GENERAL ASSEMBLY

DATA SCIENCE

MACHINE LEARNING / KNN

I. WHAT IS MACHINE LEARNING?
II. SUPERVISED LEARNING
III. CLASSIFICATION WITH K-NEAREST NEIGHBORS
IV. MODEL EVALUATION

I. WHAT IS MACHINE LEARNING?

WHAT IS MACHINE LEARNING?

- **Arthur Lee Samuel** (1901-1990): “Machine learning is the field of study that **gives computers the ability to learn without being explicitly programmed.”**

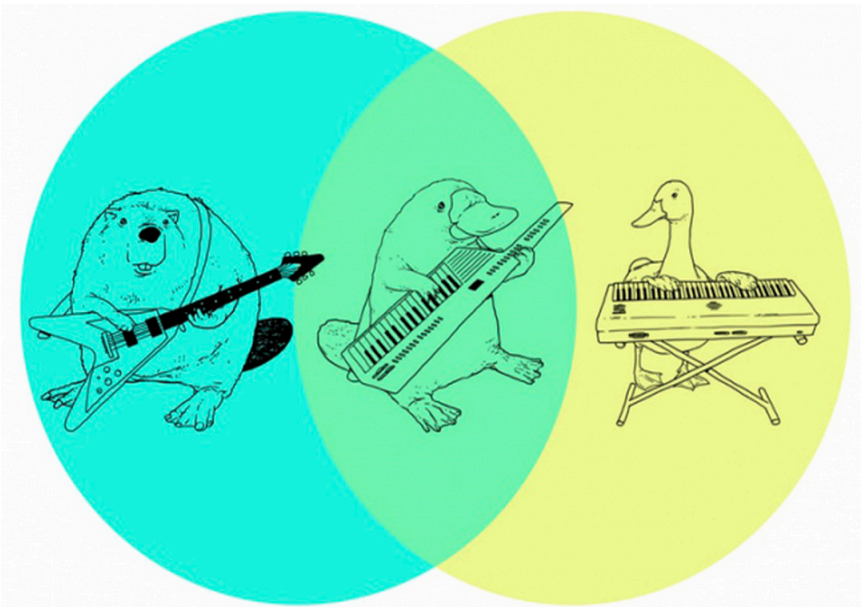
- **Using statistical methods,** machine learning allows a program or function **to extract a structure that best describes data.** (called *representation)*

- This structure is then used to do things like **visualize** and **make predictions** on new or existing data. *(called generalization)*

Arthur Samuel, AI pioneer
Source: Stanford

SO, HOW DOES AN ALGORITHM REPRESENT THE DATA?

- **Simply put, it depends on your algorithm!**
- **Some examples:**
  - Linear regression -> linear equation
  - Decision trees -> list of tree splits
  - Clustering - > distance matrix
- Are all representations created equal? **No!**
  - Prediction: we use quantitative measures to determine predictive accuracy.
  - Description: we qualitatively evaluate usefulness in describing what we want.

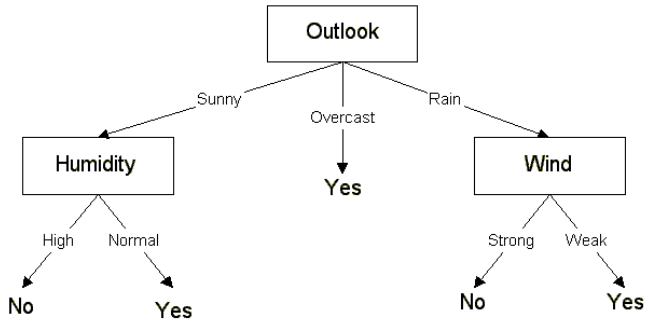


WHAT ARE ITS ADVANTAGES OVER OTHER TECHNIQUES?

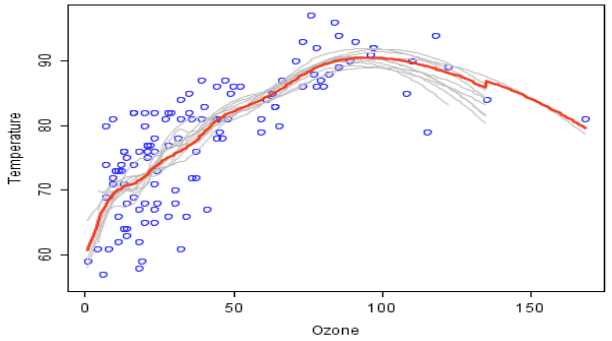
- Allows the computer to be flexible to changes in the data
- Allows instantaneous response and feedback thousands of times faster than what's done by a human
- Allows it to micro-target predictions and data served to millions of people at a time.
- Allows decisions on the data to be made with less bias (but beware – bias is still there!)

WHAT ARE THE MAIN TOOLS USED IN MACHINE LEARNING?

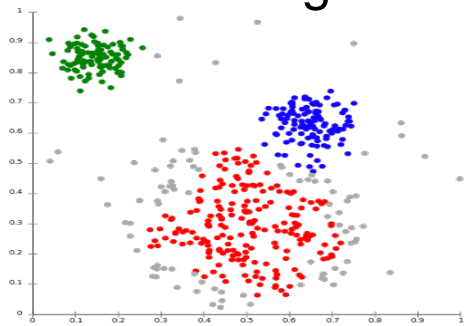
Classification



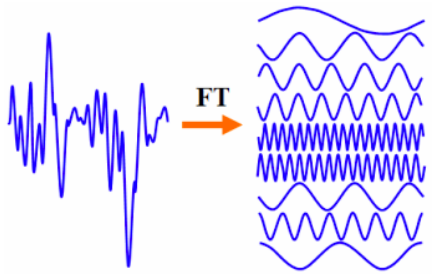
Regression



Clustering

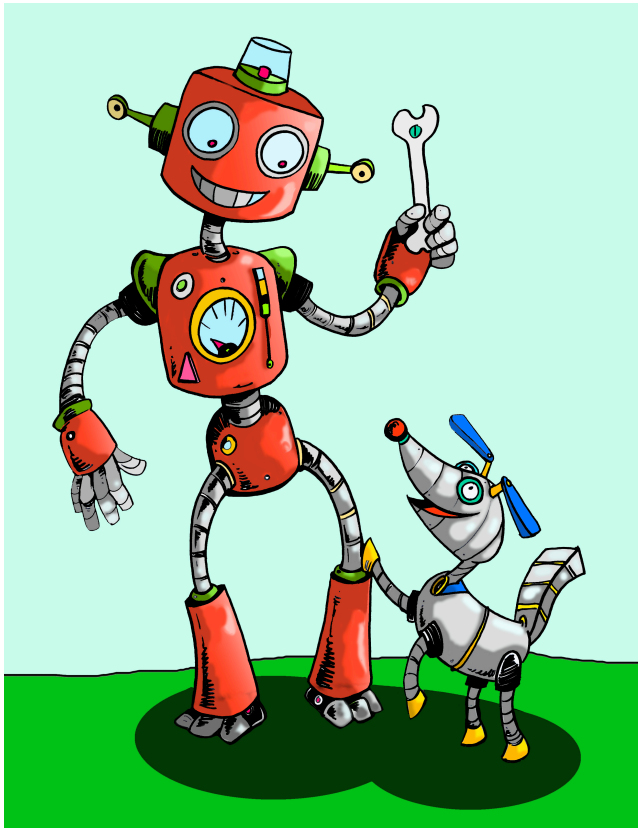


Dimensionality Reduction



WHAT ARE THE MAIN APPLICATIONS OF MACHINE LEARNING?

- Recommendation Systems
- Fraud Detection
- Site Customization
- Time Series Analysis & Automated Forecasting
- Systems and Process Simulation
- Natural Language Processing & Textual Analysis
- Deep Learning and Artificial Intelligence



SO, WHAT ROLE DO I PLAY IN ALL OF THIS?

- Remember, since we're using machine learning models, **you yourself are not setting the rules it will use** to make decisions.

- **Think of yourself more like a general or referee–** you does not tell the computer to make decisions, but provides the **structure and constraints** that guides the computer how to make its decisions.

supervised *(regression, classification)*	making predictions *(generalization)*
unsupervised *(clustering)*	extracting structure *(representation)*

II. SUPERVISED LEARNING

- **In supervised learning, you have a concrete prediction goal in mind.**
  - Example: Is this e-mail spam or not spam? What temperature will it be tomorrow?

- **This 'goal' variable has many names, but in essence, it is your 'y' variable.**
  - 'response variable' , 'dependent variable', 'left-hand-side variable', 'outcome', 'label', and 'target' are other names for it.
  - If your 'y' variable is continuous: **use regression.**
  - If your 'y' variable is categorical: **use classification.**

- **The data that you use to predict your 'goal' variable are your 'features'.**
  - 'explanatory variables', 'independent variables', 'right-hand-side variables', 'predictors', 'inputs', 'regressors', 'covariates', and 'attributes' are other names for it.

- Your data is composed of 'observations', 'samples', 'examples', 'instances', or 'records'.

III. SUPERVISED LEARNING EXAMPLE: CLASSIFICATION WITH K-NEAREST NEIGHBORS

Q: How does a classification algorithm work?
A: Data in, predicted labels out.

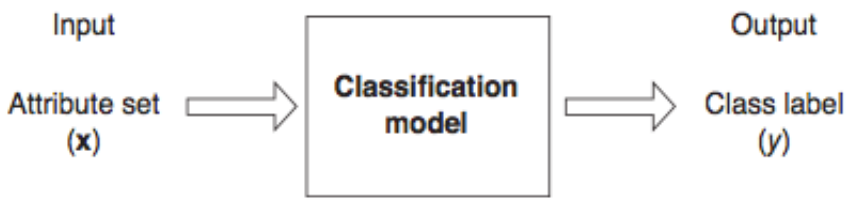


Figure 4.2. Classification as the task of mapping an input attribute set $\mathbf{x}$ into its class label y.

Source: http://www-users.cs.umn.edu/~kumar/dmbook/ch4.pdf

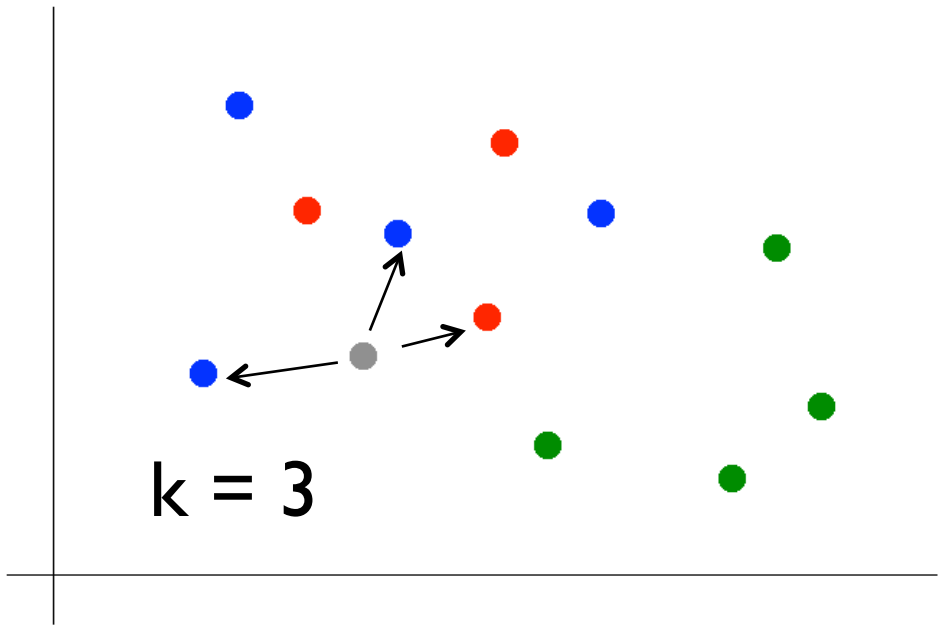
- **The first machine learning algorithm we are going to learn in this class is K-Nearest Neighbors (KNN)**
- Why?
  - The algorithm is simple to understand and explain.
  - Model training is fast.
  - The algorithm is non-parametric – so it does not presume a “form” of the “decision boundary”.
  - In cases with little or indeterminate signal, performs just as well as more complex algorithms.
- Why don’t I use it as my go-to algorithm?
  - It is sensitive to irrelevant features.
  - Vanilla implementations do not consider feature importance when making predictions.
  - Training for the optimal K can be slow.
  - The prediction phase can be slow when n is large.

- **The first (and trickiest) thing to do is to create a distance matrix using the features (explanatory variables) of each training observation in your dataset.**
  - To create the matrix, you must define a distance function.
  - Euclidian distance, the triangular distance between two points in a two-dimensional space, is the most popular.
  - If you have two spatial features that are continuous and of the same magnitude, calculating Euclidian distance is easy: just the find triangular distance per your high school geometry textbook.
  - If you have many features not of the same magnitude, or categorical ones, this is hard to do well!

	A	B	C	D	E	F
A	**0.00**	0.71	5.66	3.61	4.24	3.20
B	0.71	**0.00**	4.95	2.92	3.54	2.50
C	5.66	4.95	**0.00**	2.24	1.41	2.50
D	3.61	2.92	2.24	**0.00**	1.00	0.50
E	4.24	3.54	1.41	1.00	**0.00**	1.12
F	3.20	2.50	2.50	0.50	1.12	**0.00**

CLASSIFICATION WITH KNN

- Once that step is over, you can use your distance matrix to compute a two-dimensional representation of the data like to the left.
- Suppose now that we want to predict the color of the gray dot.
- To implement our algorithm, we:
  1. Pick a value for k (3 in this example).
  2. Find the colors of the 3 nearest neighbors.
  3. Assign the frequently occurring color to the gray dot.



BUT WAIT, THERE'S MORE!

- Before we can do our k-nearest neighbors example, I need to expose you to an additional core concept in data science: **cross-validation.**

- There's much, much on this in the next section of this class, but for the sake of the in-class exercise, you'll just need to know that to test the model's accuracy, we only expose the model to a certain percentage of the data.
  - This is called our **training** set.

- Then, we test the mode's accuracy on a hold-out group of the that the model has not seen.
  - This is called our **test** or **holdout** set.

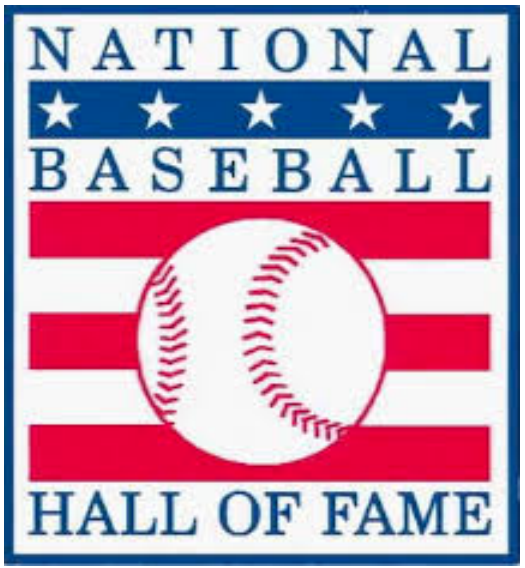
IN-CLASS EXERCISE: KNN

- Let's build a K-nearest neighbors model to predict whether a baseball player made it into the Baseball Hall of Fame.
  1. In Pandas, query the following information and coerce it into a DataFrame using the from_sql() method:
     - In the HallofFame table, the player ID, number of ballots, number of votes, and whether the player was inducted for all Hall of Fame votes before the year 2000.
  2. Drop all rows in the DataFrame with NaNs in them.
  3. Create a derived feature in Pandas called 'percent_of_ballots' that shows the percent of ballots the person's baseball hall of fame candidacy received.
  4. Split the data into training and test groups.
  5. Run a KNN classifier on the training group.
  6. Test the classifier's accuracy on the test group.

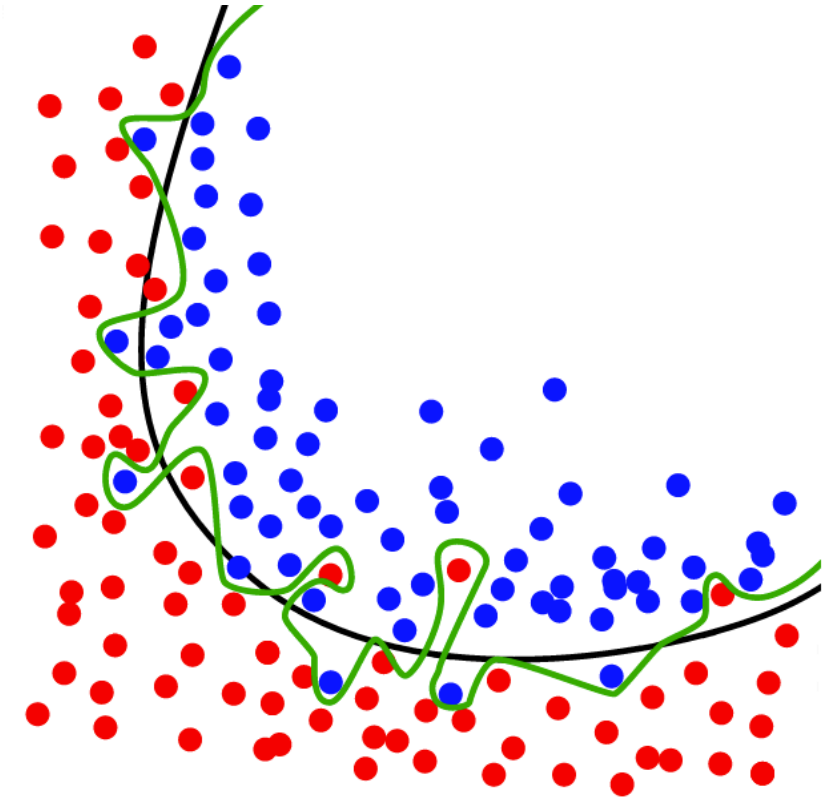
IV. MODEL EVALUATION

TRAINING ERROR

- Suppose that in the last exercise, we trained our model using the entire dataset.
- How low can we push the training error?
  - We could have made the model arbitrarily complex (effectively “memorizing” the entire training set), pushing error **all the way down to zero!**
- Does this mean that in the real world, we can predict all future Baseball Hall of Fame votes perfectly?
  - Hell no!
  - This is called **overfitting** the dataset.

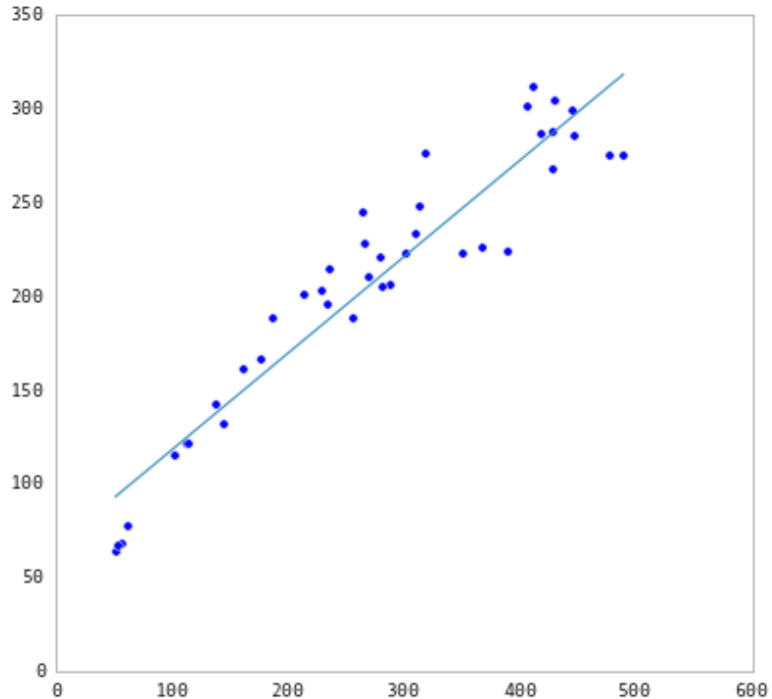


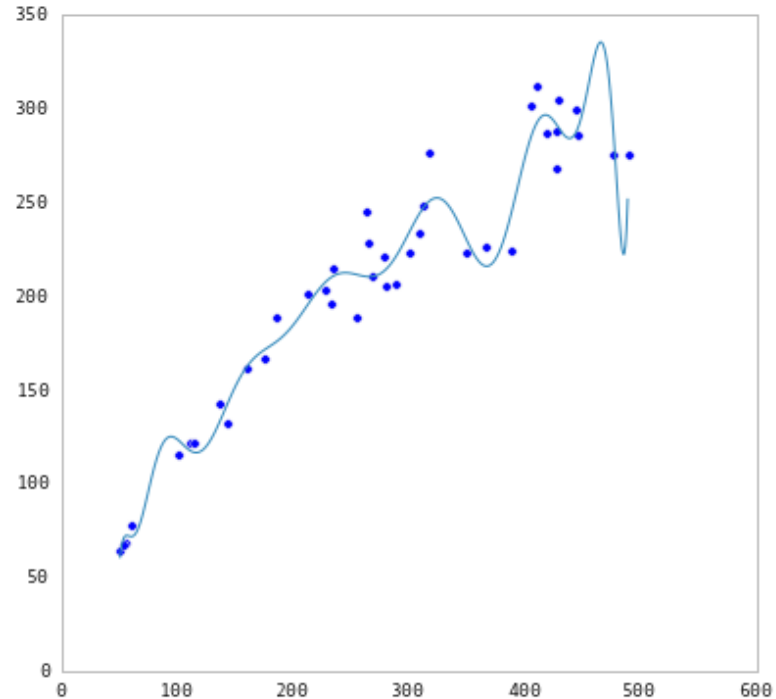
OVERFITTING



Source: http://www.dtreg.com

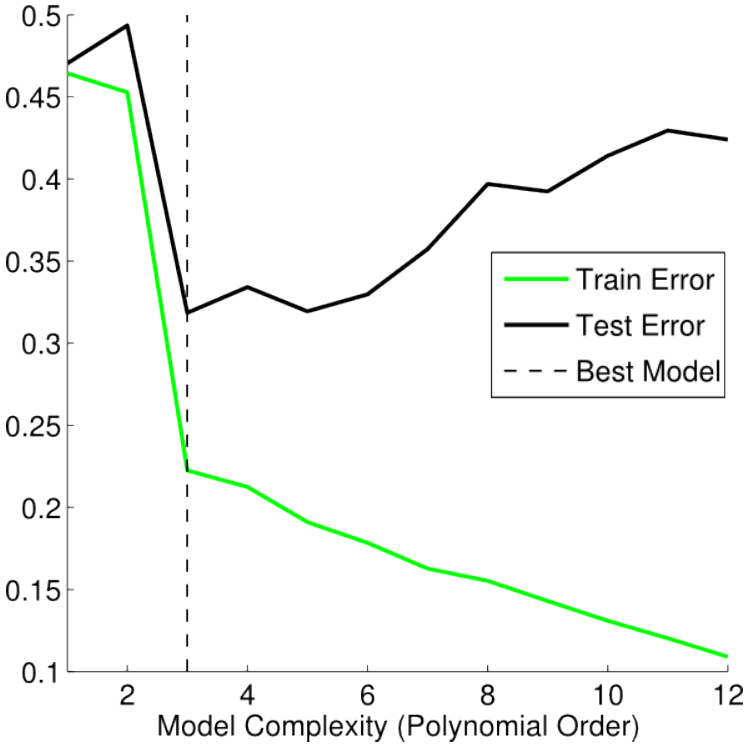
UNDERFITTING AND OVERFITTING

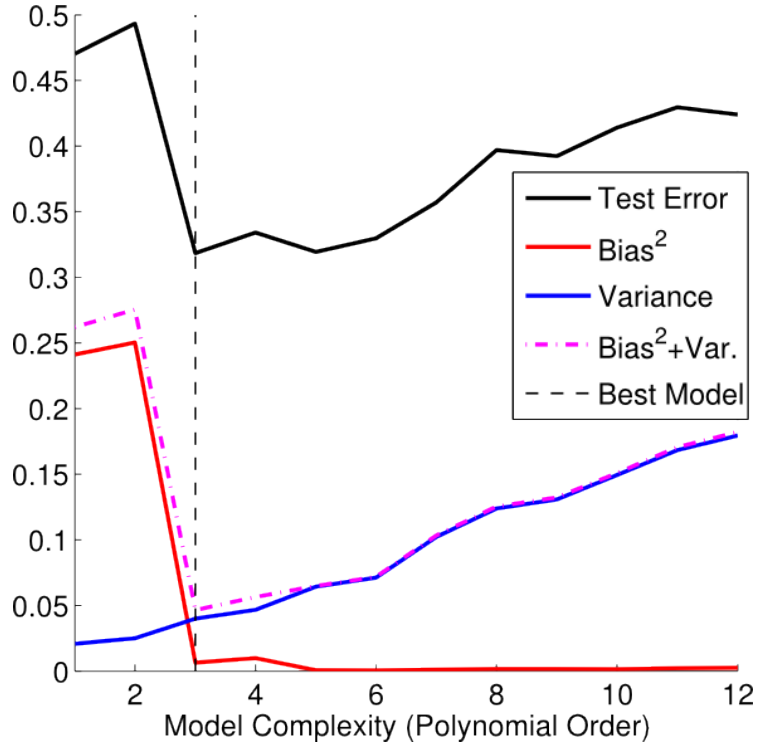




Source: http://nbviewer.ipython.org/github/fonnesbeck/Bios366/blob/master/notebooks/Section6_3-Model-Selection-and-Validation.ipynb

UNDERFITTING AND OVERFITTING



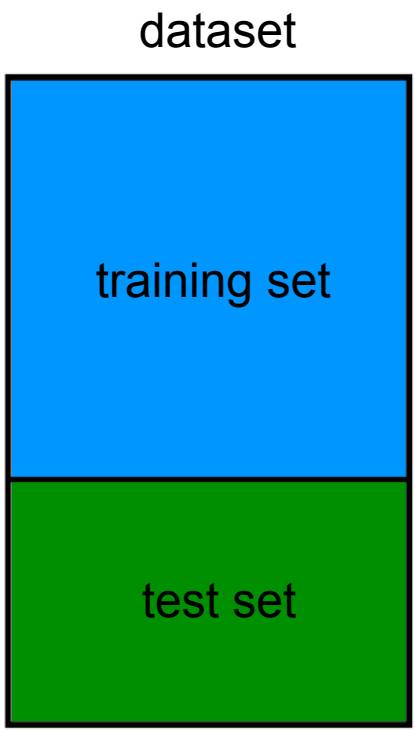


Source: http://theclevermachine.wordpress.com/2013/04/21/model-selection-underfitting-overfitting-and-the-bias-variance-tradeoff/

SO, HOW CAN WE MAKE A MODEL THAT GENERALIZES WELL?

- By splitting the data! Most data scientists use this pipeline:

1. **Fit your model on the training dataset.**
2. **Test your model on a holdout group.**
3. **Tune your model's parameters by comparing accuracy on resampled holdout groups.**
4. **Chose the model (and tuning parameters) that best predict across *all* resampled holdout groups.**
5. **Train your final model across *all* the data.**
6. **Estimate out-of-sample prediction error.**
7. **Make predictions on your on new (out-of-sample) data.**



WHY DON'T WE JUST USE ONE HOLDOUT GROUP (LIKE WE DID BEFORE?)

- If you only do one train-test split, you'll get a **high-variance estimate** of out-of-sample accuracy.
  - **Why?** Because accuracy depends on the particularities of your hold-out test!

- We use **cross-validation (CV)** to get around this issue.

- The most popular type of CV is called **K-fold cross validation.** In K-fold CV, you:
  1. Randomly split the dataset into K equal partitions.
  2. Use partition 1 as a test set and combine all other partitions into a training set.
  3. Calculate test set error.
  4. Repeat steps 2-3 using a different partition as the test set at each iteration.
  5. Take the average test set error as the estimate of OOS accuracy.

CROSS-VALIDATION IN A NUTSHELL

FEATURES OF K-FOLD CROSS-VALIDATION

- K- fold CV gives a **more accurate estimate** of out-of-sample (OOS) prediction error.
  - As you use each record in our dataset for both training and testing, the model is exposed to the **full range of data**, allowing the data scientist to understand better how it reacts in different situations pertaining to the data.
- K-fold CV can be used for parameter tuning and model selection.
- However, K-fold CV **presents a tradeoff** between accuracy, efficiency, and computational expense.
  - 10-fold CV is ten times more expensive than a single train/test split.
  - Especially when the dataset is large, you may not be able to do this in a reasonable period of time (say, overnight!).

IN-CLASS EXERCISE: K-FOLD CV

- Continue with your work on the Hall of Fame dataset.

1. First, verify that a different test/train split will give you different accuracy results each time you try it.
2. Next, use 10-fold cross validation to score our model.
3. Next, use a ‘for’ loop to use cross validation to tune the model for the optimal number of K .
4. Use scikit-learn’s grid search to automatically find the optimal number of K.
5. Compare accuracy of grid search optimized model with data from 2000 onwards against its estimated accuracy on out-of-bag data.

QUESTIONS?