GENERAL ASSEMBLY

EVALUATING MODEL FIT

Abbas Chokor, Ph.D.

Staff Data Scientist, Seagate Technology

OUR PROGRESS SO FAR

UNIT 1: RESEARCH DESIGN AND EXPLORATORY DATA ANALYSIS	~~What is Data Science~~	~~Lesson 1~~
	~~Research Design and Pandas~~	~~Lesson 2~~
	~~Statistics Fundamentals I~~	~~Lesson 3~~
	~~Statistics Fundamentals II~~	~~Lesson 4~~
	~~Flexible Class Session~~	~~Lesson 5~~
UNIT 2: FOUNDATIONS OF DATA MODELING	~~Introduction to Regression~~	~~Lesson 6~~
	Evaluating Model Fit	Lesson 7
	Introduction to Classification	Lesson 8
	Introduction to Logistic Regression	Lesson 9
	Communicating Logistic Regression Results	Lesson 10
	Flexible Class Session	Lesson 11
UNIT 3: DATA SCIENCE IN THE REAL WORLD	Decision Trees and Random Forests	Lesson 12
	Natural Language Processing	Lesson 13
	Dimensionality Reduction	Lesson 14
	Time Series Data I	Lesson 15
	Time Series Data II	Lesson 16
	Database Technologies	Lesson 17
	Where to Go Next	Lesson 18
	Flexible Class Session	Lesson 19
	Final Project Presentations	Lesson 20

Today's Class

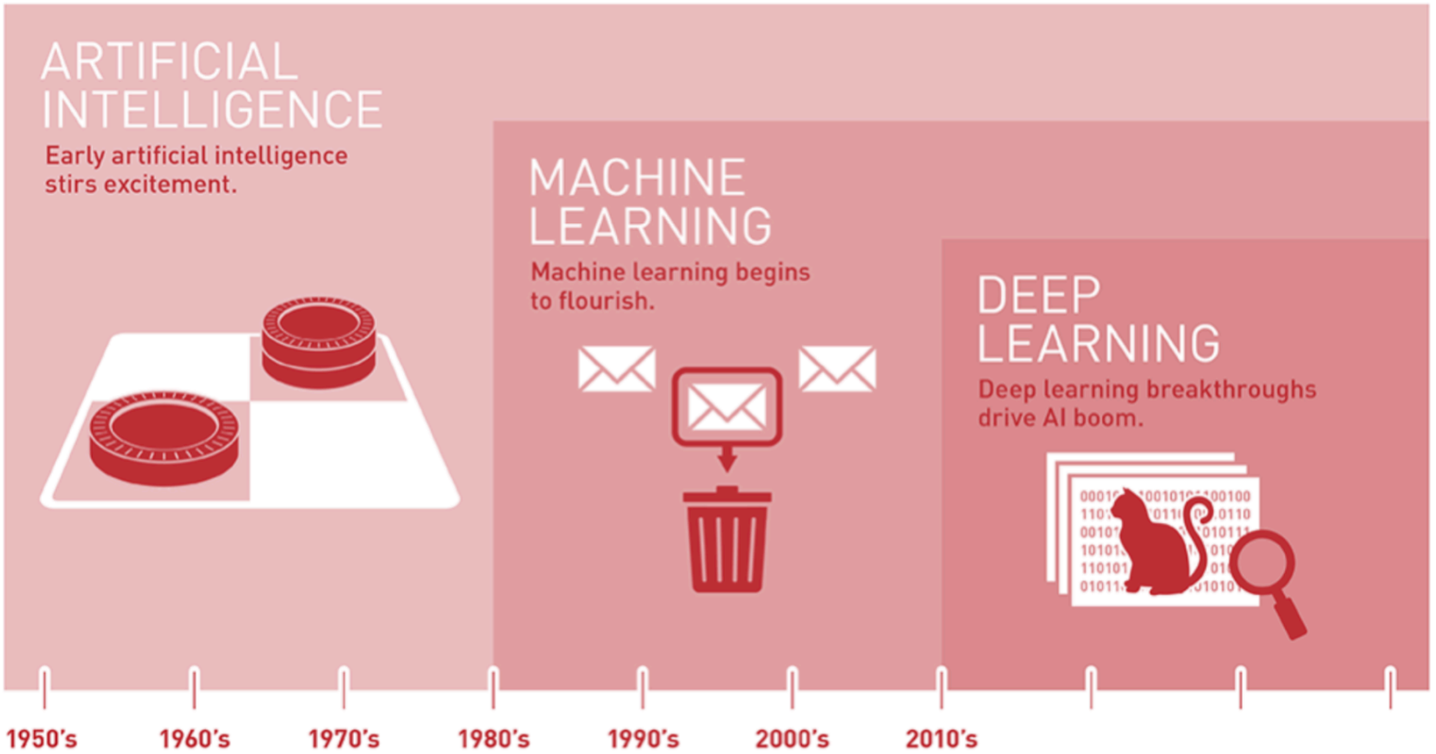
LAST CLASS

- Overview of machine learning modeling
- Define data modeling and simple linear regression
- Build a multivariate linear regression model using a dataset that meets the linearity assumption using the scikit-learn library
- Understand and identify multicollinearity in a multiple regression.

WHAT IS MACHINE LEARNING

[Machine Learning is the] field of study that gives computers the ability to learn without being explicitly programmed.

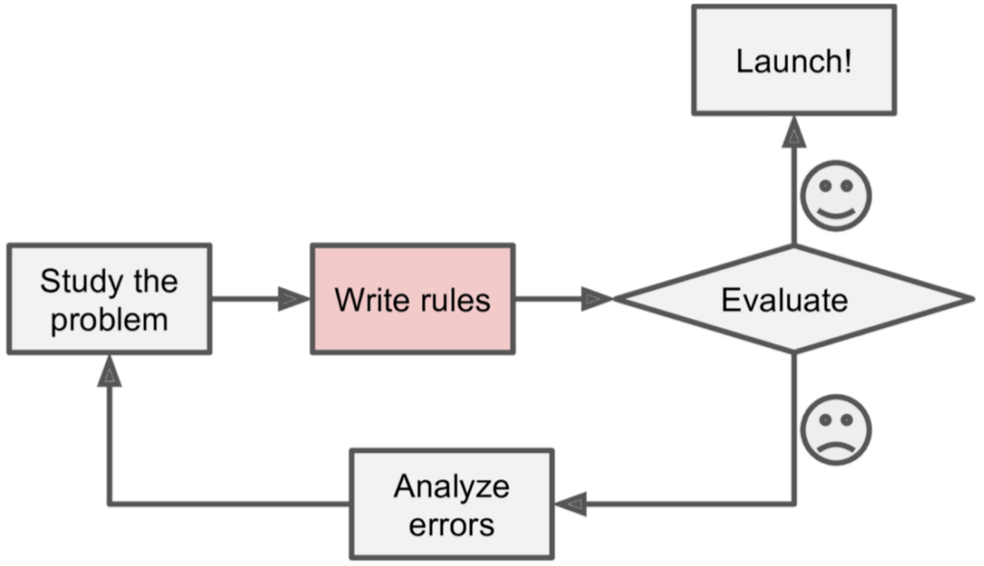
Arthur Samuel, *1959*

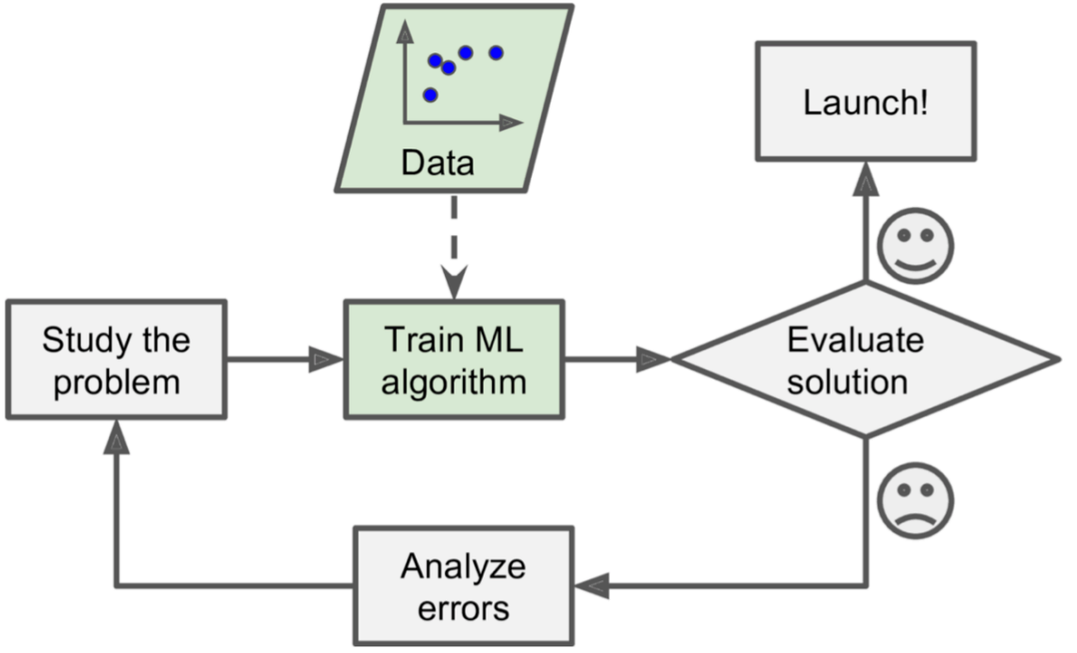


Since an early flush of optimism in the 1950s, smaller subsets of artificial intelligence – first machine learning, then deep learning, a subset of machine learning – have created ever larger disruptions.

<u>Machine Learning</u>: is the science (and art) of programming computers so they can *learn from data*.

WHY TO USE MACHINE LEARNING?



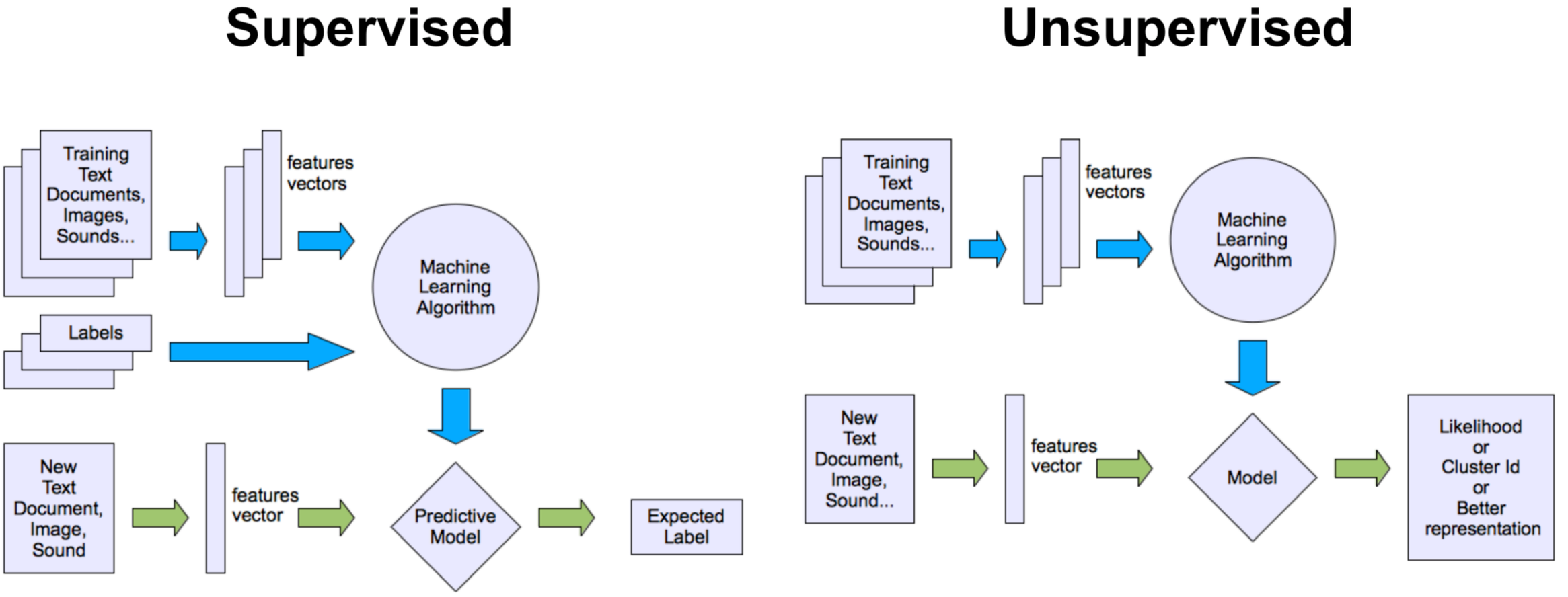


Traditional approach **Vs.** ***Machine Learning approach***

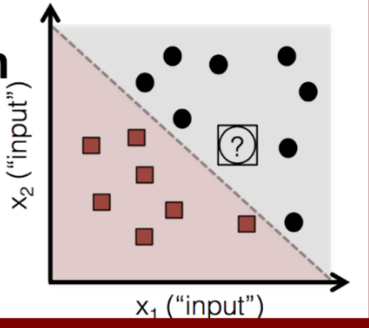
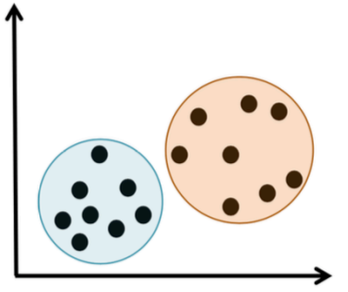
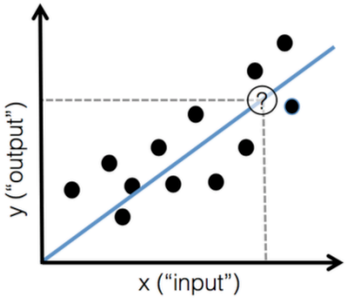
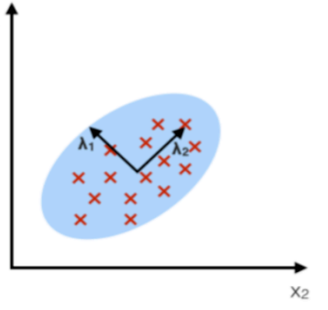
Did you know?

Machine learning algorithms are expected to replace 25% of the jobs across the world, in the next 10 years.

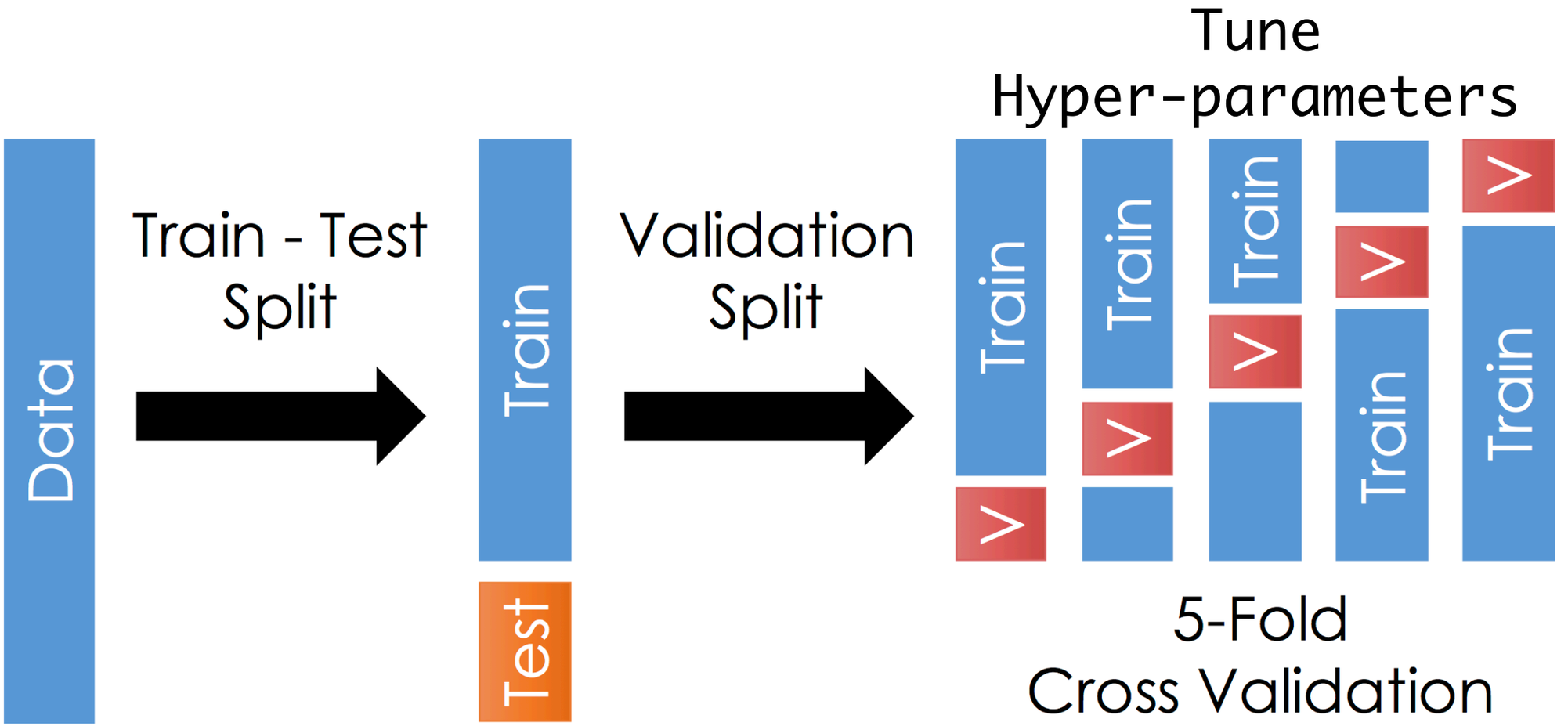
TYPES OF MACHINE LEARNING



TYPES OF MACHINE LEARNING

	Supervised Working with Labeled Data	**Unsupervised** Working with Unlabeled Data
Discrete Countable Data	**Classification** 	**Clustering** 
Continuous Infinite Data	**Regression** 	**Dimensionality Reduction** 

TRAINING – VALIDATING - TESTING



SIMPLE LINEAR REGRESSION

- However, linear regression uses linear algebra to explain the relationship between *multiple* x's and y.

- The more sophisticated version: y = beta * X + alpha (+ error)

- Explain the relationship between the matrix **X** and a dependent vector **y** using a y-intercept **alpha** and the relative coefficients **beta**.

REVIEW

UNIT PROJECT 2 REVIEW

Unit Project 2

Project 2

In this project, you will implement the exploratory analysis plan developed in Project 1. This will lay the groundwork for our our first modeling exercise in Project 3.

Step 1: Load the python libraries you will need for this project

In [1]:
```
#imports
from __future__ import division
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import statsmodels.api as sm
import pylab as pl
import numpy as np
%matplotlib inline
```

Step 2: Read in your data set

In [2]:
```
#Read in data from source
df_raw = pd.read_csv("../assets/admissions.csv")
print df_raw.head()
```

```
   admit  gre   gpa  prestige
0      0  380  3.61         3
1      1  660  3.67         3
2      1  800  4.00         1
3      1  640  3.19         4
4      0  520  2.93         4
```

Questions

Question 1. How many observations are in our dataset?

In [3]:
```
df_raw.count()
```

Out[3]:
```
admit       400
gre         398
gpa         398
prestige    399
dtype: int64
```

Answer: 400

Question 2. Create a summary table

In [4]:
```
print df_raw.describe()
```

```
            admit         gre         gpa    prestige
count  400.000000  398.000000  398.00000  399.000000
mean     0.317500  588.040201    3.39093    2.486216
std      0.466087  115.628513    0.38063    0.945333
min      0.000000  220.000000    2.26000    1.000000
25%      0.000000  520.000000    3.13000    2.000000
50%      0.000000  580.000000    3.39500    2.000000
75%      1.000000  660.000000    3.67000    3.000000
max      1.000000  800.000000    4.00000    4.000000
```

Unit Project 2

Question 3. Why would GRE have a larger STD than GPA?

Answer: Because it has a much larger range/mean.

Question 4. Drop data points with missing data

In [6]:
```
df = df_raw.dropna()
print df.count()
```

```
admit       397
gre         397
gpa         397
prestige    397
dtype: int64
```

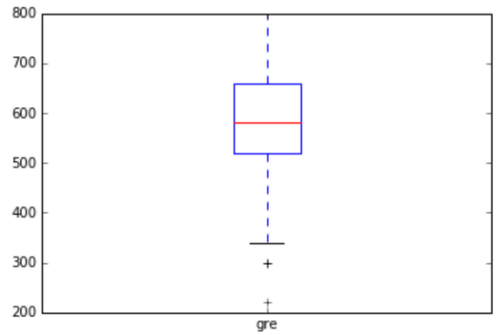
Question 5. Confirm that you dropped the correct data. How can you tell?

Answer: Depending on the method, you may need to do more or less checking. In this case, using dropna() is good so just count() is a useful check.

Question 6. Create box plots for GRE and GPA

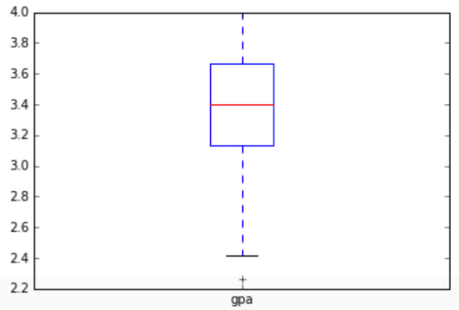
In [7]:
```
#boxplot 1
df['gre'].plot(kind='box')
# Alternative method
#df.boxplot(column='gre', return_type='axes')
```

Out[7]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7ff282971b90>
```



In [8]:
```
#boxplot 2
df['gpa'].plot(kind='box')
# Alternative method
#df.boxplot(column='gpa', return_type='axes')
```

Out[8]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7ff282971bd0>
```



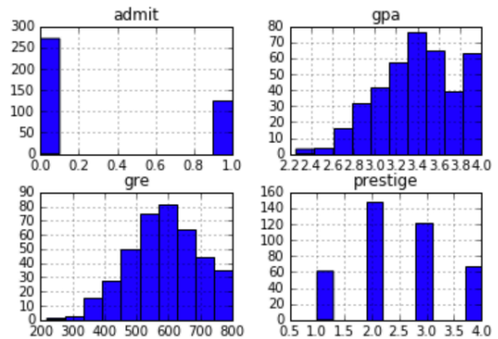
Unit Project 2

Question 7. What do this plots show?

Answer: min, max, median, 25%, 75%, outliers

Question 8. Describe each distribution

In [9]:
```
# plot the distribution of each variable
df.hist()
pl.show()
```



Question 9. If our model had an assumption of a normal distribution would we meet that requirement?

Answer: Yes, GPA and GRE are skewed toward the upper ranges.

Question 10. Does this distribution need correction? If so, why? How?

Answer: Though slightly skewed, in this setting there is no need to correct anything.

Question 11. Which of our variables are potentially colinear?

In [10]:
```
# create a correlation matrix for the data
df.corr()
```

Out[10]:

	admit	gre	gpa	prestige
admit	1.000000	0.181202	0.174116	-0.243563
gre	0.181202	1.000000	0.382408	-0.124533
gpa	0.174116	0.382408	1.000000	-0.060976
prestige	-0.243563	-0.124533	-0.060976	1.000000

Question 12. What did you find?

Answer: Colinearity is not a concern here.

Question 13. Write an analysis plan for exploring the association between grad school admissions rates and prestige of undergraduate schools.

Answer: 1) Do a crude analysis with the following model P(admit = 1) = alpha + beta(prestige) 2) Repeat controlling for gre and gpa

Question 14. What is your hypothesis?

Answer: Students who attended a more prestigious undergraduate institution are more likely to be admitted to graduate school.

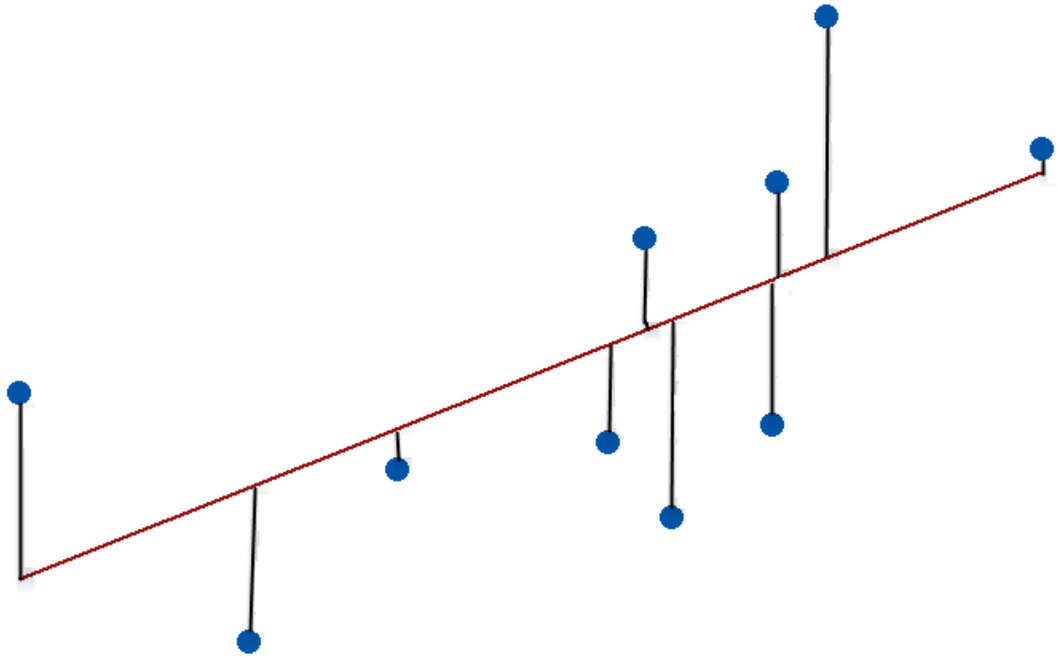
LEARNING OBJECTIVES

- Define R-squares and residuals
- Define bias and error metrics for regression problems
- Understand cross validation
- Define regularization and grid search
- Evaluate model fit using Gradient Descent

OPENING

R-SQUARES AND RESIDUALS

WHAT IS A RESIDUAL?



Residuals are the distance between the observed value and the fitted value.

WHAT IS R-SQUARED?

- R-squared is the percentage of the dependent variable variation that a linear model explains.

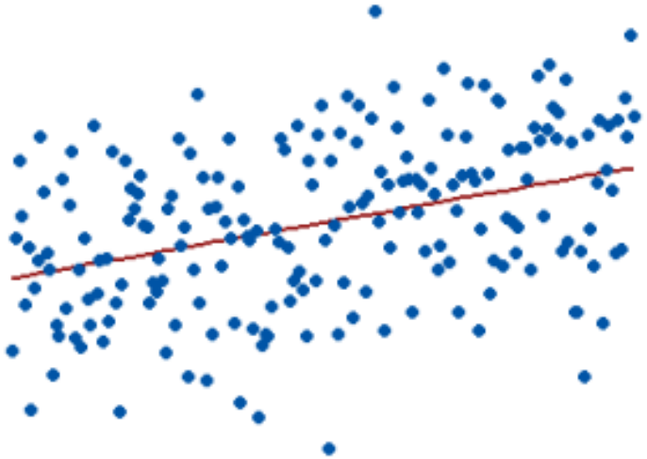
$$R^2 = \frac{\text{Variance explained by the model}}{\text{Total variance}}$$

- R-squared is always between 0 and 100%

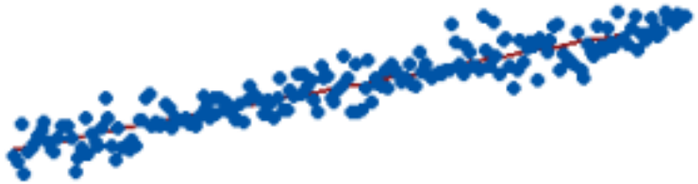
- 0% represents a model that does not explain any of the variation in the response variable around its mean.

- 100% represents a model that explains all of the variation in the response variable around its mean.

WHAT IS R-SQUARED?



R-squared = 15%

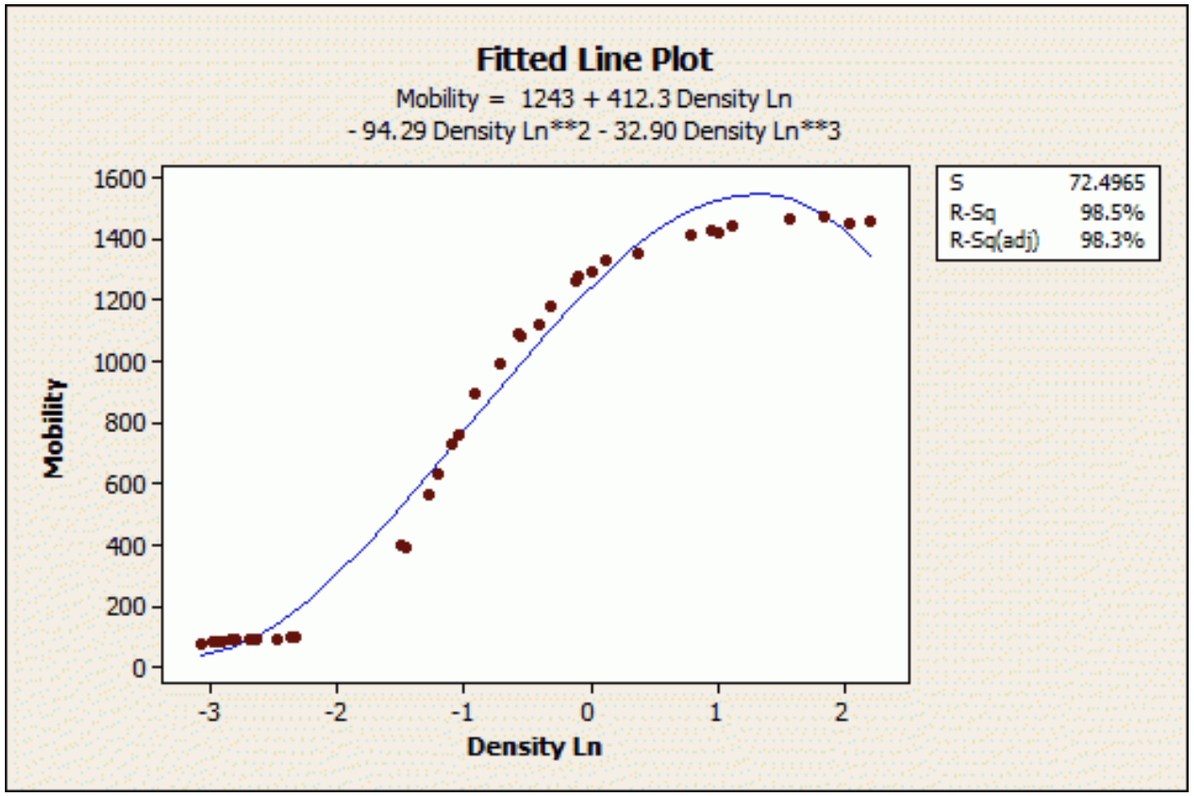


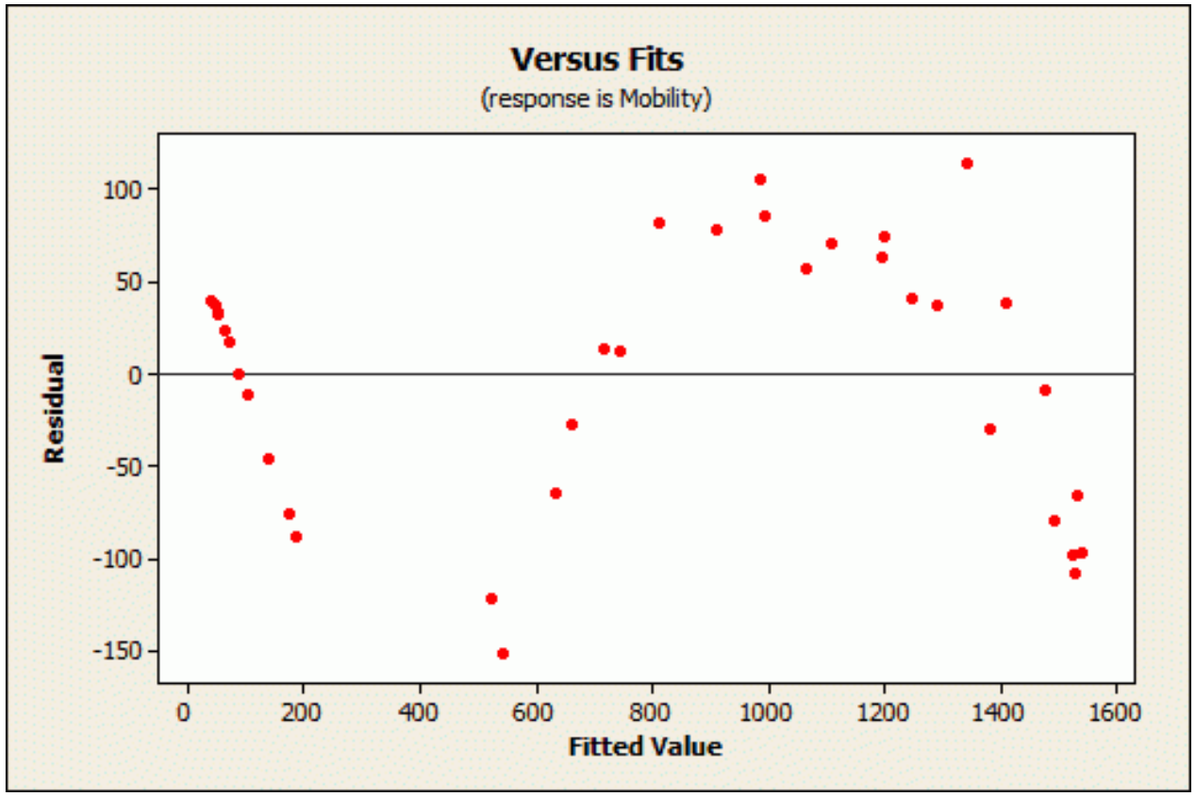
R-squared = 85%

WHAT IS R-SQUARED? WHAT IS A RESIDUAL?

- R-squared, the central metric introduced for linear regression
- Which model performed better, one with an r-squared of 0.79 or 0.81?
- R-squared measures explain variance.
- But does it tell the magnitude or scale of error?
- You **cannot** use R-squared to determine whether the predictions are biased, which is why you must assess the residual plots.

WHAT IS R-SQUARED? WHAT IS A RESIDUAL?





TODAY'S CLASS: EVALUATING MODEL FIT

LEARNING OBJECTIVES

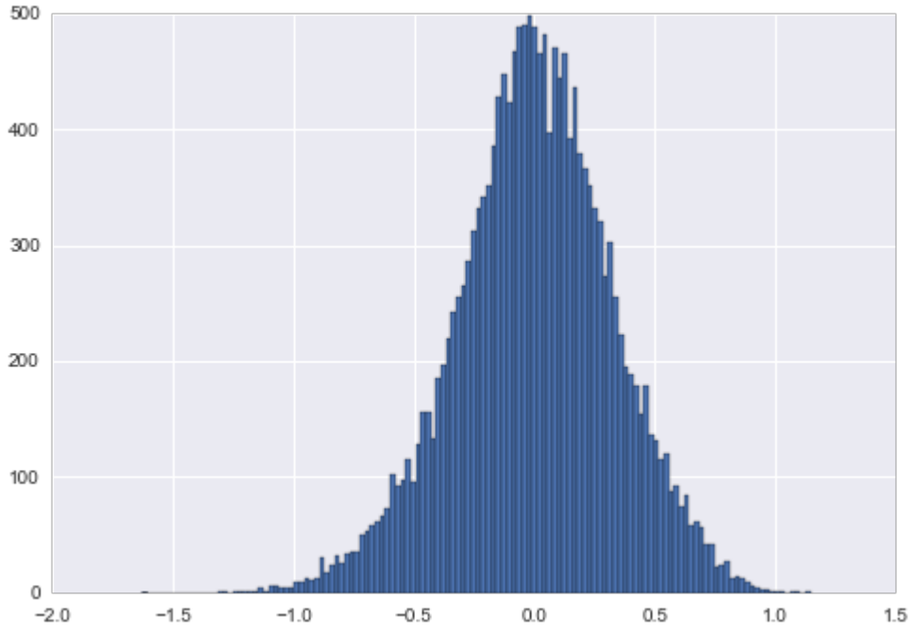
- ~~Define R-squares and residuals~~
- Define bias and error metrics for regression problems
- Understand cross validation
- Define regularization and grid search
- Evaluate model fit using Gradient Descent

INTRODUCTION

LINEAR MODELS AND ERROR

RECALL: WHAT'S RESIDUAL ERROR?

- In linear models, residual error must be normal with a median close to zero.
- Individual residuals are useful to see the error of specific points, but it doesn't provide an overall picture for optimization.
- We need a metric to summarize the error in our model into one value.
- Mean square error: the mean residual error in our model



MEAN SQUARED ERROR (MSE)

- To calculate MSE:
  - Calculate the difference between each target y and the model's predicted value y-hat (i.e. the residual)
  - Square each residual.
  - Take the mean of the squared residual errors.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \tilde{y}_i)^2$$

MEAN SQUARED ERROR (MSE)

- Follow me in **Part 1** of the starter code
- sklearn's metrics module includes a mean_squared_error function.

```
from sklearn import metrics
metrics.mean_squared_error(y, model.predict(X))
```

MEAN SQUARED ERROR (MSE)

- For example, two arrays of the same values would have an MSE of 0.

```
from sklearn import metrics
metrics.mean_squared_error([1, 2, 3, 4, 5], [1, 2, 3, 4, 5])
0.0
```

MEAN SQUARED ERROR (MSE)

- Two arrays with different values would have a positive MSE.

```
from sklearn import metrics
metrics.mean_squared_error([1, 2, 3, 4, 5], [5, 4, 3, 2, 1])
# (4^2 + 2^2 + 0^2 + 2^2 + 4^2) / 5
8.0
```

OTHER METRICS

Mean squared error	$\text{MSE} = \frac{1}{n}\sum_{t=1}^{n} e_t^2$
Root mean squared error	$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{t=1}^{n} e_t^2}$
Mean absolute error	$\text{MAE} = \frac{1}{n}\sum_{t=1}^{n} \lvert e_t \rvert$

Comparing these metrics:

- MAE is the easiest to understand, because it's the average error.
- MSE is more popular than MAE, because MSE "punishes" larger errors.
- RMSE is even more popular than MSE, since RMSE is interpretable in the "y" units.

HOW DO WE MINIMIZE ERROR?

- The regression method we've used is called "Ordinary Least Squares".
- This means that given a matrix X, solve for the *least* amount of square error for y.
- However, this assumes that X is unbiased, that it is representative of the population.

LET'S COMPARE TWO RANDOM MODELS

```
import numpy as np
import pandas as pd
from sklearn import linear_model

df = pd.DataFrame({'x': range(100), 'y': range(100)})
biased_df  = df.copy()
biased_df.loc[:20, 'x'] = 1
biased_df.loc[:20, 'y'] = 1

def append_jitter(series):
    jitter = np.random.random_sample(size=100)
    return series + jitter
```

LET'S COMPARE TWO RANDOM MODELS

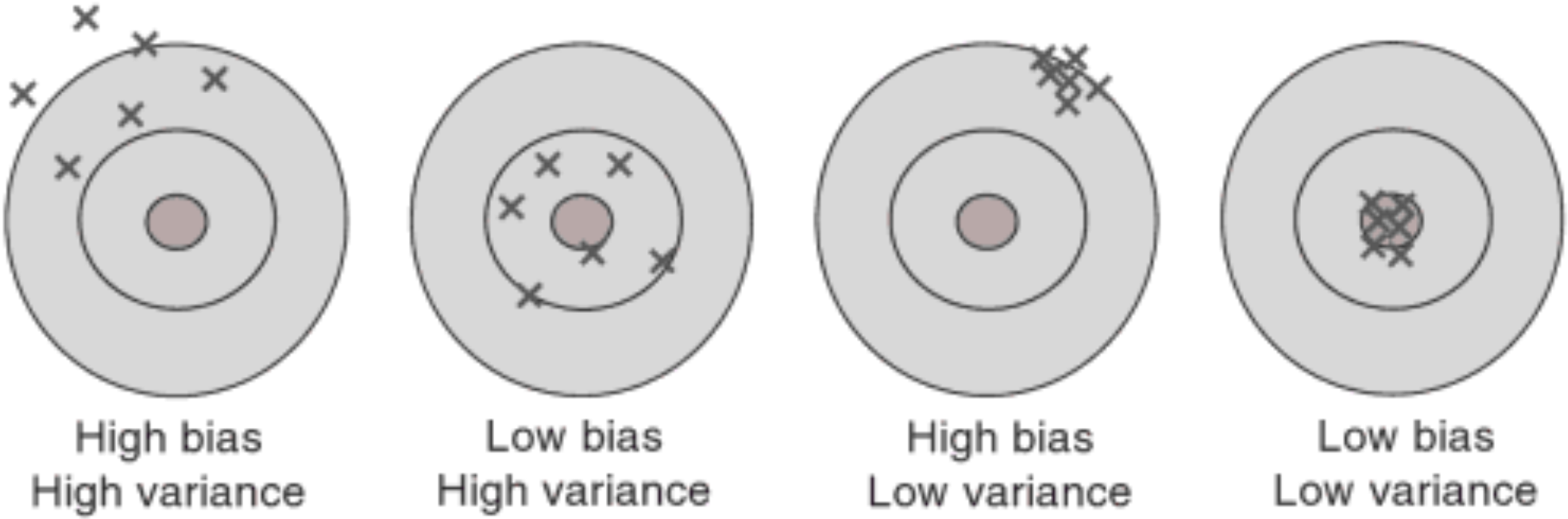
```
df['x'] = append_jitter(df.x)
df['y'] = append_jitter(df.y)

biased_df['x'] = append_jitter(biased_df.x)
biased_df['y'] = append_jitter(biased_df.y)

- Fit:
lm = linear_model.LinearRegression().fit(df[['x']], df['y'])
print metrics.mean_squared_error(df['y'], lm.predict(df[['x']]))

- Biased fit:
lm = linear_model.LinearRegression().fit(biased_df[['x']], biased_df['y'])
print metrics.mean_squared_error(df['y'], lm.predict(df[['x']]))
```

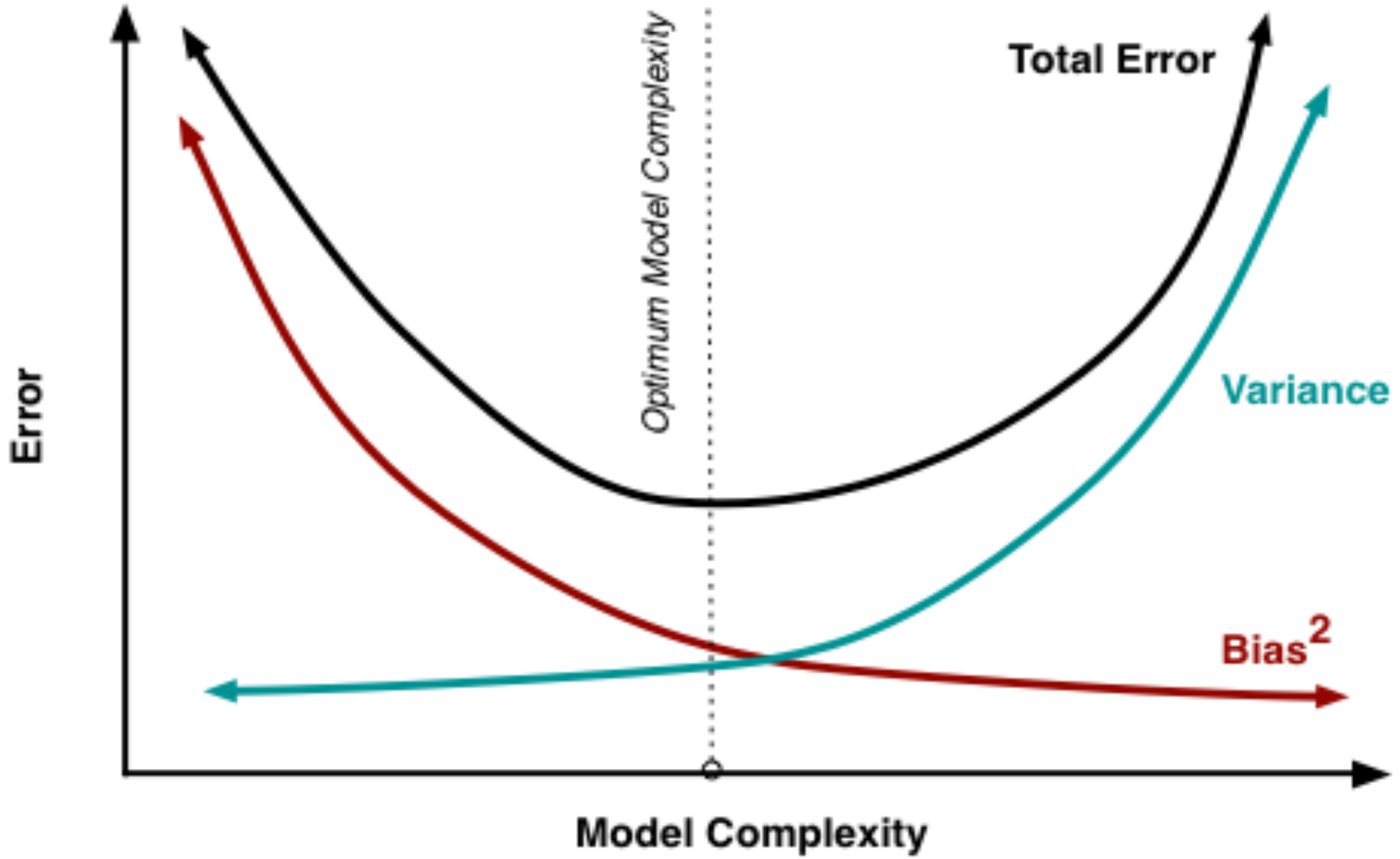
BIAS VS. VARIANCE



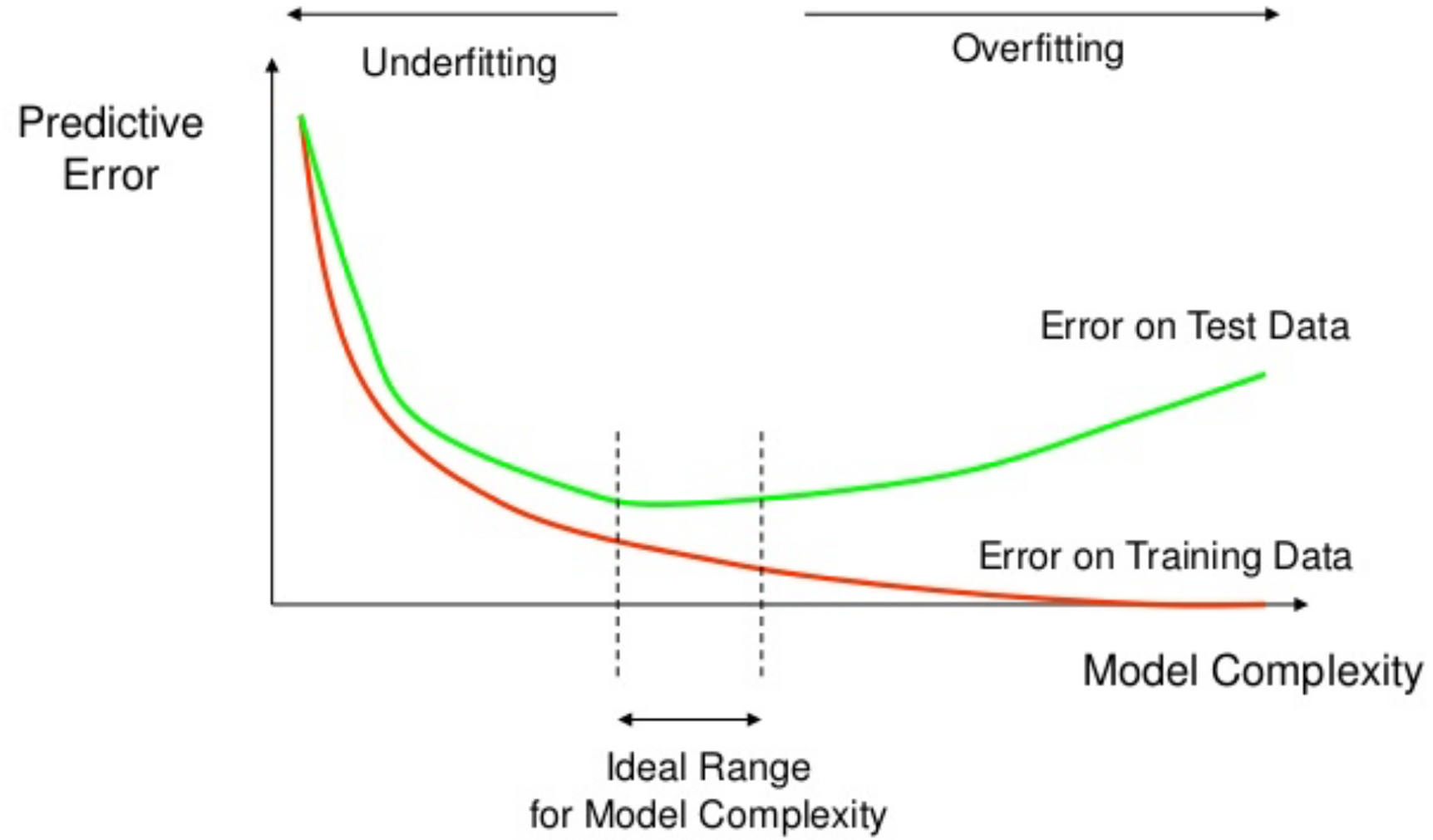
BIAS VARIANCE TRADEOFF

- When our error is ***biased***, it means the model's prediction is consistently far away from the actual value.
- This could be a sign of poor sampling and poor data.
- One objective of a biased model is to trade bias error for generalized error. We prefer the error to be more evenly distributed across the model.
- This is called error due to ***variance***.
- We want our model to *generalize* to data it hasn't seen even if doesn't perform as well on data it has already seen.

BIAS VARIANCE TRADEOFF



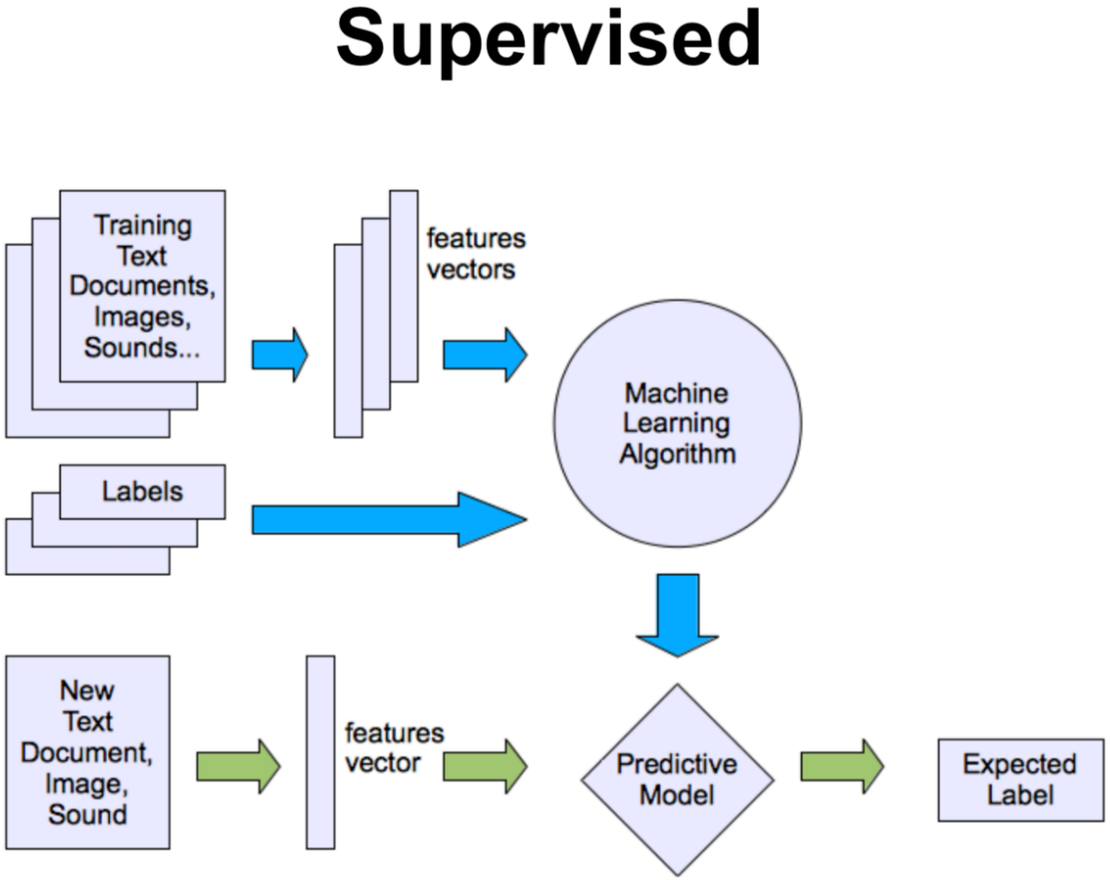
BIAS VARIANCE TRADEOFF



EXERCISE

TRAINING YOUR FIRST MACHINE LEARNING MODEL

ADVERTISING DATA



Follow **Part 2** of the starter code

TRAINING/TESTING

What if you was ”lucky” and got a low error when you split the data into training and testing sets?

LEARNING OBJECTIVES

- ~~Define R-squares and residuals~~
- ~~Define bias and error metrics for regression problems~~
- Understand cross validation
- Define regularization and grid search
- Evaluate model fit using Gradient Descent

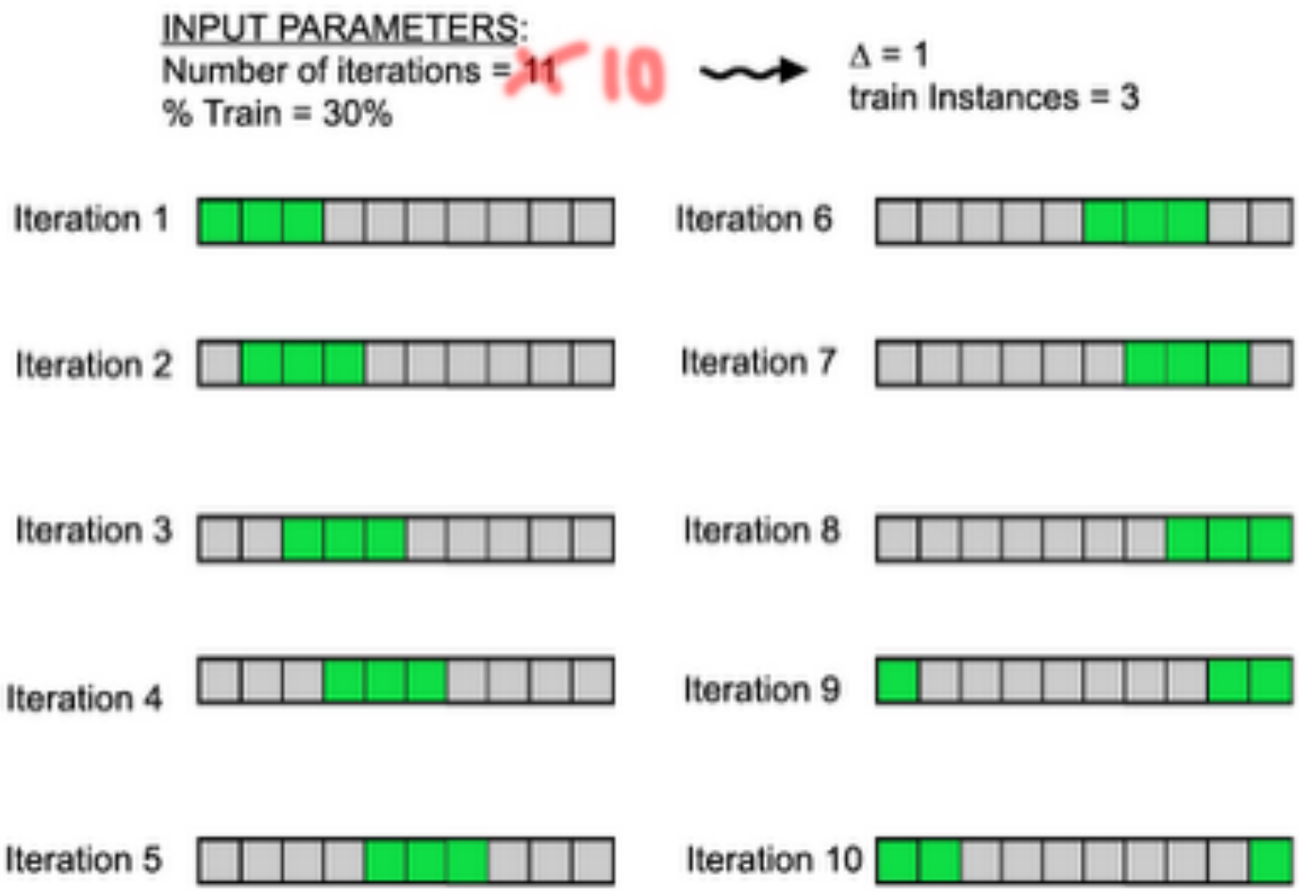
DEMO

CROSS VALIDATION

CROSS VALIDATION

- Cross validation can help account for bias.
- The general idea is to
  - Generate several models on different cross sections of the data
  - Measure the performance of each
  - Take the mean performance
- This technique swaps bias error for generalized error, describing previous trends accurately enough to extend to future trends.

CROSS VALIDATION



K-FOLD CROSS VALIDATION

- k-fold cross validation
  - Split the data into *k* group
  - Train the model on all segments except one
  - Test model performance on the remaining set
- If k = 5, split the data into five segments and generate five models.

USING K-FOLD CROSS VALIDATION WITH MSE

- Follow **Part 3** of the starter code

- Import the appropriate packages and load data.

```
from sklearn import cross_validation
wd = '../../datasets/'
bikeshare = pd.read_csv(wd + 'bikeshare/bikeshare.csv')
weather = pd.get_dummies(bikeshare.weathersit, prefix='weather')
modeldata = bikeshare[['temp', 'hum']].join(weather[['weather_1',
'weather_2', 'weather_3']])
y = bikeshare.casual
```

USING K-FOLD CROSS VALIDATION WITH MSE

- Build models on subsets of the data and calculate the average score.

```
kf = cross_validation.KFold(len(modeldata), n_folds=5, shuffle=True)
scores = []
for train_index, test_index in kf:
    lm = linear_model.LinearRegression().fit(modeldata.iloc[train_index], y.iloc[train_index])
    scores.append(metrics.mean_squared_error(y.iloc[test_index], lm.predict(modeldata.iloc[test_index])))

print np.mean(scores)
```

USING K-FOLD CROSS VALIDATION WITH MSE

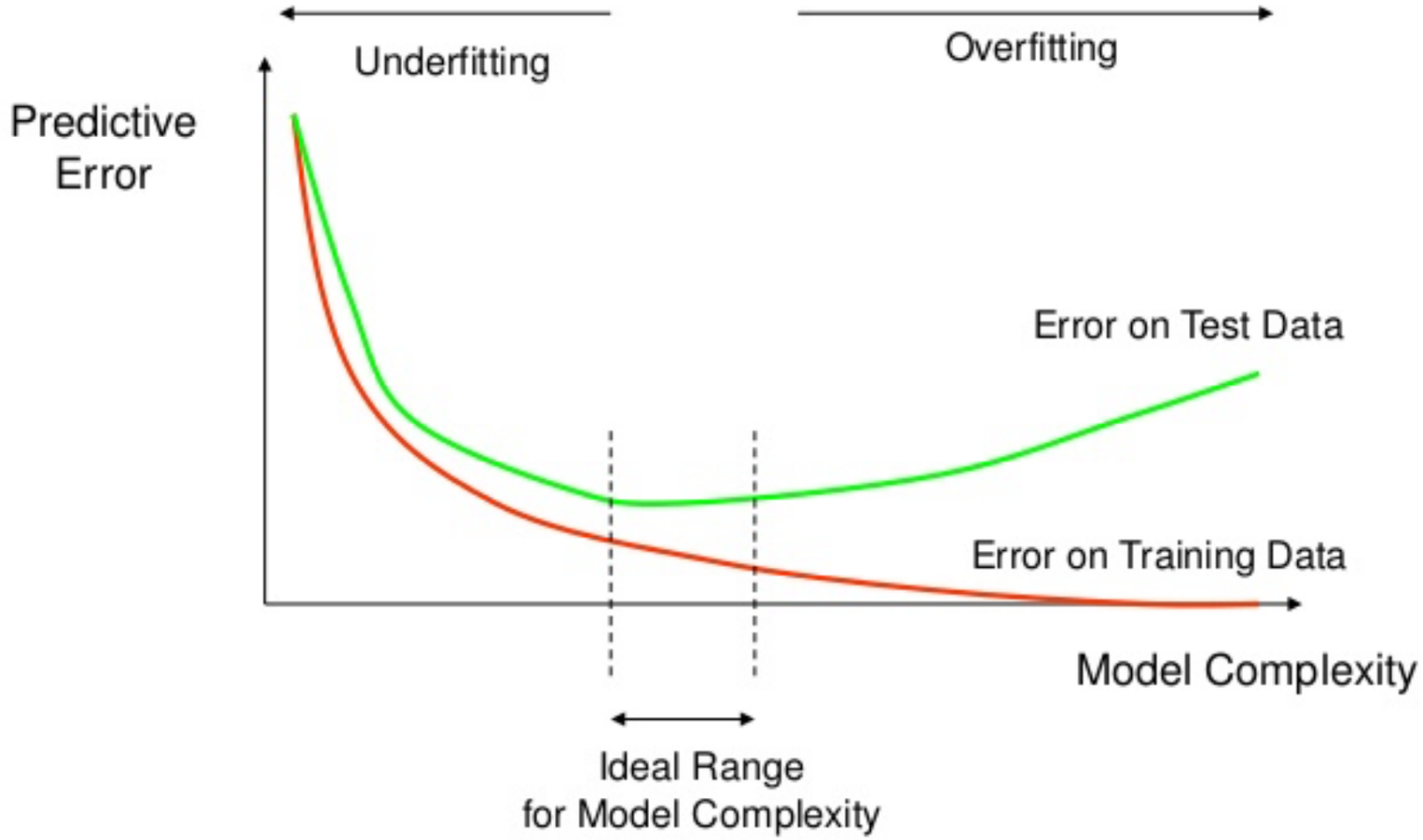
- This can be compared to the model built on all of the data.

```
- This score will be lower, but we're trading off bias error for
generalized error:
lm = linear_model.LinearRegression().fit(modeldata, y)
print metrics.mean_squared_error(y, lm.predict(modeldata))
```

- Which approach would predict new data more accurately?

- Follow **Part 4** of the starter code.

BIAS VARIANCE TRADEOFF



How to protect my model against overfitting?

LEARNING OBJECTIVES

- ~~Define R-squares and residuals~~
- ~~Define bias and error metrics for regression problems~~
- ~~Understand cross validation~~
- Define regularization and grid search
- Evaluate model fit using Gradient Descent

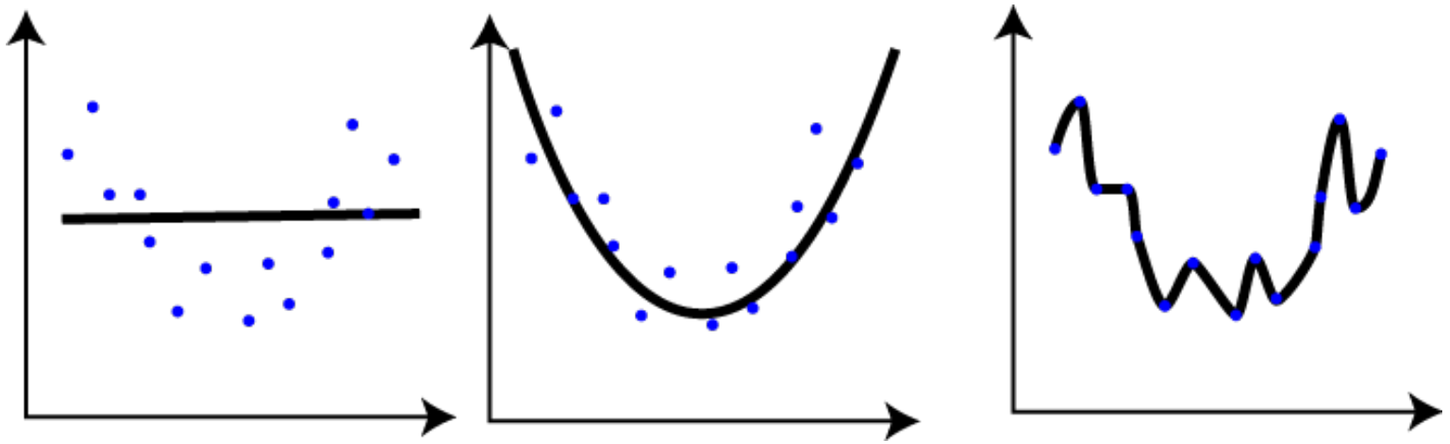
INTRODUCTION

REGULARIZATION AND CROSS VALIDATION

WHAT IS REGULARIZATION? AND WHY DO WE USE IT?

- Regularization is an additive approach to protect models against overfitting (being potentially biased and overconfident, not generalizing well).
- Regularization becomes an additional weight to coefficients, shrinking them closer to zero.
- L1 (Lasso Regression) adds the extra weight to coefficients.
- L2 (Ridge Regression) adds the square of the extra weight to coefficients.
- Use Lasso when we have more features than observations (k > n) and Ridge otherwise.

WHAT IS OVERFITTING?



- The first model poorly explains the data.
- The second model explains the general curve of the data.
- The third model drastically overfits the model, bending to every point.
- Regularization helps prevent the third model.

WHERE REGULARIZATION MAKES SENSE

- Follow **Part 5** of the starter

- What happens to MSE if use Lasso or Ridge Regression directly?

```
lm = linear_model.LinearRegression().fit(modeldata, y)
print metrics.mean_squared_error(y, lm.predict(modeldata))
lm = linear_model.Lasso().fit(modeldata, y)
print metrics.mean_squared_error(y, lm.predict(modeldata))
lm = linear_model.Ridge().fit(modeldata, y)
print metrics.mean_squared_error(y, lm.predict(modeldata))
```

```
1672.58110765 # OLS
1725.41581608 # L1
1672.60490113 # L2
```

WHERE REGULARIZATION MAKES SENSE

- It doesn't seem to help. Why is that?

- We need to optimize the regularization weight parameter (called alpha) through cross validation.

ACTIVITY: KNOWLEDGE CHECK

ANSWER THE FOLLOWING QUESTIONS (5 minutes)

1. Why is regularization important?
2. What does it protect against and how?

DELIVERABLE

Answers to the above questions

DEMO

UNDERSTANDING REGULARIZATION EFFECTS

QUICK CHECK

- We are working with the bikeshare data to predict riders over hours/days with a few features.
- Does it make sense to use a ridge regression or a lasso regression?
- Why?
- Follow **Part 6** of the starter code.

UNDERSTANDING REGULARIZATION EFFECTS

- Let's test a variety of alpha weights for Ridge Regression on the bikeshare data.

```
alphas = np.logspace(-10, 10, 21)
for a in alphas:
    print 'Alpha:', a
    lm = linear_model.Ridge(alpha=a)
    lm.fit(modeldata, y)
    print lm.coef_
    print metrics.mean_squared_error(y, lm.predict(modeldata))
```

- What happens to the weights of the coefficients as alpha increases? What happens to the error as alpha increases?

WE CAN MAKE THIS EASIER WITH GRID SEARCH!

- Grid search exhaustively searches through all given options to find the best solution. Grid search will try all combos given in `param_grid`.

```
param_ grid = {
    'intercept': [True, False],
    'alpha': [1, 2, 3],
}
```

WE CAN MAKE THIS EASIER WITH GRID SEARCH!

- This param grid has six different options:
  - intercept True, alpha 1
  - intercept True, alpha 2
  - intercept True, alpha 3
  - intercept False, alpha 1
  - intercept False, alpha 2
  - intercept False, alpha 3

```
param_ grid = {
    'intercept': [True, False],
    'alpha': [1, 2, 3],
}
```

WE CAN MAKE THIS EASIER WITH GRID SEARCH!

- This is an incredibly powerful, automated machine learning tool!

```
from sklearn import grid_search

alphas = np.logspace(-10, 10, 21)
gs = grid_search.GridSearchCV(
    estimator=linear_model.Ridge(),
    param_grid={'alpha': alphas},
    scoring='mean_squared_error')
```

WE CAN MAKE THIS EASIER WITH GRID SEARCH!

```
gs.fit(modeldata, y)

print -gs.best_score_ # mean squared error here comes in negative, so let's make it positive.
print gs.best_estimator_ # explains which grid_search setup worked best
print gs.grid_scores_ # shows all the grid pairings and their performances.
```

TODAY'S CLASS: EVALUATING MODEL FIT

LEARNING OBJECTIVES

- ~~Define R-squares and residuals~~
- ~~Define bias and error metrics for regression problems~~
- ~~Understand cross validation~~
- ~~Define regularization and grid search~~
- Evaluate model fit using Gradient Descent

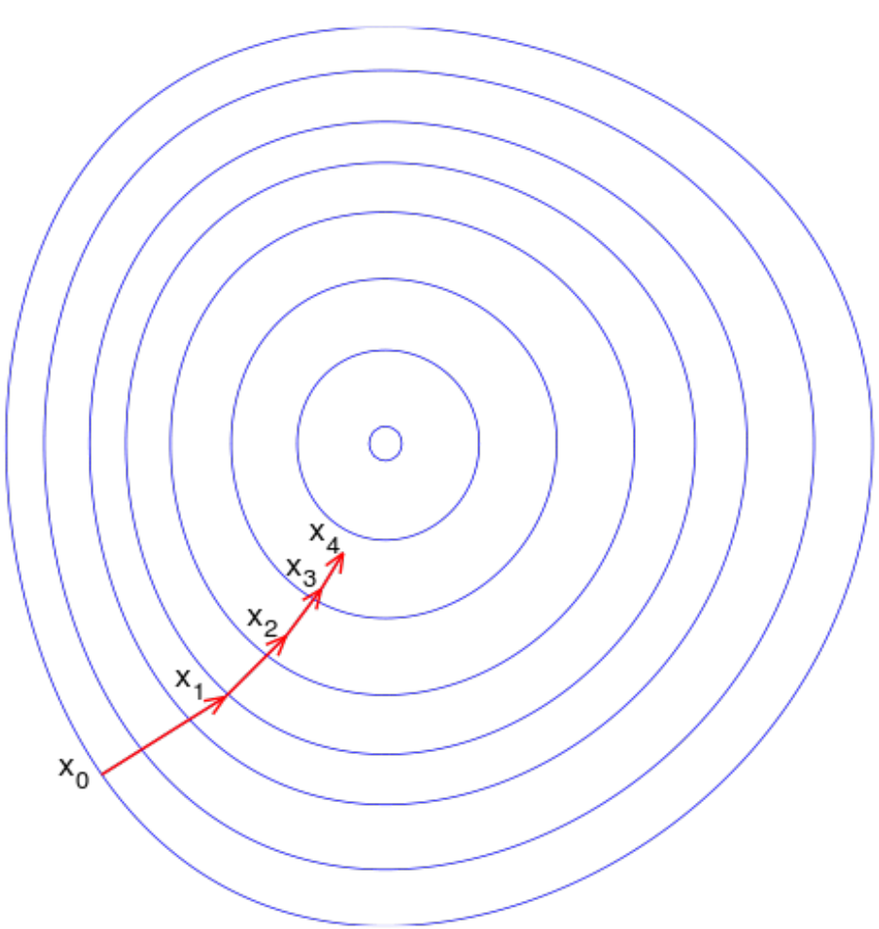
INTRODUCTION

MINIMIZING LOSS THROUGH GRADIENT DESCENT

GRADIENT DESCENT

- Gradient Descent can also help us minimize error.
- Follow **Part 7** of the starter code
- How Gradient Descent works:
  - A random linear solution is provided as a starting point
  - The solver attempts to find a next "step": take a step in any direction and measure the performance.
  - If the solver finds a better solution (i.e. lower MSE), this is the new starting point.
  - Repeat these steps until the performance is optimized and no "next steps" perform better. The size of steps will shrink over time.

GRADIENT DESCENT



A CODE EXAMPLE OF GRADIENT DESCENT

```
num_to_approach, start, steps, optimized = 6.2, 0., [-1, 1], False
while not optimized:
    current_distance = num_to_approach - start
    got_better = False
    next_steps = [start + i for i in steps]
    for n in next_steps:
        distance = np.abs(num_to_approach - n)
        if distance < current_distance:
            got_better = True
            print distance, 'is better than', current_distance
            current_distance = distance
            start = n
```

A CODE EXAMPLE OF GRADIENT DESCENT

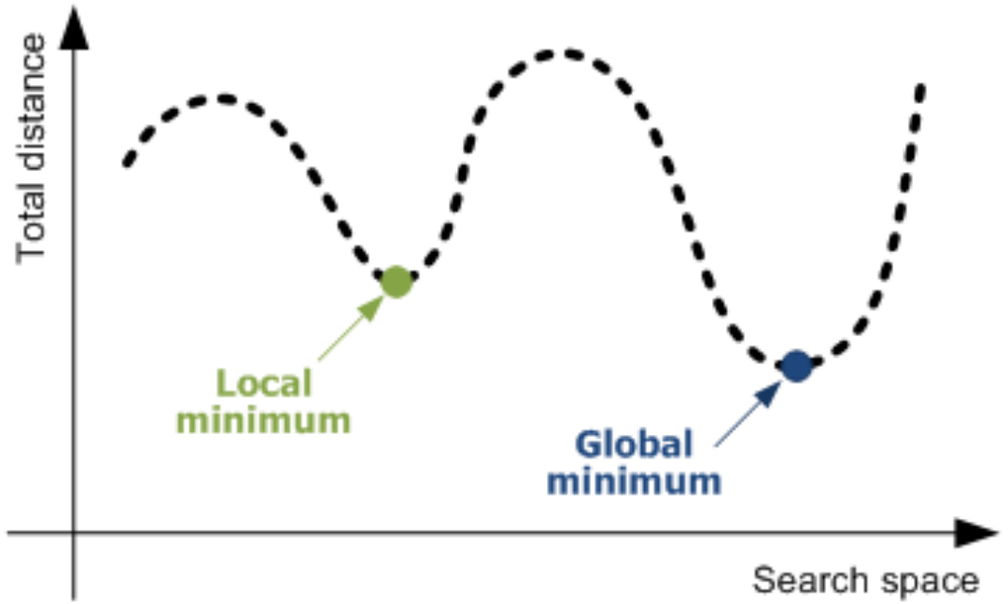
```
    if got_better:
        print 'found better solution! using', current_distance
        a += 1
    else:
        optimized = True
        print start, 'is closest to', num_to_approach
```

- What is the code doing? What could go wrong?

GLOBAL VS LOCAL MINIMUMS

- Gradient Descent could solve for a *local* minimum instead of a *global* minimum.

- A *local* minimum is confined to a very specific subset of solutions. The *global* minimum considers all solutions. These could be equal, but that's not always true.



DEMO

APPLICATION OF GRADIENT DESCENT

APPLICATION OF GRADIENT DESCENT

- Gradient Descent works best when:
  - We are working with a large dataset. Smaller datasets are more prone to error.
  - Data is cleaned up and normalized.
- Gradient Descent is significantly faster than OLS. This becomes important as data gets bigger.

APPLICATION OF GRADIENT DESCENT

- We can easily run a Gradient Descent regression.

- Note: The verbose argument can be set to 1 to see the optimization steps.

```
lm = linear_model.SGDRegressor()
lm.fit(modeldata, y)
print lm.score(modeldata, y)
print metrics.mean_squared_error(y, lm.predict(modeldata))
```

- Untuned, how well did gradient descent perform compared to OLS?

APPLICATION OF GRADIENT DESCENT

- Gradient Descent can be tuned with
  - the learning rate: how aggressively we solve the problem
  - epsilon: at what point do we say the error margin is acceptable
  - iterations: when should be we stop no matter what

INDEPENDENT PRACTICE

ON YOUR OWN

ACTIVITY: ON YOUR OWN

DIRECTIONS (30 minutes)

There are tons of ways to approach a regression problem.

1. Implement the Gradient Descent approach to our bikeshare modeling problem.
2. Show how Gradient Descent solves and optimizes the solution.
3. Demonstrate the grid_search module.
4. Use a model you evaluated last class or the simpler one from today. Implement param_grid in grid search to answer the following questions:
    a. With a set of values between 10^-10 and 10^-1, how does MSE change?
    b. Our data suggests we use L1 regularization. Using a grid search with l1_ratios between 0 and 1, increasing every 0.05, does this statement hold true? If not, did gradient descent have enough iterations to work properly?
    c. How do these results change when you alter the learning rate?

DELIVERABLE

Gradient Descent approach and answered questions

ACTIVITY: ON YOUR OWN

Starter Code

```
params = {} # put your gradient descent parameters here
gs = grid_search.GridSearchCV(
    estimator=linear_model.SGDRegressor(),
    cv=cross_validation.KFold(len(modeldata), n_folds=5, shuffle=True),
    param_grid=params,
    scoring='mean_squared_error',
    )

gs.fit(modeldata, y)

print 'BEST ESTIMATOR'
print -gs.best_score_
print gs.best_estimator_
print 'ALL ESTIMATORS'
print gs.grid_scores_
```

TODAY'S CLASS: EVALUATING MODEL FIT

LEARNING OBJECTIVES

- ~~Define R-squares and residuals~~
- ~~Define bias and error metrics for regression problems~~
- ~~Understand cross validation~~
- ~~Define regularization and grid search~~
- ~~Evaluate model fit using Gradient Descent~~

CONCLUSION

TOPIC REVIEW

LESSON REVIEW

- What's the (typical) range of r-squared?
- What's the range of mean squared error?
- How would changing the scale or interpretation of y (your target variable) effect mean squared error?
- What's cross validation, and why do we use it in machine learning?
- What is error due to bias? What is error due to variance? Which is better for a model to have, if it had to have one?
- How does gradient descent try a different approach to minimizing error?

COURSE

BEFORE NEXT CLASS

OUR PROGRESS SO FAR

UNIT 1: RESEARCH DESIGN AND EXPLORATORY DATA ANALYSIS	~~What is Data Science~~	~~Lesson 1~~
	~~Research Design and Pandas~~	~~Lesson 2~~
	~~Statistics Fundamentals I~~	~~Lesson 3~~
	~~Statistics Fundamentals II~~	~~Lesson 4~~
	~~Flexible Class Session~~	~~Lesson 5~~
UNIT 2: FOUNDATIONS OF DATA MODELING	~~Introduction to Regression~~	~~Lesson 6~~
	~~Evaluating Model Fit~~	~~Lesson 7~~
	Introduction to Classification	Lesson 8
	Introduction to Logistic Regression	Lesson 9
	Communicating Logistic Regression Results	Lesson 10
	Flexible Class Session	Lesson 11
UNIT 3: DATA SCIENCE IN THE REAL WORLD	Decision Trees and Random Forests	Lesson 12
	Natural Language Processing	Lesson 13
	Dimensionality Reduction	Lesson 14
	Time Series Data I	Lesson 15
	Time Series Data II	Lesson 16
	Database Technologies	Lesson 17
	Where to Go Next	Lesson 18
	Flexible Class Session	Lesson 19
	Final Project Presentations	Lesson 20

BEFORE NEXT CLASS

DUE DATE

- Project: Final Project, Deliverable 1

LESSON

Q & A

LESSON

EXIT TICKET

DON’T FORGET TO FILL OUT YOUR EXIT TICKET