - SKLearn Pipelines for prepping data
  - Full learning pipeline
  - Finetune model:
    - GridSearch versus Random
- **Which model is better? Significance Tests**
- **Summary**

Run test on heldout testset

- **run your full_pipeline to transform the data (call transform(), not fit_transform()!), and evaluate the final model on the test set**

```
]:  1 final_model = grid_search.best_estimator_
    2
    3 X_test = strat_test_set.drop("median_house_value", axis=1)
    4 y_test = strat_test_set["median_house_value"].copy()
    5
    6 X_test_prepared = full_pipeline.transform(X_test)
    7 final_predictions = final_model.predict(X_test_prepared)
    8
    9 final_mse = mean_squared_error(y_test, final_predictions)
   10 final_rmse = np.sqrt(final_mse)
```

```
]: final_rmse
```

```
]: 47766.003966433083
```

Run test on heldout testset

- run your full_pipeline to transform the data (call transform(), not fit_transform()!), and evaluate the final model on the test set

```
]:  1 final_model = grid_search.best_estimator_
    2
    3 X_test = strat_test_set.drop("median_house_value", axis=1)
    4 y_test = strat_test_set["median_house_value"].copy()
    5
    6 X_test_prepared = full_pipeline.transform(X_test)
    7 final_predictions = final_model.predict(X_test_prepared)
    8
    9 final_mse = mean_squared_error(y_test, final_predictions)
   10 final_rmse = np.sqrt(final_mse)
```

```
]:
```

```
]: 4
```

Are the results significantly better than SVMs?

Do a t.test?

Don’t fit the heldout test set!

- The performance will usually be slightly worse than what you measured using crossvalidation if you did a lot of hyperparameter tuning (because your system ends up fine-tuned to perform well on the validation data, and will likely not perform as well on unknown datasets).
- It is not the case in this example, but when this happens you must resist the temptation to tweak the hyperparameters to make the numbers look good on the test set; the improvements would be unlikely to generalize to new data.

Deploy and AB Test

- Deploy (engineering)
- AB Test if possible
- Dashboards
- Success failure analysis
- Maintain

Next steps for you!

- **Try a competition website such as KAGGLE http://kaggle.com/ :**
  - you will have a dataset to play with, a clear goal, and people to share the experience

Outline

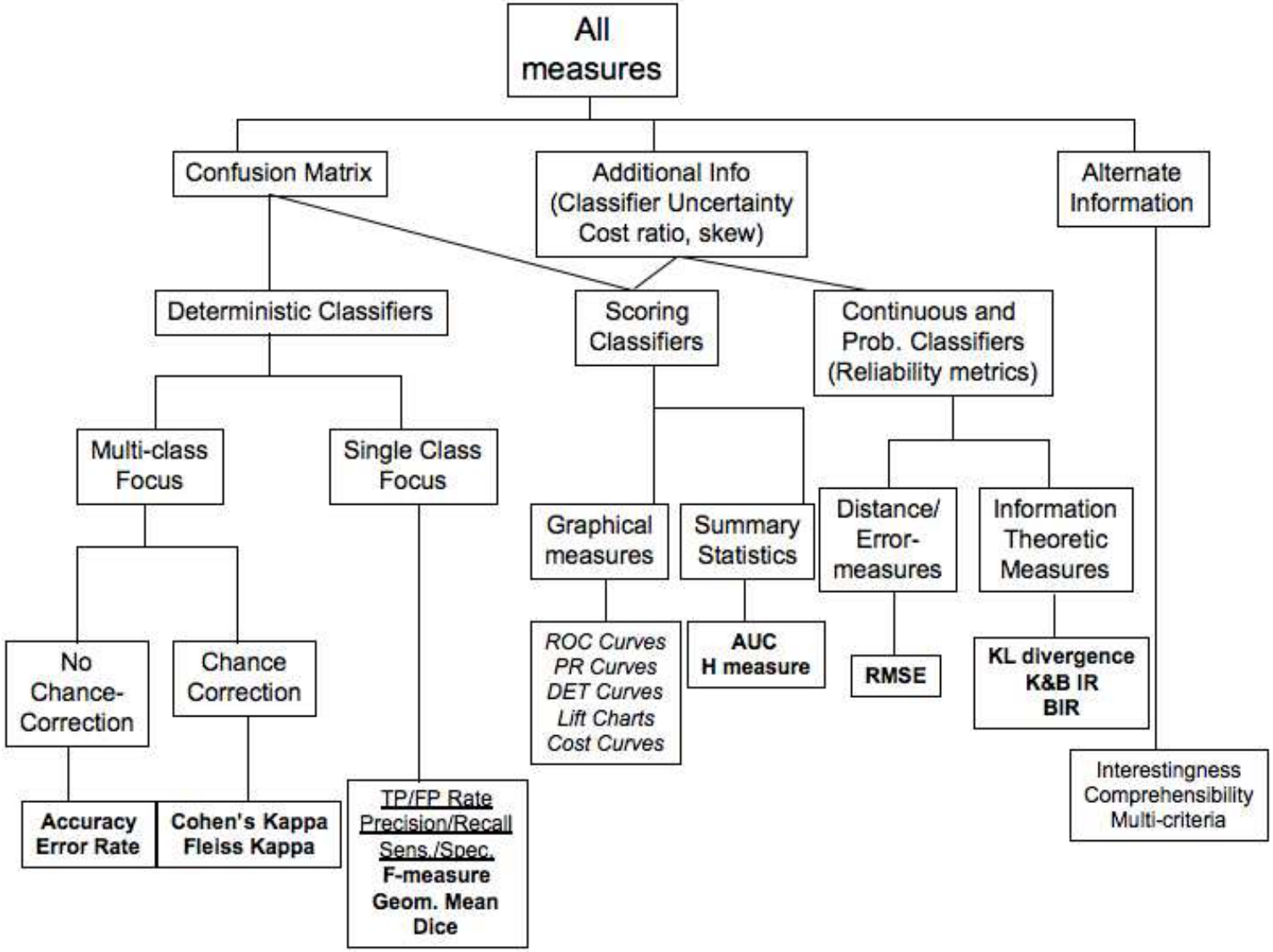
- **Introduction**
- **End to end ML project in SKLearn**
  - Problem definition and data
  - EDA
  - SKLearn Pipelines for prepping data
  - Full learning pipeline
  - Finetune model:
    - GridSearch versus Random
- **Which model is better? Significance Tests**
- **Summary**

In practice: Labs and in the wild

- **Is system A better that system B in a laboratory setting**
  - Paired
    - (one-sided [sure challenger is better] versus 2-sided [not sure if A and B are different; generally go with 2-sided)
  - Not paired
    - Is system A better that system B in the wild
      - one-sided vs two-sided

- **Scenario 1: Lab based experiments**
  - System A versus System B
- **Scenario 2: Deployed in the wild**
  - Incumbent system versus challenger (that you built!)
  - AB Testing (Control versus treatment)

Overview of Performance Measures
Focus on classification



Regression Performance Measures

- **Frame as a regression problem so use these performance measures**
- **Root Mean Square Error (RMSE).**

$$\mathrm{RMSE}(\mathbf{X}, h) = \sqrt{\frac{1}{m}\sum_{i=1}^{m}\left(h\left(\mathbf{x}^{(i)}\right) - y^{(i)}\right)^2}$$

- **Mean Absolute Error (also called the Average Absolute Deviation**

$$\mathrm{MAE}(\mathbf{X}, h) = \frac{1}{m}\sum_{i=1}^{m}\left|h\left(\mathbf{x}^{(i)}\right) - y^{(i)}\right|$$

- **Mean absolute percentage error (MAPE)**

$$\mathrm{MAPE}\ (\mathbf{X}, h) = \frac{100}{m}\sum_{i=1}^{m}\frac{\left|h\left(\mathbf{x}^{(i)}\right) - y^{(i)}\right|}{y^{(i)}}$$

Model evaluation in SKLearn : quantifying the quality of predictions

Scoring	Function	Comment
Classification		
'accuracy'	`metrics.accuracy_score`	
'average_precision'	`metrics.average_precision_score`	
'f1'	`metrics.f1_score`	for binary targets
'f1_micro'	`metrics.f1_score`	micro-averaged
'f1_macro'	`metrics.f1_score`	macro-averaged
'f1_weighted'	`metrics.f1_score`	weighted average
'f1_samples'	`metrics.f1_score`	by multilabel sample
'neg_log_loss'	`metrics.log_loss`	requires `predict_proba` support
'precision' etc.	`metrics.precision_score`	suffixes apply as with 'f1'
'recall' etc.	`metrics.recall_score`	suffixes apply as with 'f1'
'roc_auc'	`metrics.roc_auc_score`	
Clustering		
'adjusted_mutual_info_score'	`metrics.adjusted_mutual_info_score`	
'adjusted_rand_score'	`metrics.adjusted_rand_score`	
'completeness_score'	`metrics.completeness_score`	
'fowlkes_mallows_score'	`metrics.fowlkes_mallows_score`	
'homogeneity_score'	`metrics.homogeneity_score`	
'mutual_info_score'	`metrics.mutual_info_score`	
'normalized_mutual_info_score'	`metrics.normalized_mutual_info_score`	
'v_measure_score'	`metrics.v_measure_score`	
Regression		
'explained_variance'	`metrics.explained_variance_score`	
'neg_mean_absolute_error'	`metrics.mean_absolute_error`	
'neg_mean_squared_error'	`metrics.mean_squared_error`	
'neg_mean_squared_log_error'	`metrics.mean_squared_log_error`	
'neg_median_absolute_error'	`metrics.median_absolute_error`	
'r2'	`metrics.r2_score`	

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

Are the differences in my machine learning pipelines just due to random chance or are there Statistically Significant?

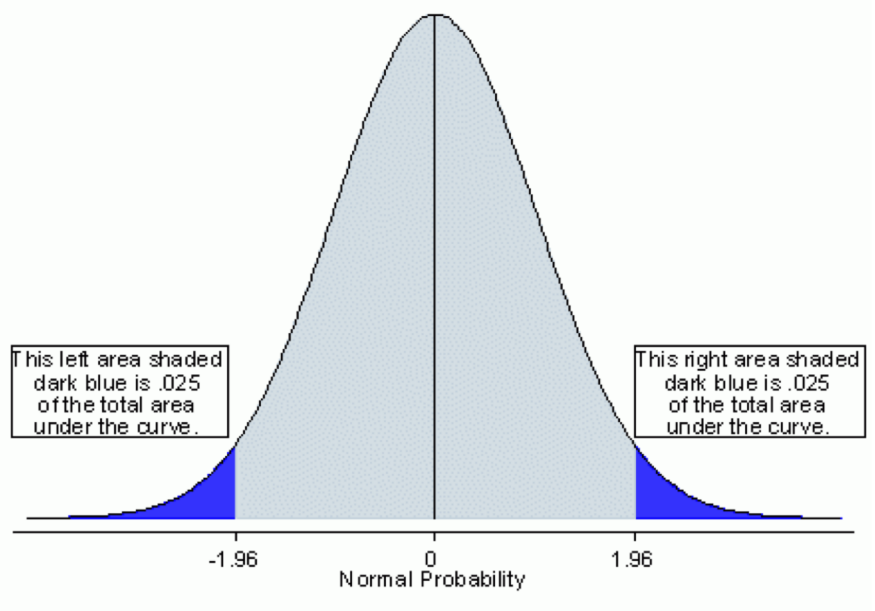
- **Performance metrics alone may not be enough**
  - Performance metrics (such as as misclassification error) allow us to make observations about different machine learning pipeline (preprocessing steps; learning algorithms, hyperparameter settings).
- **The question we ask here is:**
  - can the observed results be attributed to real characteristics of the ML pipeline under scrutiny or are they observed by chance?
  - The purpose of statistical significance testing is to help us gather evidence of the extent to which the results returned by an evaluation metric are representative of the general behaviour of our models be they classifiers or regressors.

Comparing two systems

- **Significance tests**
  - Establish and Examine the H_0, null hypothesis
    - "the two systems being compared have effectively the same predictive characteristics; any difference between them occurred by random chance"
  - Establish H_1: "the two systems being compared are not equal"
- **Most tests consider**
  - Mean, standard deviation
  - Compute a probability, p, H_0 that holds.
  - If p is less than a threshold, reject H_0.
    - 0.05 or 0.01
    - Some controversy about this, generally
      - Say that H_1 holds
        - » Definition depends on 1 or 2 tail test; use 2 tail test by default
        - » H_1: "the two systems being compared are not equal"
        - » There is a significant difference between systems

Significance test errors

- **Type I (set p-value to 0.05)**
  - false positives 1 in 20 one can make a mistake and reject the null hypothesis when the null hypothesis is true
- **Type II. (power of test; typically set power to 0.2)**
  - false negatives
  - Reject H1 when H1 is true



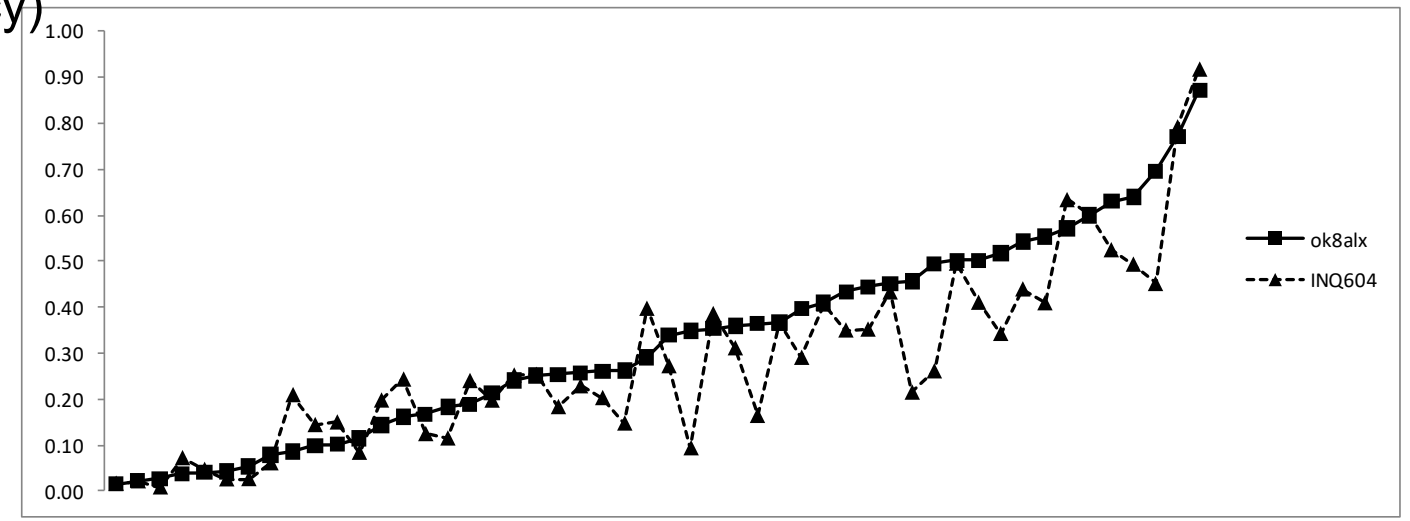
Which type of test?

- **Non-parametric**
  - Sign test
    - Known for type II errors
  - Wilcoxon test
- **Parametric**
  - T-test
  - Known for type I errors
  - Assumptions of data
  - Get it on Excel
    - (paired test for test collections)

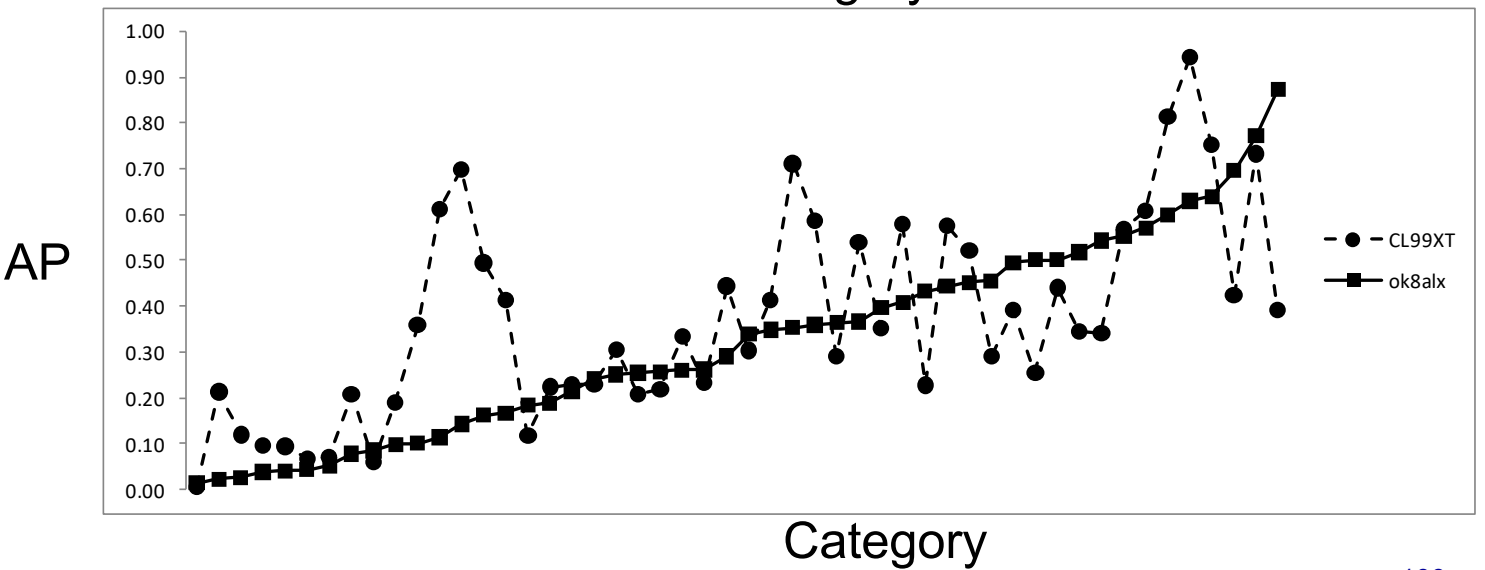
T-test: Control (OK8) versus treatments (INQ; CL9)

Average precision (AP)
(think of this as accuracy)

- **p=0.002**

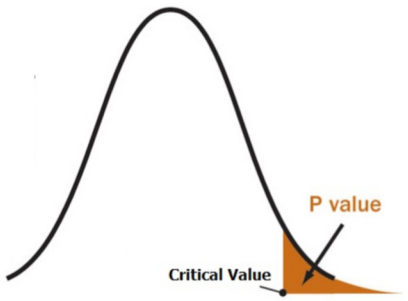


- **p=0.072**



Hypothesis Testing in words

- **Hypothesis testing consists of stating a null hypothesis which usually is the opposite of what we wish to test (for example, classifiers A and B perform equivalently)**
- **We then choose a suitable statistical test and statistic that will be used to reject the null hypothesis.**
  - We also choose a critical region for the statistic to lie in that is extreme enough for the null hypothesis to be rejected.
  - We calculate the observed test statistic from the data and check whether it lies in the critical region. If so, reject the null hypothesis. If not, we fail to reject the null hypothesis, but do not accept it either.
- **Rejecting the null hypothesis gives us some confidence in the belief that our observations did not occur merely by chance.**



A p-value is an area in the tail of a distribution that tells you the odds of a result

significance tests help explain differences

- **A common approach to more fully understanding a difference Δ measured between two ML experiments is to use one or more significance tests.**
- **Significance tests for ML Experiments**
  - STEP 1 set up the following
    - Null hypothesis (H_0)
      - In the context of ML experiments, H0 states that the systems producing the two experimental runs under examination have effectively the same predictive characteristics and that any difference between the runs occurred by random chance.
    - Alternative hypothesis (H_1)
      - This hypothesis states that the two ML systems have different predictive characteristics, leading the experimenter to conclude that a significant difference was observed.
      - The exact nature of H1 depends on whether a one- or a two-tailed test is chosen. (explained later)
  - STEP 2: Set the significance level p, and whether the test should be one-sided or two-sided (good default!)

Reject H_0 ➔ conclude H_1 is true

- The tests estimate the probability p of observing a difference at least as large as Δ given that a so-called null hypothesis (H0) is true.
- The convention when using such tests is that if it is found that p is below a certain threshold — typically either 0.05 or 0.01; this is established before the experiments start — it is concluded that H_0 is unlikely and consequently should be rejected.

- Although it is not universally agreed upon, the common interpretation of rejecting H_0 is to conclude that an alternate hypothesis, H_1 is true.

Significance Tests

A common approach to more fully understanding a difference Δ measured between two runs is to use one or more significance tests. The tests estimate the probability p of observing a difference at least as large as Δ given that a so-called *null hypothesis* (H_0) is true. In the context of IR experiments, H_0 states that the systems producing the two runs under examination have effectively the same retrieval characteristics and that any difference between the runs occurred by random chance. The convention when using such tests is that if it is found that p is below a certain threshold — typically either 0.05 or 0.01 — it is concluded that H_0 is unlikely and consequently should be rejected. Although it is not universally agreed upon, the common interpretation of rejecting H_0 is to conclude that an alternate hypothesis, H_1 is true. This hypothesis states that the two IR systems have different retrieval characteristics, leading the experimenter to conclude that a significant difference was observed. The exact nature of H_1 depends on whether a *one-* or a *two-tailed* test is chosen. This topic is discussed below in Section 5.1.2.

The tests are not infallible and can make errors, which have been classified into Type I and Type II errors. Type I errors are false positives: leading the experimenter to incorrectly reject H_0 and conclude that H_1 is true; Type II errors are false negatives: leading the experimenter to incorrectly conclude that they cannot reject the null hypothesis. In IR parlance, Type I measures the precision of the test, Type II measures its recall. Different significance tests tend to produce a different balance between these two errors. For example, the sign test is known for its high number of Type II errors whereas the t-test is known for producing Type I.

http://marksanderson.org/publications/my_papers/FnTIR.pdf

[Mark Sanderson Book on Evaluation in IR]

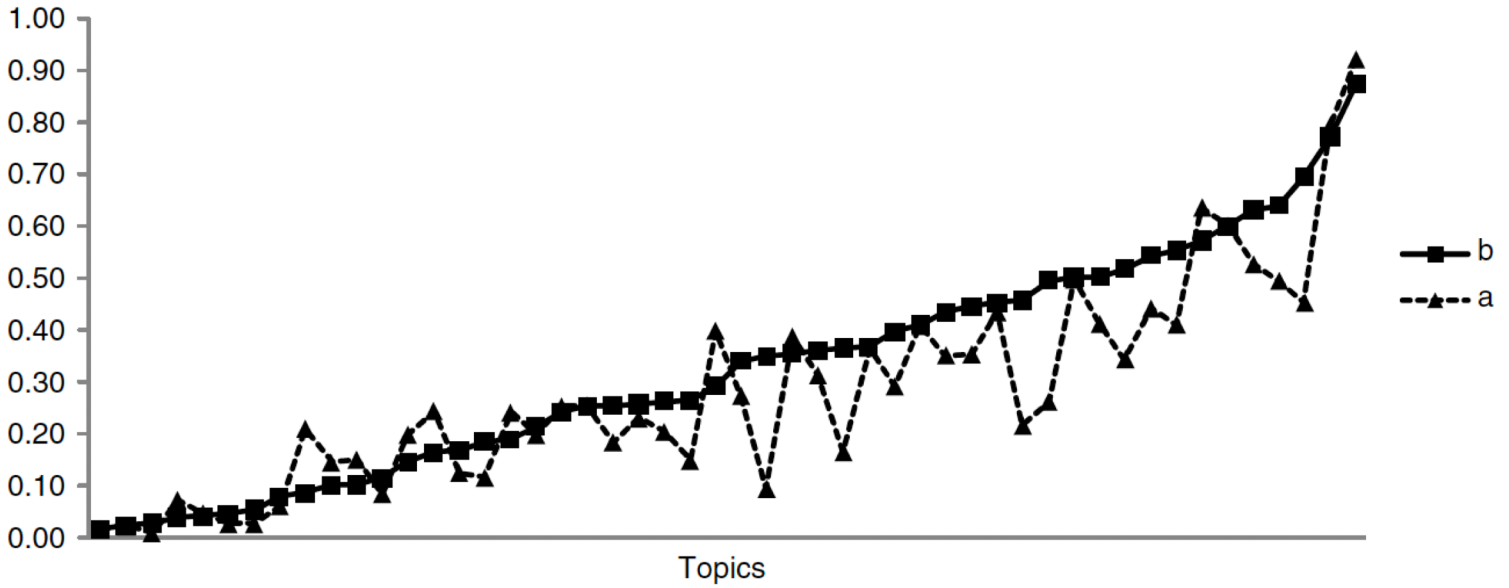


Fig. 5.1 Topic-by-topic comparison of two TREC-8 runs based on average precision scores.

[Mark Sanderson Book on Evaluation in IR]

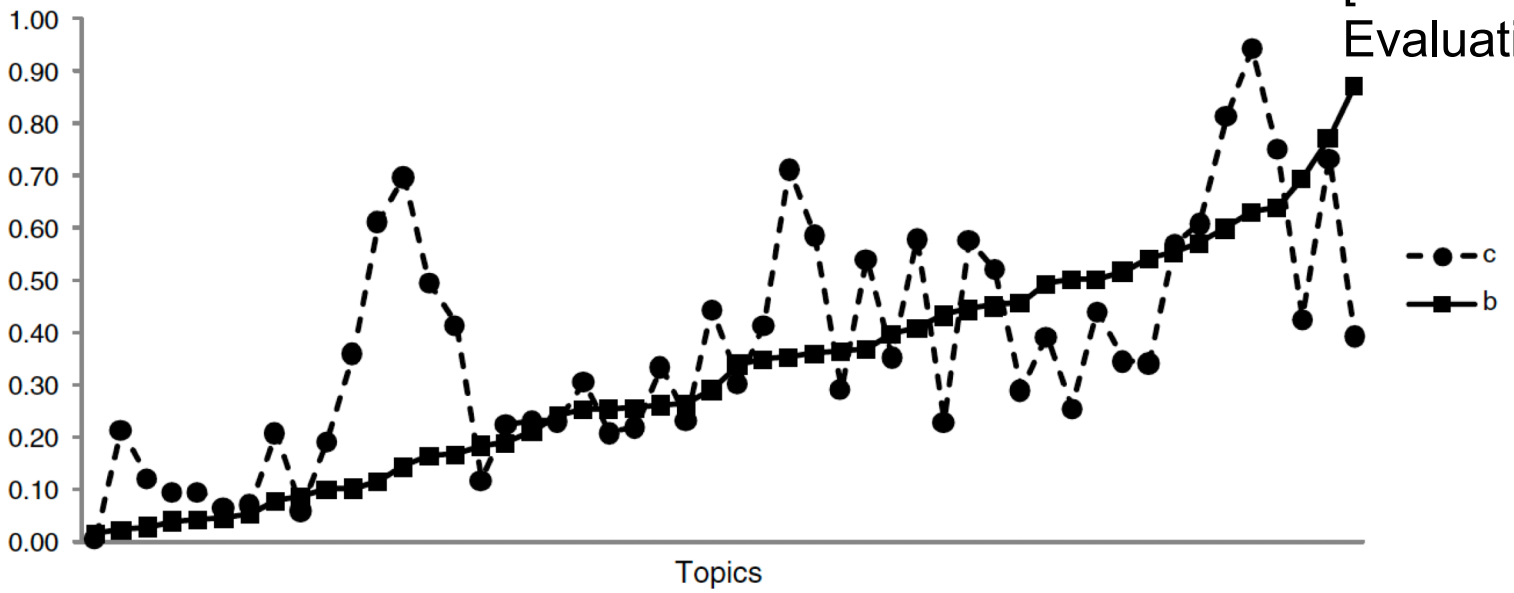


Fig. 5.2 Topic-by-topic comparison of two TREC-8 runs based on average precision scores – taken from Harman's work [104].

5.1.2 One or Two-Tail Tests?

So far H_0 has been described, but H_1 has not. There are two types of hypothesis that can be chosen for H_1, which correspond to different types of test: a one- and a two-*tailed* test (also known as a one- or two-*sided* test). In a two-tailed test, H_1 states that the two systems under examination are not equal, e.g., from the runs in Figure 5.1, H_1 would state that system a does not have the same retrieval characteristics as system b. Comparing a and b, a two-tailed t-test of H_0 returns $p = 0.002$; the Wilcoxon signed-rank test returns $p = 0.004$; and the sign test $p = 0.015$. Assuming a 5% threshold, regardless of which test was used, the experimenter would reject the null hypothesis and consider the difference between a and b to be significant.

Since IR experiments are often concerned with determining if a new type of IR system is better than an existing baseline, experimenters sometimes use a form of significance test that focuses only on the question of difference in one direction between two runs: this

is the one-tailed test. Here the experimenter predicts before conducting the experiment that one of the systems will be better than the other and sets H_1 to reflect that prediction. Taking this time the comparison from Figure 5.2, if system b is a baseline and system c is a new system under test, the experimenter sets H_1 to predict that $c > b$. A one-tailed t-test returns $p = 0.036$; the Wilcoxon $p = 0.026$; and the sign test, $p = 0.102$.[3] Despite the lack of significance in the sign test, most experimenters would consider the improvement of c over b as significant. The one-tailed test is recommended for use in IR experiments by Van Rijsbergen [262, Section 7] and more recently by Croft et al. [74], however, it is worth noting that in some areas of experimental science the one-tailed test is viewed as almost always inappropriate [8, p. 171].

The one-tailed version of a significance test has a p value that is half that of the two-tailed version, which makes it a tempting choice for experimenters as its use doubles the chance of finding significance. Note for example that all of the two-tailed tests comparing b and c would have failed to reject H_0. However, if using the one-tailed test, it is important to understand what its use entails. If from Figure 5.1 an experimenter had incorrectly predicted that system $a >$ baseline b and chose to use a one-tailed test; and upon discovering that a $< b$, the experimenter would *have* to conclude that they had failed to reject H_0. In other words, the experimenter would be obliged to report that a and b had the same retrieval characteristics, no matter how much worse a was compared to b; to many a strange conclusion to draw. The experimenter could of course conduct the one-tailed test in the opposite direction, but this second test could *only* be conducted on a new data set.

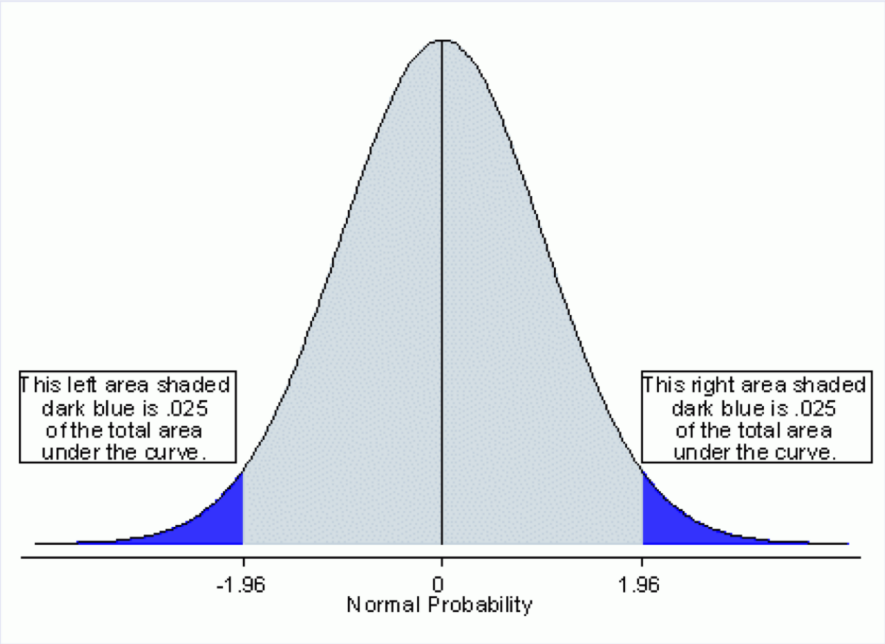
- Recalling the example in Figure 5.2, an abuse of significance tests would arise if an experimenter decided to use a two-tailed test, when comparing c and b, found no significance and so switched to a one-tailed test in the favorable direction in order to search out significance.

[3] It is also worth noting in the c and b comparison how the Wilcoxon and t-tests produced p values below the 0.05 threshold but the sign test did not. The former tests were more influenced by the substantial improvements of c over b in some topics. The sign test ignored the size of a difference; considering only the sign of the difference.

The choice of a one- or two-tailed test needs to be made before analyzing the data of an experiment and not after. If you are not sure of the direction of difference you wish to test for when comparing two systems, a two-tailed test is the appropriate choice. If you are certain that you only wish to test for a difference in one pre-selected direction, the one-tailed test can be used. It is important that the experimenter always states which "tailed version" they used when describing their work.

What is a two-tailed test?

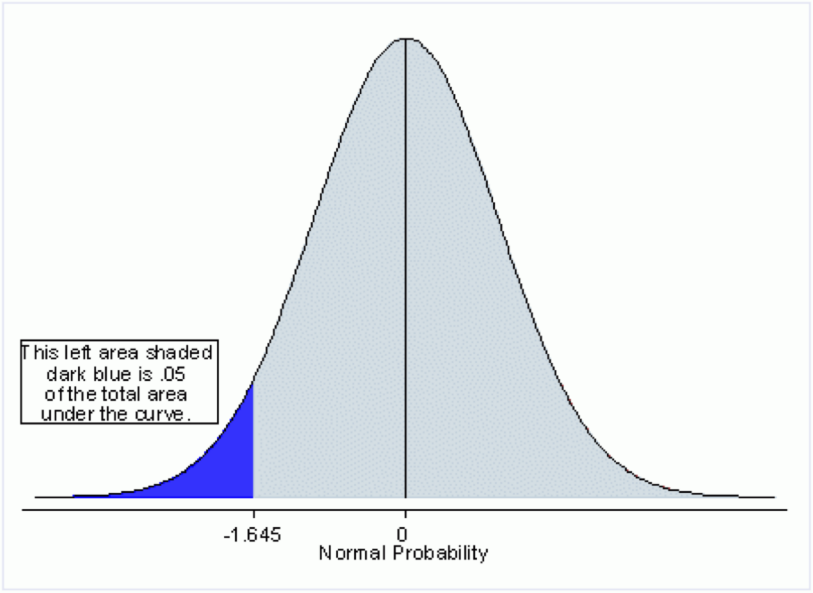
First let's start with the meaning of a two-tailed test. If you are using a significance level of 0.05, a two-tailed test allots half of your alpha to testing the statistical significance in one direction and half of your alpha to testing statistical significance in the other direction. This means that .025 is in each tail of the distribution of your test statistic. When using a two-tailed test, regardless of the direction of the relationship you hypothesize, you are testing for the possibility of the relationship in both directions. For example, we may wish to compare the mean of a sample to a given value *x* using a t-test. Our null hypothesis is that the mean is equal to *x*. A two-tailed test will test both if the mean is significantly greater than *x* and if the mean significantly less than *x*. The mean is considered significantly different from *x* if the test statistic is in the top 2.5% or bottom 2.5% of its probability distribution, resulting in a p-value less than 0.05.

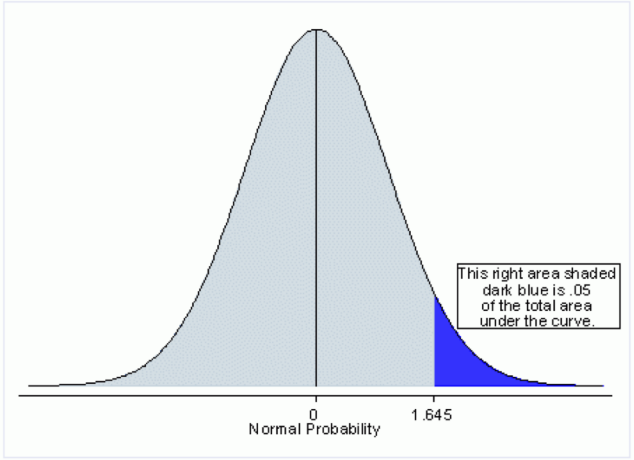


https://stats.idre.ucla.edu/other/mult-pkg/faq/general/faq-what-are-the-differences-between-one-tailed-and-two-tailed-tests/

What is a one-tailed test?

Next, let's discuss the meaning of a one-tailed test. If you are using a significance level of .05, a one-tailed test allots all of your alpha to testing the statistical significance in the one direction of interest. This means that .05 is in one tail of the distribution of your test statistic. When using a one-tailed test, you are testing for the possibility of the relationship in one direction and completely disregarding the possibility of a relationship in the other direction. Let's return to our example comparing the mean of a sample to a given value *x* using a t-test. Our null hypothesis is that the mean is equal to *x*. A one-tailed test will test either if the mean is significantly greater than *x* or if the mean is significantly less than *x*, but not both. Then, depending on the chosen tail, the mean is significantly greater than or less than *x* if the test statistic is in the top 5% of its probability distribution or bottom 5% of its probability distribution, resulting in a p-value less than 0.05. The one-tailed test provides more power to detect an effect in one direction by not testing the effect in the other direction. A discussion of when this is an appropriate option follows.





When is a one-tailed test appropriate?

Because the one-tailed test provides more power to detect an effect, you may be tempted to use a one-tailed test whenever you have a hypothesis about the direction of an effect. Before doing so, consider the consequences of missing an effect in the other direction. Imagine you have developed a new drug that you believe is an improvement over an existing drug. You wish to maximize your ability to detect the improvement, so you opt for a one-tailed test. In doing so, you fail to test for the possibility that the new drug is less effective than the existing drug. The consequences in this example are extreme, but they illustrate a danger of inappropriate use of a one-tailed test.

So when is a one-tailed test appropriate? If you consider the consequences of missing an effect in the untested direction and conclude that they are negligible and in no way irresponsible or unethical, then you can proceed with a one-tailed test. For example, imagine again that you have developed a new drug. It is cheaper than the existing drug and, you believe, no less effective. In testing this drug, you are only interested in testing if it less effective than the existing drug. You do not care if it is significantly more effective. You only wish to show that it is not less effective. In this scenario, a one-tailed test would be appropriate.

When is a one-tailed test NOT appropriate?

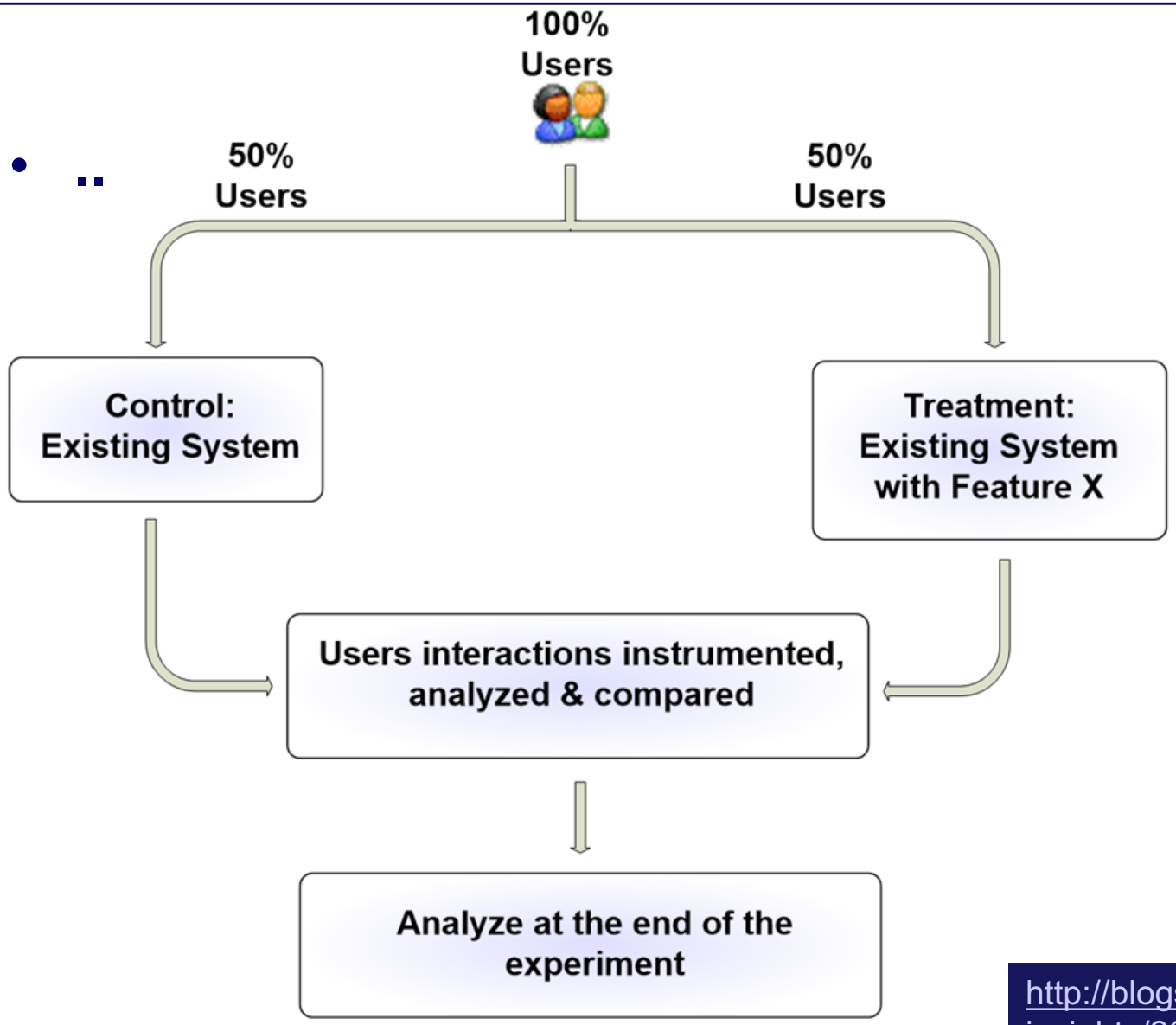
Choosing a one-tailed test for the sole purpose of attaining significance is not appropriate. Choosing a one-tailed test after running a two-tailed test that failed to reject the null hypothesis is not appropriate, no matter how "close" to significant the two-tailed test was. Using statistical tests inappropriately can lead to invalid results that are not replicable and highly questionable–a steep price to pay for a significance star in your results table!

In practice: Labs and in the wild

- **Is system A better that system B in a laboratory setting**
  - Paired
    - one-sided [you are sure challenger one is better
    - versus 2-sided [not sure if A and B are different; generally go with 2-sided)
  - Not paired
    - Is system A better that system B in the wild
      - one-sided vs two-sided

- **Scenario 1: Lab based experiments**
  - System A versus System B
- **Scenario 2: Deployed in the wild**
  - Incumbent system versus challenger (that you built!)
  - AB Testing (Control versus treatment)

Control versus treatment



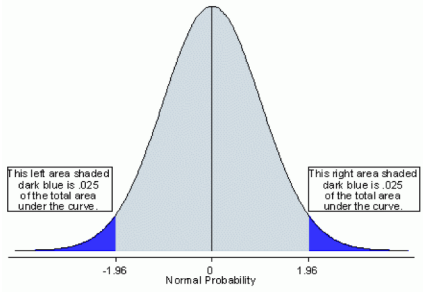
http://blogs.bing.com/search-quality-insights/2013/08/08/large-scale-experimentation-at-bing/

Two Classifier: paired t-test

Let x = test score before the module, y = test score after the module

To test the null hypothesis that the true mean difference is zero, the procedure is as follows:

1. Calculate the difference $(d_i = y_i - x_i)$ between the two observations on each pair, making sure you distinguish between positive and negative differences.
2. Calculate the mean difference, $\bar{d}$.
3. Calculate the standard deviation of the differences, s_d, and use this to calculate the standard error of the mean difference, $SE(\bar{d}) = \frac{s_d}{\sqrt{n}}$
4. Calculate the t-statistic, which is given by $T = \frac{\bar{d}}{SE(\bar{d})}$. Under the null hypothesis, this statistic follows a t-distribution with $n - 1$ degrees of freedom.
5. Use tables of the t-distribution to compare your value for T to the t_{n-1} distribution. This will give the p-value for the paired t-test.



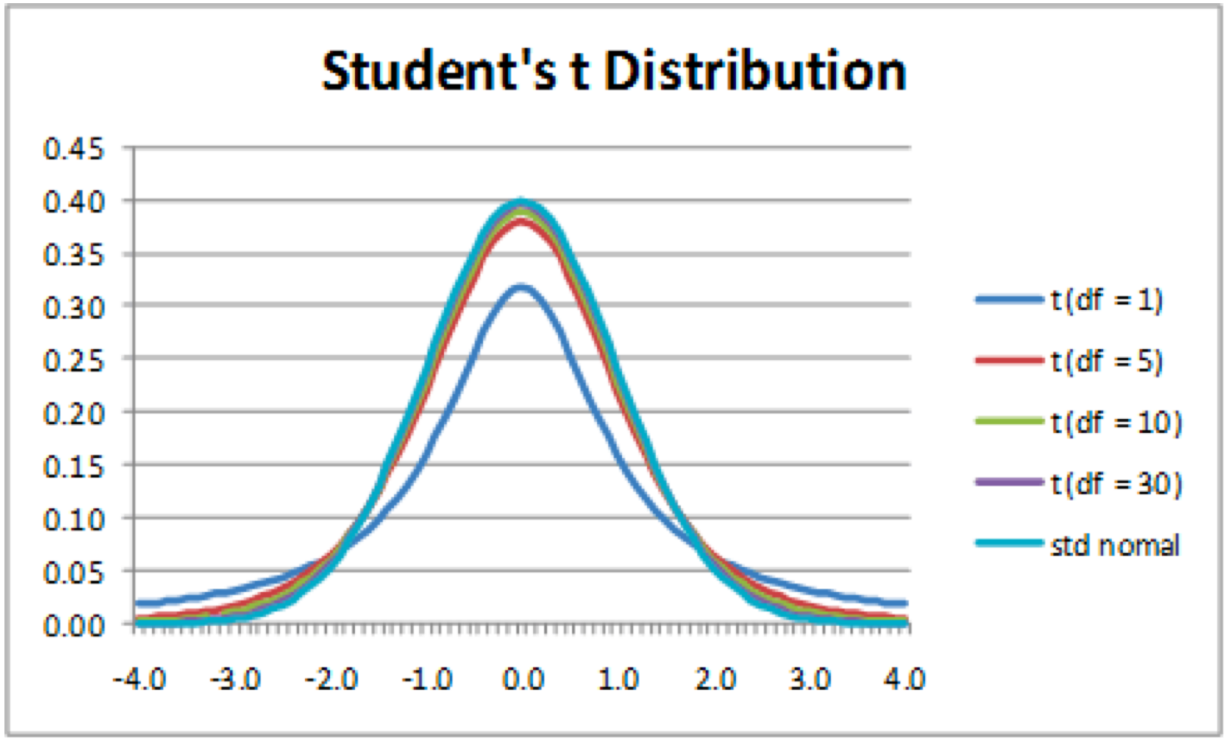
Paired t-test example

- **Paired t-test**
  - A paired t-test is used to compare two population means where you have two samples in which observations in one sample can be paired with observations in the other sample
- **Scenario:**
  - Assume a 20-class text classification problem.
  - Available here http://scikit-learn.org/stable/datasets/twenty_newsgroups.html
- **Build two systems**
  - SystemA: uses binary features for each word and a logistic regression classifier
  - SystemB: using TDIDF based features for each word and a logistic regression classifier
- **Calculate classification error foreach of the 20 classes using SystemA, and SystemB.**
- **Is SystemB significantly better than system A**
  - We will use an alpha value of 0.05 which means the results are significant if the p-value is below 0.05.

	SystemA	SystemB	
Class	Binary	TFIDF	Difference
1	18	22	+4
2	21	25	+4
3	16	17	+1
4	22	24	+2
5	19	16	-3
6	24	29	+5
7	17	20	+3
8	21	23	+2
9	23	19	-4
10	18	20	+2
11	14	15	+1
12	16	15	-1
13	16	18	+2
14	19	26	+7
15	18	18	0
16	20	24	+4
17	12	18	+6
18	22	25	+3
19	15	19	+4
20	17	16	-1

T-Test

- The t test (also called Student's T Test) compares two averages (means) and tells you if they are different from each other.
- The t test also tells you how significant the differences are; In other words it lets you know if those differences could have happened by chance.



cum. prob	$t_{.50}$	$t_{.75}$	$t_{.80}$	$t_{.85}$	$t_{.90}$	$t_{.95}$	$t_{.975}$	$t_{.99}$	$t_{.995}$	$t_{.999}$	$t_{.9995}$
one-tail	0.50	0.25	0.20	0.15	0.10	0.05	0.025	0.01	0.005	0.001	0.0005
two-tails	1.00	0.50	0.40	0.30	0.20	0.10	0.05	0.02	0.01	0.002	0.001
df											
1	0.000	1.000	1.376	1.963	3.078	6.314	12.71	31.82	63.66	318.31	636.62
2	0.000	0.816	1.061	1.386	1.886	2.920	4.303	6.965	9.925	22.327	31.599
3	0.000	0.765	0.978	1.250	1.638	2.353	3.182	4.541	5.841	10.215	12.924
4	0.000	0.741	0.941	1.190	1.533	2.132	2.776	3.747	4.604	7.173	8.610
5	0.000	0.727	0.920	1.156	1.476	2.015	2.571	3.365	4.032	5.893	6.869
6	0.000	0.718	0.906	1.134	1.440	1.943	2.447	3.143	3.707	5.208	5.959
7	0.000	0.711	0.896	1.119	1.415	1.895	2.365	2.998	3.499	4.785	5.408
8	0.000	0.706	0.889	1.108	1.397	1.860	2.306	2.896	3.355	4.501	5.041
9	0.000	0.703	0.883	1.100	1.383	1.833	2.262	2.821	3.250	4.297	4.781
10	0.000	0.700	0.879	1.093	1.372	1.812	2.228	2.764	3.169	4.144	4.587
11	0.000	0.697	0.876	1.088	1.363	1.796	2.201	2.718	3.106	4.025	4.437
12	0.000	0.695	0.873	1.083	1.356	1.782	2.179	2.681	3.055	3.930	4.318
13	0.000	0.694	0.870	1.079	1.350	1.771	2.160	2.650	3.012	3.852	4.221
14	0.000	0.692	0.868	1.076	1.345	1.761	2.145	2.624	2.977	3.787	4.140
15	0.000	0.691	0.866	1.074	1.341	1.753	2.131	2.602	2.947	3.733	4.073
16	0.000	0.690	0.865	1.071	1.337	1.746	2.120	2.583	2.921	3.686	4.015
17	0.000	0.689	0.863	1.069	1.333	1.740	2.110	2.567	2.898	3.646	3.965
18	0.000	0.688	0.862	1.067	1.330	1.734	2.101	2.552	2.878	3.610	3.922
19	0.000	0.688	0.861	1.066	1.328	1.729	2.093	2.539	2.861	3.579	3.883
20	0.000	0.687	0.860	1.064	1.325	1.725	2.086	2.528	2.845	3.552	3.850
21	0.000	0.686	0.859	1.063	1.323	1.721	2.080	2.518	2.831	3.527	3.819
22	0.000	0.686	0.858	1.061	1.321	1.717	2.074	2.508	2.819	3.505	3.792
23	0.000	0.685	0.858	1.060	1.319	1.714	2.069	2.500	2.807	3.485	3.768
24	0.000	0.685	0.857	1.059	1.318	1.711	2.064	2.492	2.797	3.467	3.745
25	0.000	0.684	0.856	1.058	1.316	1.708	2.060	2.485	2.787	3.450	3.725
26	0.000	0.684	0.856	1.058	1.315	1.706	2.056	2.479	2.779	3.435	3.707
27	0.000	0.684	0.855	1.057	1.314	1.703	2.052	2.473	2.771	3.421	3.690
28	0.000	0.683	0.855	1.056	1.313	1.701	2.048	2.467	2.763	3.408	3.674
29	0.000	0.683	0.854	1.055	1.311	1.699	2.045	2.462	2.756	3.396	3.659
30	0.000	0.683	0.854	1.055	1.310	1.697	2.042	2.457	2.750	3.385	3.646
40	0.000	0.681	0.851	1.050	1.303	1.684	2.021	2.423	2.704	3.307	3.551
60	0.000	0.679	0.848	1.045	1.296	1.671	2.000	2.390	2.660	3.232	3.460
80	0.000	0.678	0.846	1.043	1.292	1.664	1.990	2.374	2.639	3.195	3.416
100	0.000	0.677	0.845	1.042	1.290	1.660	1.984	2.364	2.626	3.174	3.390
1000	0.000	0.675	0.842	1.037	1.282	1.646	1.962	2.330	2.581	3.098	3.300
z	0.000	0.674	0.842	1.036	1.282	1.645	1.960	2.326	2.576	3.090	3.291
	0%	50%	60%	70%	80%	90%	95%	98%	99%	99.8%	99.9%
	Confidence Level										

Paired t-test in python

Previously the observations in our two samples have been completely independent of one another.

Perhaps we want to compare two related samples, e.g. a before and after test, we might use a paired T-test.

This is calculated as follows:

$$t = \frac{\bar{d}}{s/\sqrt{n}}$$

Where $\bar{d}$ is the average difference between the paired samples. The degrees of freedom is $n - 1$.

Example

We will measure the amount of sleep got by patients before and after taking soporific drugs to help them sleep.

The null hypothesis is that the soporific drug has no effect on the sleep duration of the patients.

Scipy implements the paired t-test as `ttest_rel()`

In [3]:

```
control = [8.0, 7.1, 6.5, 6.7, 7.2, 5.4, 4.7, 8.1, 6.3, 4.8]
treatment = [9.9, 7.9, 7.6, 6.8, 7.1, 9.9, 10.5, 9.7, 10.9, 8.2]

stats.ttest_rel(control, treatment)
```

Out[3]:

```
Ttest_relResult(statistic=-3.6244859951782136, pvalue=0.0055329408161001415)
```

Our t-statistic value is -3.624, and along with our degrees of freedom (9) this can be used to calculate a p-value.

The p-value is 0.0055, which again is below than the standard thresholds of 0.05 or 0.01, so we reject the null hypothesis and we can say there is a statistically significant difference in sleep duration caused by the soporific drug.

Paired t-test example

- **Paired t-test**
  - A paired t-test is used to compare two population means where you have two samples in which observations in one sample can be paired with

	SystemA	SystemB	
Class	Binary	TFIDF	Difference
1	18	22	+4
2	21	25	+4
3	16	17	+1
4	22	24	+2

Calculating the mean and standard deviation of the differences gives:

$\bar{d} = 2.05$ and $s_d = 2.837$. Therefore, $SE(\bar{d}) = \frac{s_d}{\sqrt{n}} = \frac{2.837}{\sqrt{20}} = 0.634$

$$t = \frac{\bar{X}_D - \mu_0}{\frac{s_D}{\sqrt{n}}}.$$

So, we have:

$$t = \frac{2.05}{0.634} = 3.231 \qquad \text{on 19 df}$$

Looking this up in tables gives $p = 0.004$. Therefore, there is strong evidence that, on average, the module does lead to improvements.

Based on the p-value of 0.0004, we can reject the null hypothesis. (Statisticians like us to say reject the null rather than accept the alternative.) There is **statistically significant** evidence that SystemB (TFIDF-based) classifies better on average than SystemA (Binary) at a significance level of 0.05. The p-value shows there is a 0.4% chance that our results occurred because of random noise

Using TFIDF improves the classification error significantly

Confidence interval for the true mean difference

The in above example the estimated average improvement is just over 2 points. Note that although this is statistically significant, it is actually quite a small increase. It would be useful to calculate a confidence interval for the mean difference to tell us within what limits the true difference is likely to lie. A 95% confidence interval for the true mean difference is:

$$\bar{d} \pm t^* \frac{s_d}{\sqrt{n}} \quad \text{or, equivalently} \quad \bar{d} \pm (t^* \times SE(\bar{d}))$$

where t^* is the 2.5% point of the t-distribution on $n-1$ degrees of freedom.

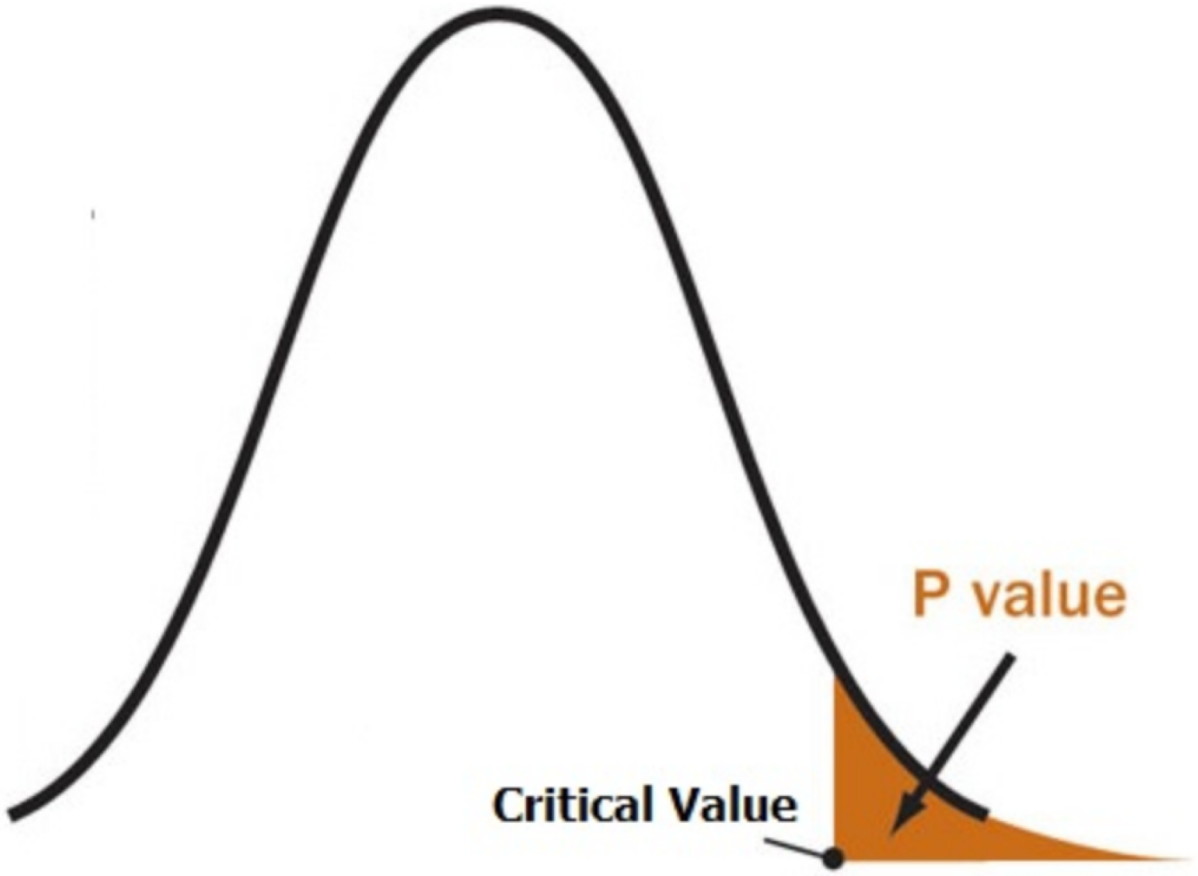
Using our example:

We have a mean difference of 2.05. The 2.5% point of the t-distribution with 19 degrees of freedom is 2.093. The 95% confidence interval for the true mean difference is therefore:

$$2.05 \pm (2.093 \times 0.634) = 2.05 \pm 1.33 = (0.72, 3.38)$$

This confirms that, although the difference in scores is statistically significant, it is actually relatively small. We can be 95% sure that the true mean increase lies somewhere between just under one point and just over 3 points.

We see this difference naturally when the null H is true



A p-value is an area in the tail of a distribution that tells you the odds of a result happening by chance.

Part

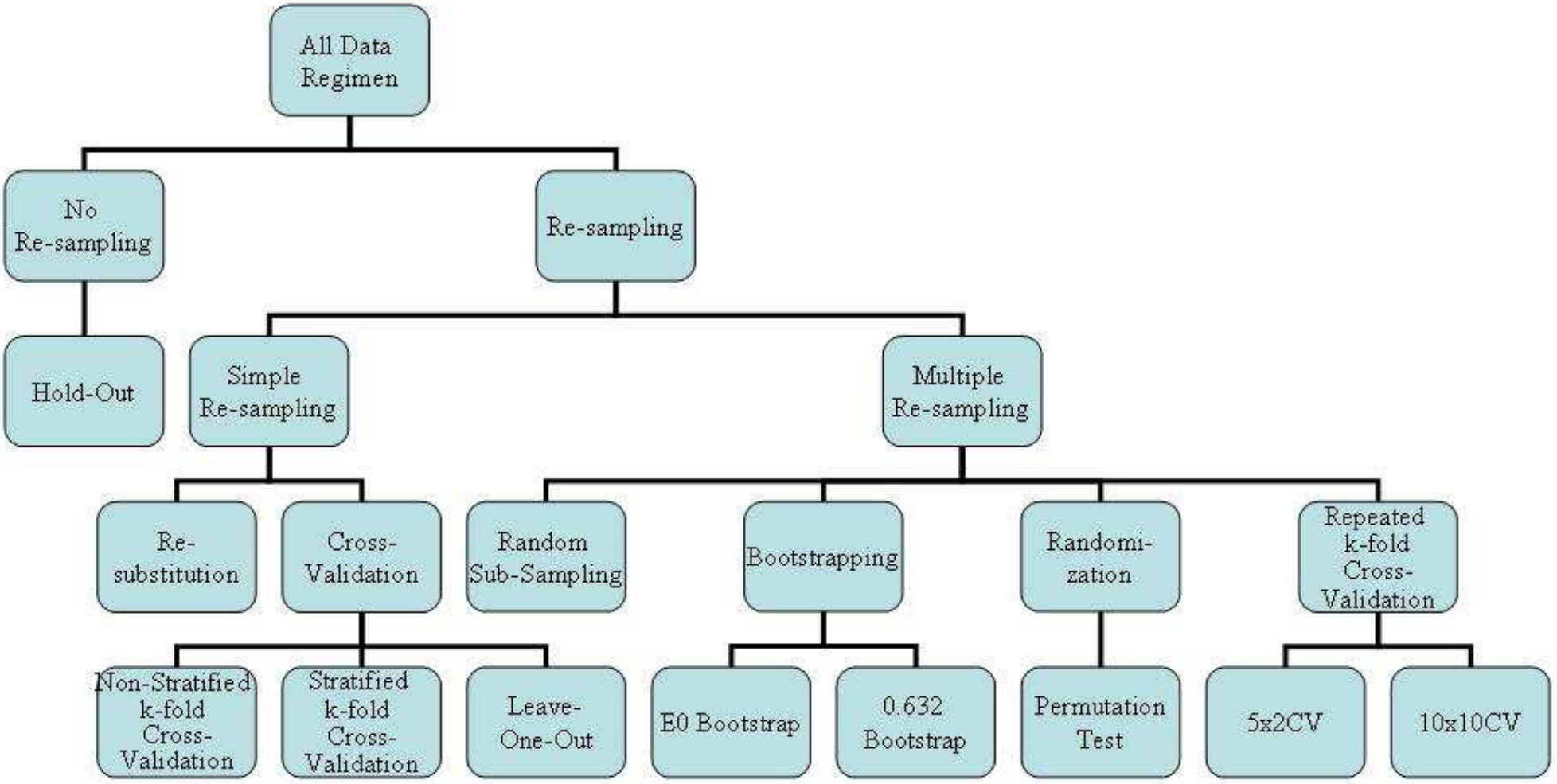
StatSig Tests via sampling

StatSig Tests via sampling

One approach; many exist

- **What is the Purpose of Resampling?**
  - Ideally, we would have access to the entire population or a lot of representative data from it.
  - This is usually not the case, and the limited data available has to be re-used in clever ways in order to be able to estimate the error of our classifiers as reliably as possible, i.e., to be re-used in clever ways in order to obtain sufficiently large numbers of samples.
  - Resampling is divided into two categories:
    - Simple re-sampling (where each data point is used for testing only once) and
    - Multiple re-sampling (which allows the use of the same data point more than once for testing)

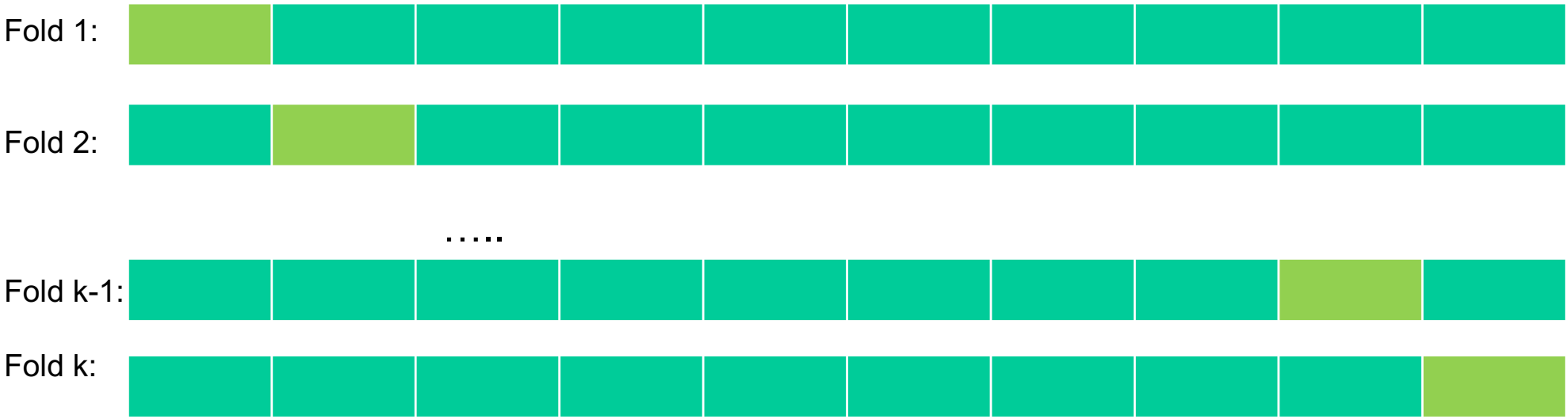
Overview of Re-Sampling Methods



Resampling Approaches

- **Simple re-sampling (where each data point is used for testing only once) and**
  - Cross-Validation and its variants
- **Multiple re-sampling (which allows the use of the same data point more than once for testing):**
  - The 0.632 Bootstrap
  - The Permutation Test
  - Repeated k-fold Cross-Validation

k-fold Cross-Validation



: training data subset
: testing data subset

In Cross-Validation, the data set is divided into k folds and at each iteration, a different fold is reserved for testing and all the others, used for training the classifiers.

StatSig Tests for ML Pipeline via sampling

One approach; many exist

- **StatSig Test full example for ML pipelines via crossfold validation**
- **In the case of machine learning (using SKLearn), let's assume the **CONTROL** is a BASELINE pipelines, say a linear regression pipeline, and the **TREATMENT** is a decision tree regression based pipeline (assume default settings for both modeling strategies).**
  - 1. We run both systems on using k-fold cross validation with k=30 using a regression performance metrics such as RMSE.
  - * Each system produces an array of 30 values of the performance metric being considered: control and treatment pipelines.
- **Perform a two-sided t-test to determine if there there is a statistically significant difference between the control and treatment pipelines**

9.2 StatSig Test full example for ML pipelines via crossfold validation

In the case of machine learning (using SKLearn), let's assume the **CONTROL** is a BASELINE system, say a linear regression, and the **TREATMENT** is a decision tree regression model (with all default settings).

1. We run both systems on using k-fold cross validation with k=30 using a regression performance metrics such as RMSE.
   - Each system produces an array of 30 values of the performance metric being considered: control and treatment.
2. Perform a two-sided t-test to determine if there there is a statistically significant difference between the control and treatment as follows:

```
(t_score, p_value) = stats.ttest_rel(control, treatment)
print("The p-value is %0.5f for a z-score of %0.5f." %(p_value, t_score))

#"The p-value is 0.0005 for a z-score of 3.62449."
```

Let's assume the t-statistic value is 3.624, and along with our degrees of freedom (29) this can be used to calculate a p-value for out ML pipeline evaluation.

The corresponding p-value is 0.0005, which again is below than the standard thresholds of 0.025 or 0.005, so we reject the null hypothesis and we can say there is a statistically significant difference in the models and is likely to be caused by the use of decision trees.

```
from sklearn.model_selection import cross_val_score
from sklearn.tree import DecisionTreeRegressor
from sklearn.linear_model import LinearRegression

def display_scores(scores):
    print("Scores:", scores)
    print("Mean:", scores.mean())
    print("Standard deviation:", scores.std())

# A sampling based bakeoff using *K-fold cross-validation*:
# it randomly splits the training set into K distinct subsets (k=30)
# this bakeoff framework can be used for regression or classification
#Control system is a linear regression based pipeline

kFolds=30
lin_reg = LinearRegression()
lin_scores = cross_val_score(lin_reg, housing_prepared, housing_labels,
                             scoring="neg_mean_squared_error", cv=kFolds)
control = lin_rmse_scores = np.sqrt(-lin_scores)
display_scores(lin_rmse_scores)

#Treatment system is a Decision Tree regression based pipeline
tree_reg = DecisionTreeRegressor(random_state=42)
scores = cross_val_score(tree_reg, housing_prepared, housing_labels,
                         scoring="neg_mean_squared_error", cv=kFolds)
treatment = tree_rmse_scores = np.sqrt(-scores)
display_scores(tree_rmse_scores)

#paired t-test; two-tailed p-value
(t_score, p_value) = stats.ttest_rel(control, treatment)
print("The p-value is %0.5f for a z-score of %0.5f." %(p_value, t_score))
#"The p-value is 0.00019 for a z-score of -4.28218."
```

```
#Treatment system is a Decision Tree regression based pipeline
tree_reg = DecisionTreeRegressor(random_state=42)
scores = cross_val_score(tree_reg, housing_prepared, housing_labels,
                         scoring="neg_mean_squared_error", cv=kFolds)
treatment = tree_rmse_scores = np.sqrt(-scores)
display_scores(tree_rmse_scores)

#paired t-test; two-tailed p-value
(t_score, p_value) = stats.ttest_rel(control, treatment)
print("The p-value is %0.5f for a z-score of %0.5f." %(p_value, t_score))
#"The p-value is 0.00019 for a z-score of -4.28218."
```

```
Scores: [ 71636.43163599  66412.29575468  62332.38167651  70240.16025321
  61338.71015859  68531.37361648  70457.42628915  64109.65865297
  76263.49619408  78211.46696486  69493.28334785  74791.7245885
  69431.45308878  69443.78825363  64814.60856869  72608.88698235
  70528.92240663  70395.87554003  63215.94501339  66466.24261817
  65596.82304454  66726.35521863  62728.29126687  74738.01360262
  67836.34784403  74108.12971254  71858.1996621   69174.01562603
  67857.64512403  65929.0807718 ]
Mean: 68909.2344493
Standard deviation: 4188.1766678
Scores: [ 74842.94955227  69451.80957361  65060.43115458  68743.65844281
  66612.29572654  71266.62444992  77316.25983269  72179.340228
  74387.44716592  68204.08859936  66805.12850512  78091.96466728
  74070.66132508  75552.20735138  72669.06756005  75223.36606553
  70564.65956349  74760.37185841  64733.15803735  74595.18060745
  67061.85221007  67293.62353176  66417.2951904   76668.94011649
  75358.43895348  75635.87262695  74131.01206564  73326.80104872
  75601.56056144  70590.39398243]
Mean: 71907.2153518
Standard deviation: 3904.57448872
The p-value is 0.00019 for a z-score of -4.28218.
```

The p-value is 0.00019 for a z-score of -4.28218, along with our degrees of freedom (29) means that the Control system is statistically different.

The corresponding p-value is 0.0005, which again is below than the standard thresholds of 0.025 or 0.005, so we reject the null hypothesis and we can say there is a statistically significant difference in the models and is likely to be caused by the use of decision trees. The p-value in the test output shows that the chances of seeing this large of a difference between samples due to chance (assuming the H0, the null hypothesis, is true) is just 0.019%.

Based upon this bakeoof (highly limited) we would select Linear regression and possibly deploy it into production. (In the real world, we would do lots more lab-based experimentation).

LR vs DT

10.1 Example paired t-test

We will measure the amount of sleep got by patients before and after taking soporific drugs to help them sleep.

The null hypothesis is that the soporific drug has no effect on the sleep duration of the patients.

Scipy implements the paired t-test as `ttest_rel()`

For more background see here.

```
import numpy as np
from scipy import stats

control = [8.0, 7.1, 6.5, 6.7, 7.2, 5.4, 4.7, 8.1, 6.3, 4.8]
treatment = [9.9, 7.9, 7.6, 6.8, 7.1, 9.9, 10.5, 9.7, 10.9, 8.2]

#paired t-test; two-tailed p-value
(t_score, p_value) = stats.ttest_rel(control, treatment)
print("The p-value is %0.5f for a t-score of %0.5f." %(p_value, t_score))
#"The p-value is 0.00553 for a z-score of -3.62449."

#in the case of regression lower RMSE is better
#If RMSE for system A (Control) is 8.0 and the RMSE for system B (treatment) is 9.0
#then the difference (A-B) is -1; so negative tScores meant System A is better than System B

if p_value > 0.05/2:  #Two sided
    print('There is no significant difference between the two machine learning pipelines (Accept H0)')
else:
    print('The two machine learning pipelines are different (reject H0) \n(t_score, p_value) = (%.2f, %.5f)'%(t_score,
    if t_score < 0.0: #in the case of regression lower RMSE is better; A is lower
        print('Machine learning pipeline A is better than B')
    else:
        print('Machine learning pipeline B is better than A')
```

```
The p-value is 0.00553 for a z-score of -3.62449.
The two machine learning pipelines are different (reject H0)
(t_score, p_value) = (-3.62, 0.00553)
Machine learning pipeline A is better than B
```

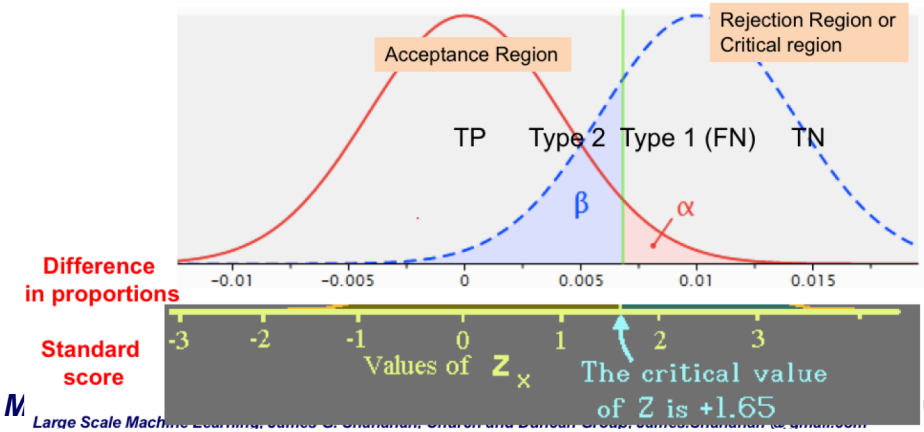
Our t-statistic value is -3.624, and along with our degrees of freedom (9) this can be used to calculate a p-value.

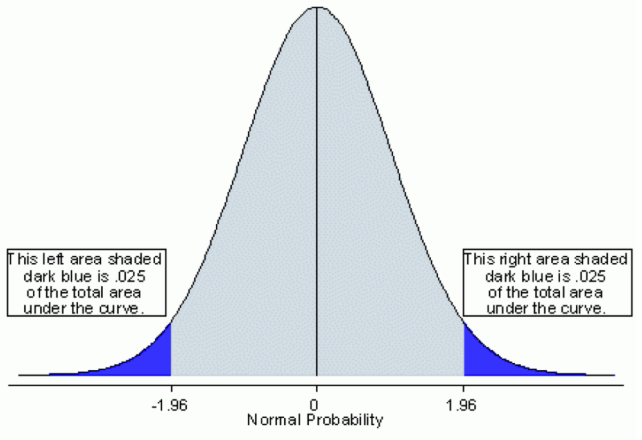
The p-value is 0.0055, which again is below than the standard thresholds of 0.05/2 or 0.01/2, so we reject the null hypothesis and we can say there is a statistically significant difference in sleep duration caused by the soporific drug.

RMSE (smaller is better)

LR	– DT	Diff
40,000	– 60,000	-20,000
30,000	– 70,000	-40,000

Average.	-30,000. - 0
Std.	~10,000
Standardize.	-3



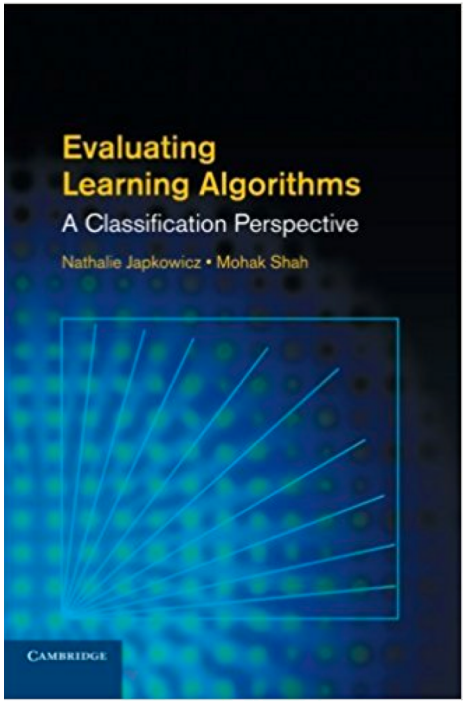


Lab experiment: not paired

- It is good practice to gather a population of results when comparing two different machine learning algorithms or when comparing the same algorithm with different configurations.
- Repeating each experimental run 30 or more times gives you a population of results from which you can calculate the mean expected performance, given the stochastic nature of most machine learning algorithms.
- If the mean expected performance from two algorithms or configurations are different, how do you know that the difference is significant, and how significant?
- Statistical significance tests are an important tool to help to interpret the results from machine learning experiments. Additionally, the findings from these tools can help you better and more confidently present your experimental results and choose the right algorithms and configurations for your predictive modeling problem.

Evaluating Learning Algorithms: A Classification Perspective

- Book on **Evaluating Learning Algorithms: A Classification Perspective**
  - There is no single test that is appropriate for all situations; I can recommend the book "Evaluating Learning Algorithms" by Nathalie Japkowicz and Mohak Shah, Cambridge University Press, 2011. The fact that a book of almost 400 pages can be written on this topic suggests it isn't a straight-forward issue. I have often found that there isn't a test that really suits the needs of my study, so it is important to have a good grasp of the advantages and disadvantages of whatever method is eventually used.



Significance tests for classification tasks

- http://citeseer.ist.psu.edu/viewdoc/download;jsessionid=A5311100581D766C7E10F90CD4C25761F?doi=10.1.1.37.3325&rep=rep1&type=pdf

Approximate Statistical Tests for Comparing Supervised Classification Learning Algorithms

Thomas G. Dietterich
tgd@cs.orst.edu
Department of Computer Science
Oregon State University
Corvallis, OR 97331

December 30, 1997

Abstract

This paper reviews five approximate statistical tests for determining whether one learning algorithm out-performs another on a particular learning task. These tests are compared experimentally to determine their probability of incorrectly detecting a difference when no difference exists (type I error). Two widely-used statistical tests are shown to have high probability of Type I error in certain situations and should never be used. These tests are (a) a test for the difference of two proportions and (b) a paired-differences t test based on taking several random train/test splits. A third test, a paired-differences t test based on 10-fold cross-validation, exhibits somewhat elevated probability of Type I error. A fourth test, McNemar's test, is shown to have low Type I error. The fifth test is a new test, 5x2cv, based on 5 iterations of 2-fold cross-validation. Experiments show that this test also has acceptable Type I error. The paper also measures the power (ability to detect algorithm differences when they do exist) of these tests. The cross-validated t test is the most powerful. The 5x2cv test is shown to be slightly more powerful than McNemar's test. The choice of the best test is determined by the computational cost of running the learning algorithm. For algorithms that can be executed only once, McNemar's test is the only test with acceptable Type I error. For algorithms that can be executed ten times, the 5x2cv test is recommended, because it is slightly more powerful and because it directly measures variation due to the choice of training set.

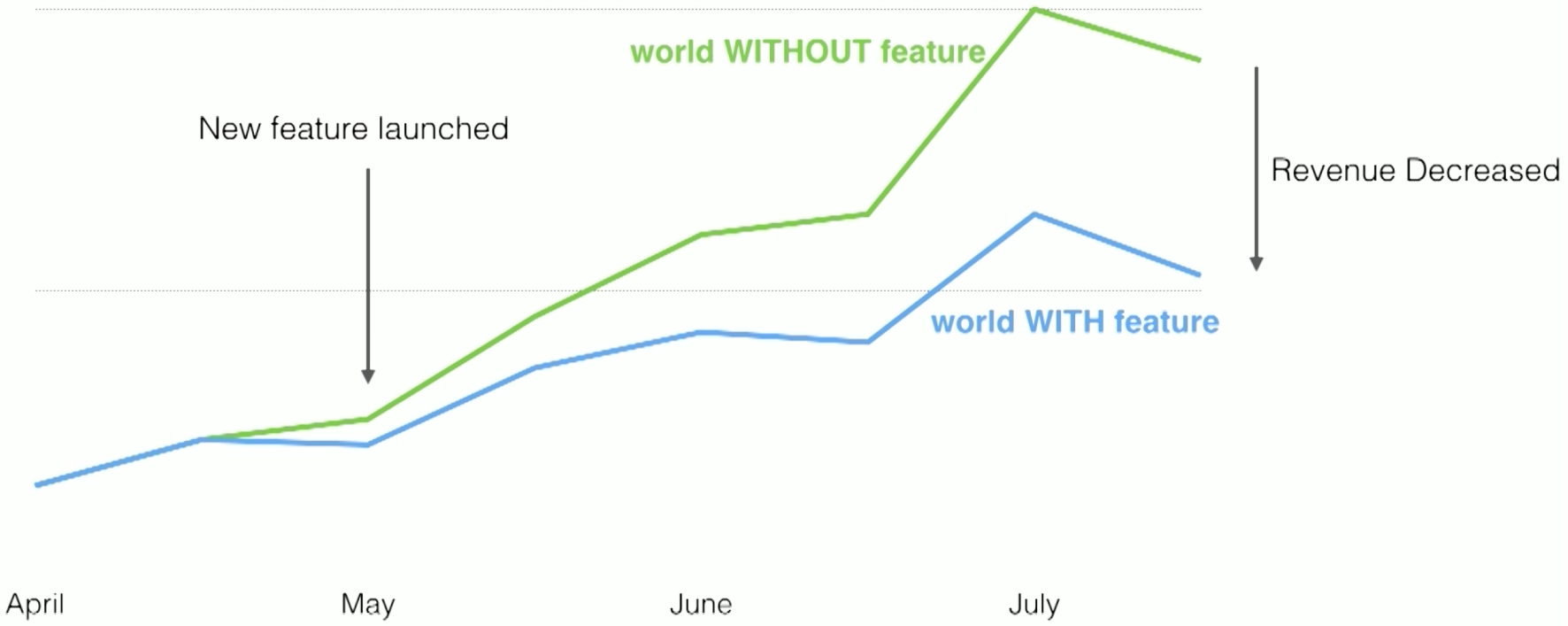
Recommend to use McNemar's test to tell if classifier A is better than B

A/B Testing, A Data Science Perspective

Setting up an Experiment

1. Define your goal and form your hypotheses.
2. Identify a control and a treatment.
3. Identify key metrics to measure.
4. Identify what data needs to be collected.
5. Make sure that appropriate logging is in place to collect all necessary data.
6. Determine how small of a difference you would like to detect.
7. Determine what fraction of visitors you want to be in the treatment
8. Run a power analysis to decide how much data you need to collect and how long you need to run the test.
9. Run the test for AT LEAST this long.
10. First time trying something new: run an A/A test (dummy test) simultaneously to check for systematic biases.

A/B Testing, A Data Science Perspective



Outline

- **Introduction**
- **End to end ML project in SKLearn**
  - Problem definition and data
  - EDA
  - SKLearn Pipelines for prepping data
  - Full learning pipeline
  - Finetune model:
    - GridSearch versus Random
- **Which model is better? Significance Tests**
- **Summary**

End of lecture