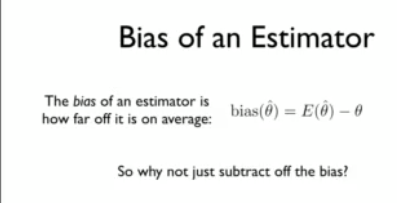
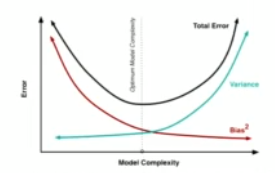
# Bias and Regression

## Some forms of bias

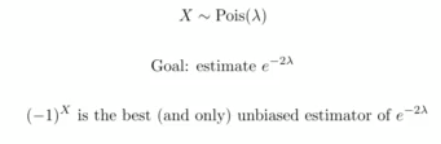
* Selection
* Publication bias (file drawer problem)
* Non-response bias
* Length bias
* Class Size Paradox
  + Why do so many schools boast small average class size, but then so many students end up in large classes?
  + It depends on your vantage point.
    - Dean calculates (100 + 50\*2)/51 = 3.92
    - Students calculates (100 \* 100 + 100 \* 2)/200 = 51

## Bias of an Estimator

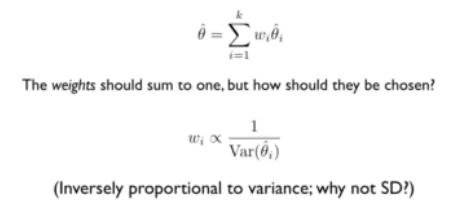


* Bias-Variance Tradeoff
  + One form: MSE(x) = Var(x) + bias^2(x)
  + Often a little bit of bias can make it possible to have much lower MSE
  + 

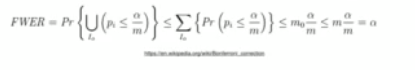
## Unbiased Estimation: Poisson Example

* 

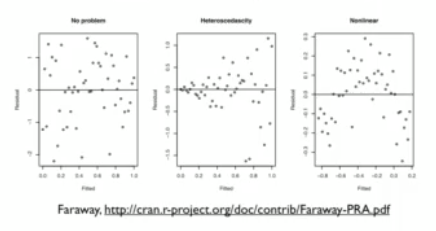
## Fisher Weighting

* How should we combine independent, unbiased estimators for a parameter into one estimator?
* 

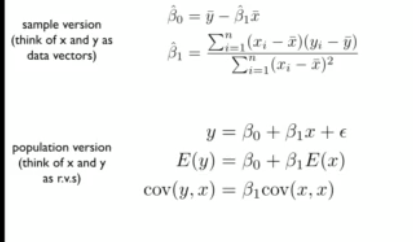
## Multiple Testing, Bonferroni

* How should we handle p-values when testing multiple hypotheses?
  + For example, what if we are looking at diet (with 10 kinds of food) and disease (with 10 diseases)?
* A simple, conservative approach is Bonferroni: divide significance level by number of hypotheses being tested
  + You’re controlling for FWER (family wide error rate)
    - 

## Linear Model

* Often called ‘OLS’, but that puts the focus on the procedure rather than the model.
* Standardizing all variables can sometimes reduce interpretability
* Always do residual plots!
  + Plot residuals vs fitted values
  + Residuals vs each predictor variable
  + 
  + You’re hoping for no pattern!

## Sample Quantities vs Population Quantities

* 

## Collinearity

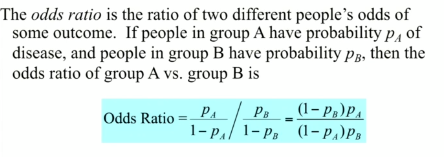
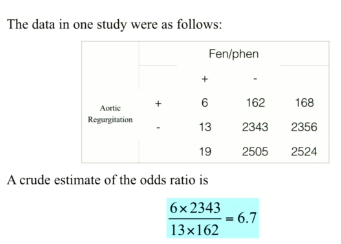
* Should avoid having predictor variables that are highly correlated with each other
  + Collinearity results in instability, high variances in estimates, and worse interpretability
* An extreme case of collinarity would be also including a Bronx indicator in the NYC housing example. Instead, use one borough as a baseline.

## Predicting a Binary Response

* Linear regression is a very poor tool for this.
* Logistic regression is much better

## Measures of Association

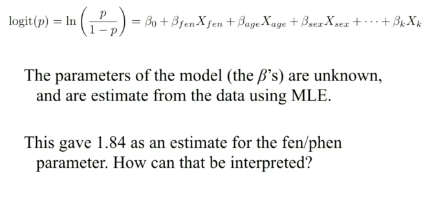
### Odds Ratio

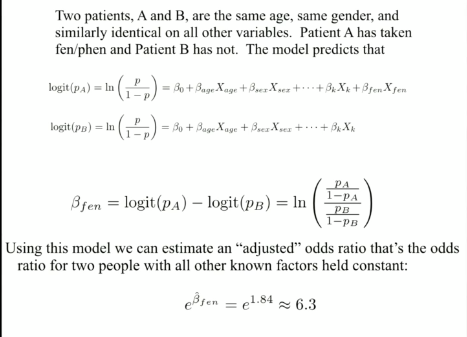
* If someone’s probability of experiencing an outcome is p, then that person’s odds of the outcome are p/(1-p)
* The odds ratio is the ratio of two different people’s odds of some outcome.
  + 
* Crude odds ratio estimate
  + 

## What about confounding factors?

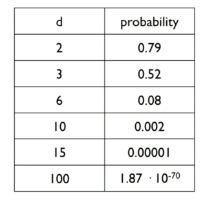
* For example, what if fen/phen users are more likely to be obese, and obesity increases the risk of heart disease?
* We can set up a logistic regression model to predict a person’s odds of heart disease, given the predictor variables
* We can also use this to compare fen/phen users vs non-fen/phen users, controlling for the other predictors (probit model)
* Then we can use the data to estimate the parameters, using Maximum Likelihood Estimation (MLE)

## A logistic Regression Model





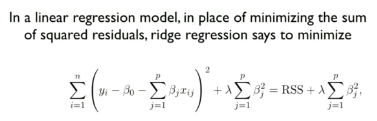
## The Curse of Dimensionality

* For a uniformly random point in the box in d dimensions with length 2 in each dimension, what is the probability that the random vector is in the unit ball in d dimensions?
  + 

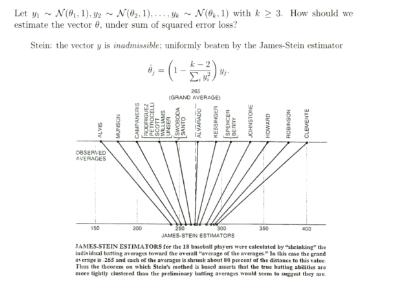
## Methods for dealing with high dimensional datasets

* Linear regression breaks down when dealing with high dimensional datasets so we have to use alternative methods.

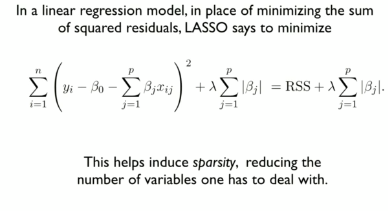
### Ridge regression and shrinkage

* 

### Stein’s Paradox and Shrinkage Estimation

* 

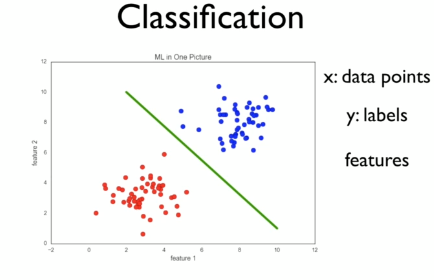
### LASSO and Sparsity



# Classification , KNN, Cross-Validation, Dimensionality Reduction

## Classification

* Choice of feature is one of the most important things you can do.



### Problems

* Viewpoint variation
* Scale variation
* Deformation
* Occlusion
* Background clutter
* Intra-class variation
* Illumination conditions

## Machine Learning

* Input: A training set of N data points, each labeled with one of K different classes.
* Learning: Use the training set to learn what every one of the classes looks like.
* Evaluation: Predict labels for a test set of data and compare the true labels (ground truth) to the ones predicted by the classifier.

#### Types of Learning

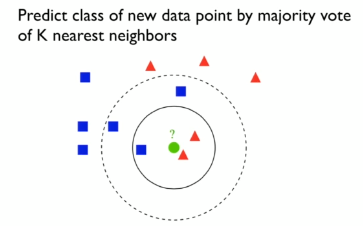
##### Supervised Learning

* KNN, SVM, Decision Trees, Random Forests, Bagging, Booting, et
* Make predictions for new data points
* Data has labels (categories)

##### Unsupervised Learning

* PCA, MDS, Clustering
* Find patterns in the data
* Data has no labels

## Nearest Neighbor Classifier

* Predict class of new data point by majority vote of K nearest neighbors
* 

### 1-Nearest Neighbor Properties

* Simple and quite good for low dimensional data
* ‘Rough’ decision boundary, may have ‘islands’
* Training complexity for N data points is of order(1)
* Test complexity for M data points is O(m \* n)
  + Basically doing all the work in testing.
* There is no error on training set
* Variance is high and bias is low

### k-NN Properties

* Gets rids of islands
* If k is too large, the boundary may become too smooth
* Lower variance, but increased bias
* How do we choose the ideal k?
  + Cross validation

### Distance Measures

#### Manhattan Distance

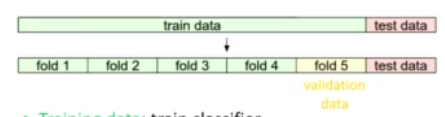
#### Euclidean Distance

#### General Lp norms

## Validation

* Train on training data, test on test data
* Pick the k with the lowest test error
* Training data: train classifier
* Test data: measure performance

### Cross-Validation

* Training data: train classifier
* Validation data: estimate (hyper) parameters (k)
* Test data: measure performance
* 
* Process
  + Iterate over choice of validation fold
  + For all parameter values
    - Train with training data
    - Validate with validation data
  + Average the parameters with best performance on validation data

## The Curse of Dimensionality

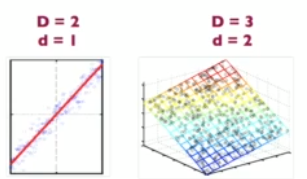
* When dimensionality increases, the volume of the space increases so fast that the available data becomes sparse
* Statistically sound result requires the sample size N to grow exponentially with d

## Dimensionality Reduction

### Basic Idea

* Project the high-dimensional data onto a lower-dimensional subspace that best ‘fits’ the data.
* We want to preserve the distance between the points in the higher dimensions.

### Linear Methods

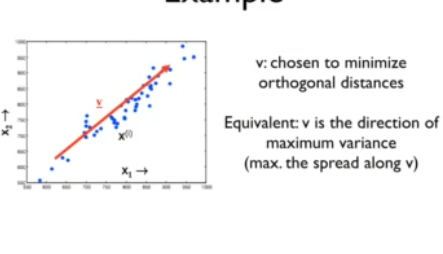
* Does the data lie mostly in a hyperplane?
* If so, what is its intrinsic dimensionality d?
* 

### PCA

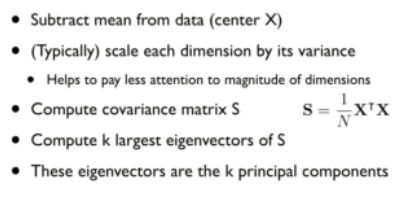
#### Uses

* Dimensionality reduction for supervised learning
* Compression
* Visualization

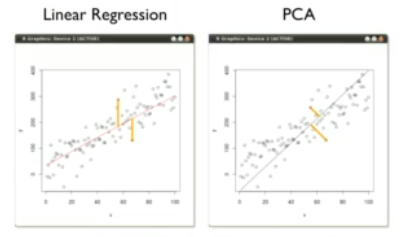
#### Example



#### PCA Algorithm



### Linear Regression vs. PCA



## Multi-Dimensional Scaling (MDS)

* Goal:
  + Find a set of points whose pairwise distances given a distance matrix

### Classical MDS

* Given n x n matrix of pairwise distances between data points
* Compute n x k matrix X with coordinates of distances with some linear algebra magic
* Perform PCA on this matrix X

# Interactive Exercise: Supervised Learning with Scikit-Learn

## Classification

### Supervised learning

#### Naming Conventions

* Features = predictor variables = independent variables
* Target variable = dependent variable = response variable

#### Supervised Learning in Python

* We will use scikit-learn/sklearn
  + Integreates well with the SciPy stack
* Other libraries
  + TensorFlow
  + keras

### The classification challenge

#### k-nearest neighbors: fit

* from sklearn.neighbors import KNeighborsClassifier

### Measuring Model Performance

#### Train/test split

* from sklearn.model\_selection import train\_test\_split
* 
  + We use stratify=y so that the labels to be distributed in the train and test set as they are in the data.
* To score the data we use knn.score(X\_test, y\_test)

#### Model Complexity

* Larger k = smoother decision boundary = less complex model
* Smaller k = more complex model = can lead to overfitting

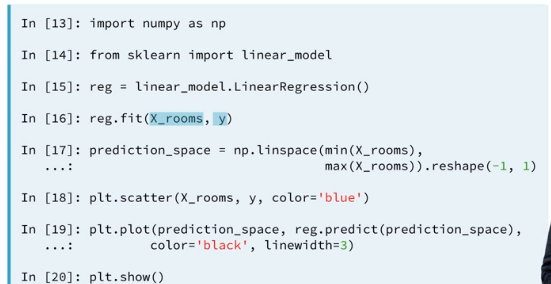
#### Overfitting and underfitting

* Model complexity curve to give some intuition about which k to choose.

## Regression

### Introduction to regression

#### Fitting a Regression Model

* 
* You will have to occasionally reshape features if you are using only one feature
  + X.reshape(-1, 1)
  + Y.reshape(-1, 1)
  + This will change the dimensions from (139,) -> (139, 1)
* You should know when you should include the intercept or not.
  + https://online.stat.psu.edu/~ajw13/stat501/SpecialTopics/Reg\_thru\_origin.pdf

### The basics of linear regression

### Cross-validation

### Regularized regression

* It penalizes model for large coefficients in order to reduce overfitting
* For linear regression, ridge regression should be performed first before lasso

## Fine-tuning Your Model

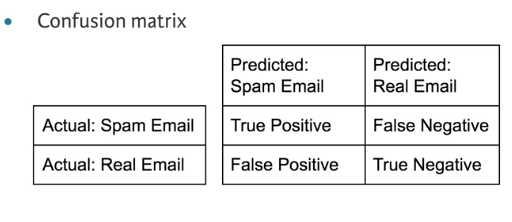
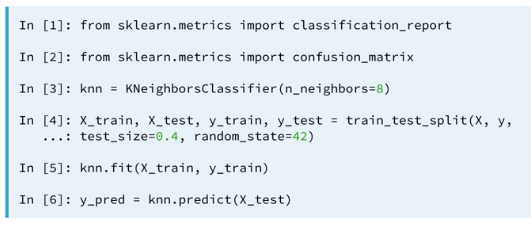
### How good is your model?

#### Classification Metrics

* Measuring model performance with accuracy
  + Fraction of correctly classified samples
* Class imbalance example: Emails
  + Spam classification
    - 99% of emails are real; 1% are spam
  + Build a classifier that predicts all emails are real
    - 99%! Accurate
    - But horrible at actually classifying spam

##### Diagnosing Classification Predictions

###### Confusion matrix

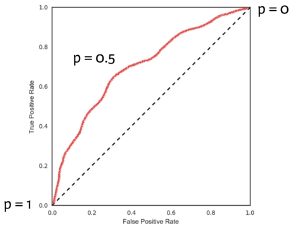
* 
* Accuracy: 
* Precision (Positive predictive value): 
* Recall (Sensitivity): 
* F1 Score: 
* High precision
  + Not many real emails predicted as spam
* High recall: predicted most spam emails correctly
* 

### Logistic regression and the ROC curve

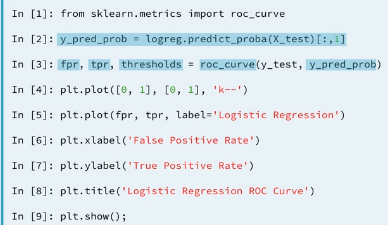
#### Logcistic Regressino for Binary Classification

* Logistic regression outputs probabilities
* If the probability ‘p’ is greater than 0.5:
  + The data is labeled ‘1’
* If the probability ‘p’ is less than 0.5:
  + The data is labeled ‘0’

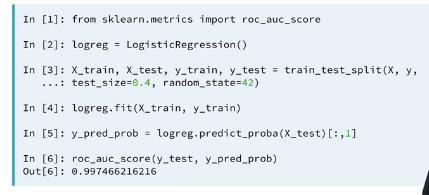
#### The ROC Curve



##### Plotting the ROC Curve



### Area under the ROC curve

* You can calculate AUC from sklearn using roc\_auc\_score
  + 
* You can calculate AUC using cross validation
  + 

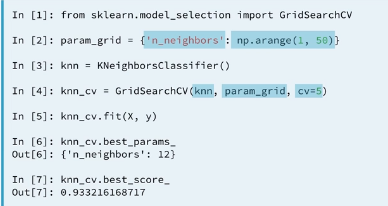
### Hyperparameter tuning

* Linear regression: choosing parameters
* Ridge/lasso regression: choosing alpha
* K-nearest neighbors: choosing n\_neighbors
* Parameters like alpha and k: hyperparameters
* Hyperparameters cannot be learned by fitting the model

#### Choosing the correct hyperparameter

* Try a bunch of different hyperparameter values
* Fit all of them separately
* See who well each performs
* Choose the best performing one
* It is essential to use cross-validation

##### Grid Search Cross-Validation

* Helps us to choose the best hyperparameters based on a slew of possible combinations
* GridSearchCV in scikit-learn
  + 

##### Randomized Grid Search Cross-Validation

* We use RandomizedSearchCV because GridSearchCV can be computationally expensive.
* RandomizedSearchCV will never outperform GridSearchCV.

### Hold-out set for final evaluation

* How well can the model perform on never before seen data?
* Using all data for cross-validation is not ideal
* Split data into training and hold-out set at the beginning
* Perform grid search cross-validation on training set
* Choose best hyperparameters and evaluate on hold-out set

## Preprocessing and Pipelines

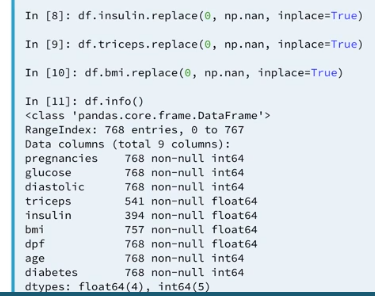
### Preprocessing data

#### Dealing with categorical features

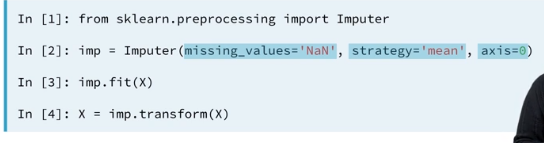
* Scikit-learn will not accept categorical features by default
* Need to encode categorical features numerically
* Convert to ‘dummy’ variables
  + 0: Observation was not that category
  + 1: Observation was that category
* We can use the following functions
  + Scikit-learn: OneHotEncoder()
  + Pandas: get\_dummies()

### Handling missing data

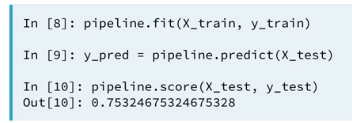
#### Dropping Missing Data

* 
* Use df.dropna()

#### Imputing Missing data

* Making an educated guess about the missing values
* Example: using the mean of the non-missing entries
  + 

#### Imputing within a pipeline

* 
* 

### Centering and scaling

#### Why scale your data?

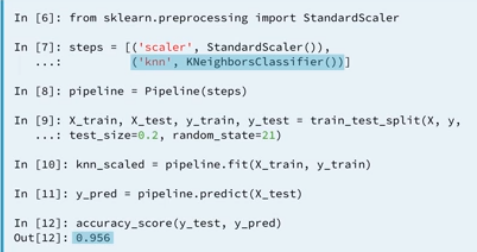
* Many models use some form of distance to inform them
* Features on larger scales can unduly influence the model
  + Example: KNN

#### Ways to normalize your data

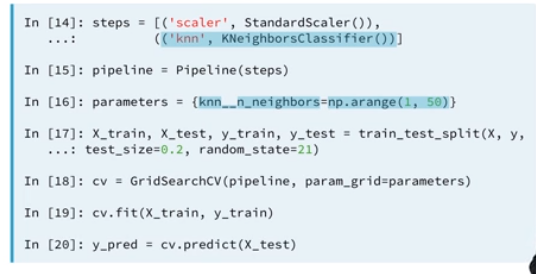
* Standardization: subtract the mean and divide by variance
  + All features are centered around zero and have variance one
* Can also subtract the minimum and divide by the range
  + Minimum zero and maximum one
* Can also normalize so that data ranges from -1 to +1

#### Scaling in scikit-learn

#### Scaling in a pipeline



#### CV and Scaling in a pipeline



# Linear Regression

## Comparing Models

### The F-Statistic

* Compare two nested models

### The AIC score

* Lower values of AIC indicate better fitting models.

## Evaluating the model via model assumptions

* Assumptions
  + Linearity
    - Can be verified with a scatterplot of each x vs y and plotting correlations among x.
    - Can be resolved by transforming one or more independent variables, the dependent variable or both.
    - In other cases, a generalized linear model or a nonlinear model may be better.
  + Constant standard deviation
    - Check this by plotting each x against y and verifying that there is no funnel shape showing data points fanning out as x increases or decreases
    - Use weighted least squares, robust standard errors, or variance stabilizing transformations to deal with this.
  + Normal distribution for errors
    - Verify with a fitted values vs residuals plot and that there is no pattern.
      * Also use a quantile plot
  + Independent errors
  + Correlated predictors
    - Independent variables in a regression model should not be too highly correlated
    - Common solutions are dropping the least important variables involved in the correlation, using regularization, or when many predictors are highly correlated, using something like PCA
  + Influential points
    - High leverage points or outliers
    - Usually just remove them
* Creating the statsmodels package to create these diagnostic plots.
  + Go to this website <http://mpastell.com/2013/04/19/python_regression/>