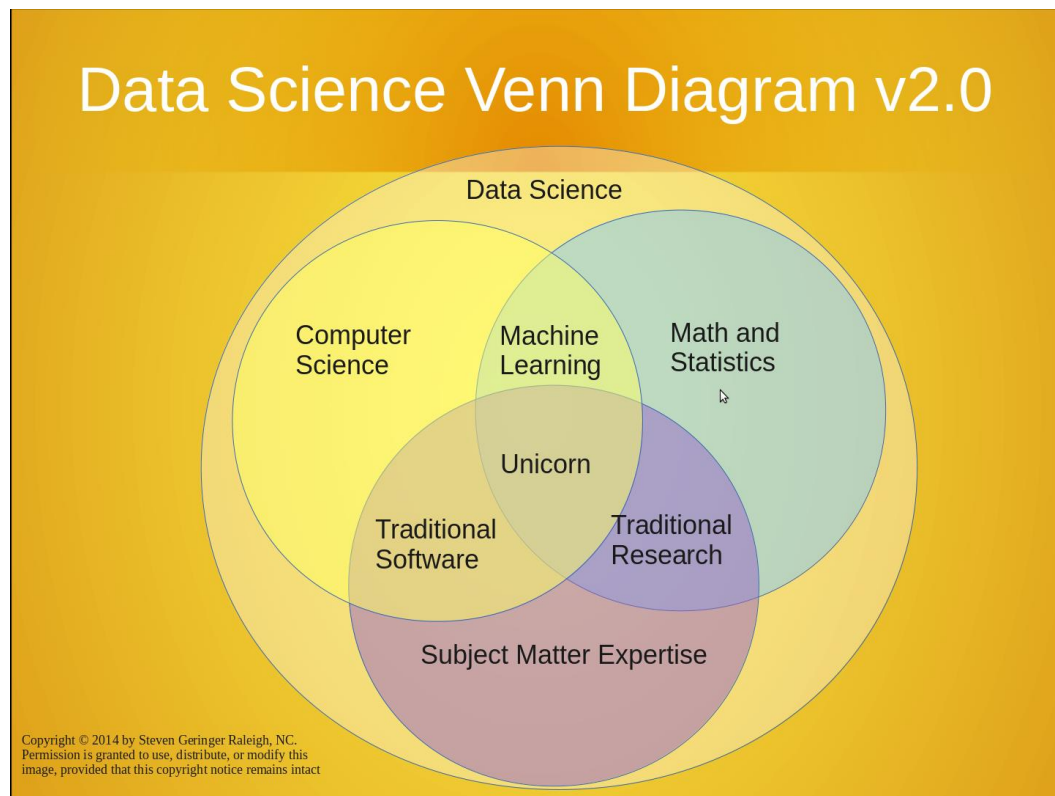
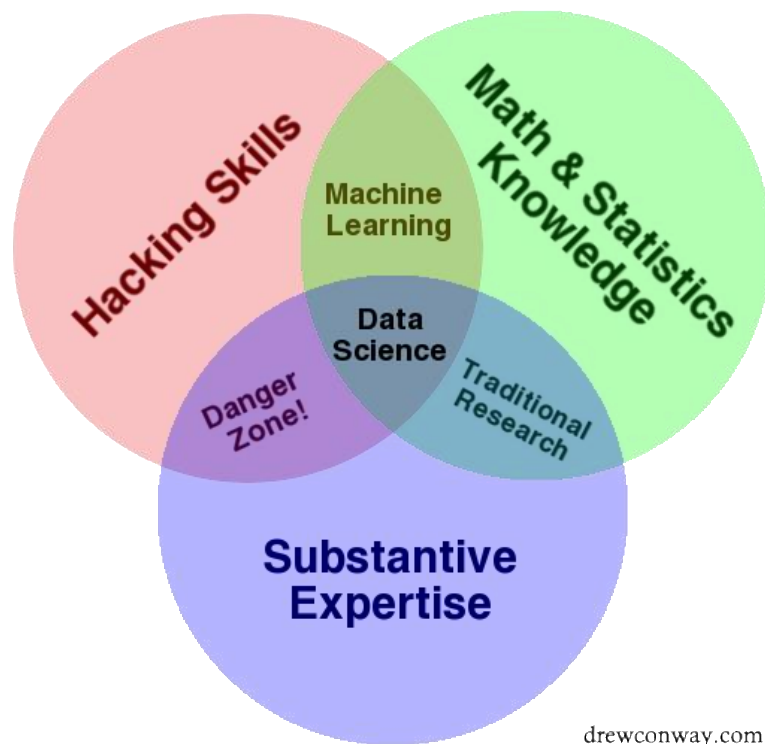


Rice Data Science Bootcamp

Instructor: Natalie Berestovsky, PhD
Occidental Petroleum/
Anadarko Petroleum

- Education
 - BS in Computer Science from UT-Austin
 - MS and PhD in Computer Science from Rice
- Professional Experience:
 - Worked in Chevron for 6 years in Chevron
 - Data Scientist in Anadarko for 2.5 years
 - Next: Oxy?
- AI experience:
 - ML techniques, deep learning
 - Google Cloud Platform





Unsupervised vs. Supervised Learning

Data Matrix:

$$\mathbf{X}_{n \times p} = \begin{pmatrix} x_{11} & x_{12} & \dots & x_{1p} \\ \vdots & & \ddots & \\ x_{n1} & x_{n2} & \dots & x_{np} \end{pmatrix}$$

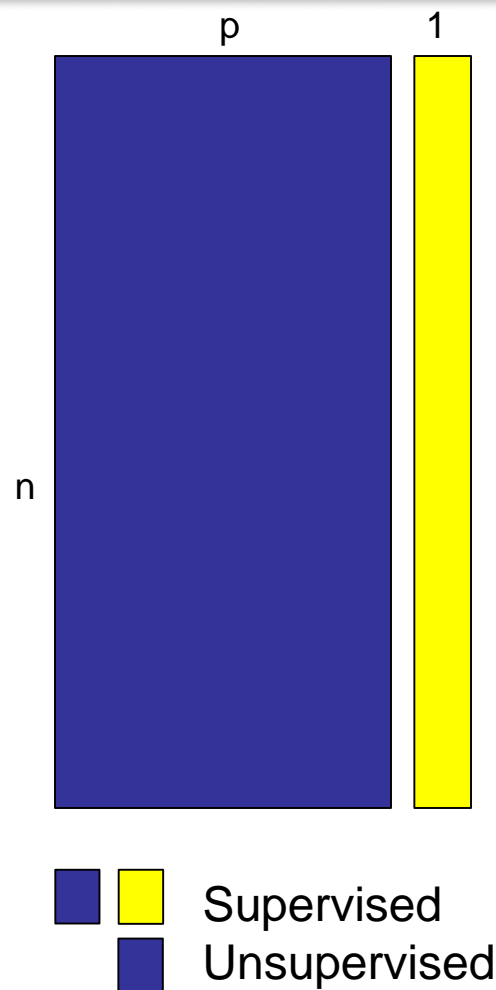
- Rows: n observations / samples / subjects.
- Columns: p features / variables.

Supervised Learning:

$$\mathbf{Y} = (y_1, y_2, \dots, y_n)^T$$

- Classification: \mathbf{Y} - n labels.
- Regression: \mathbf{Y} - n continuous outcomes.

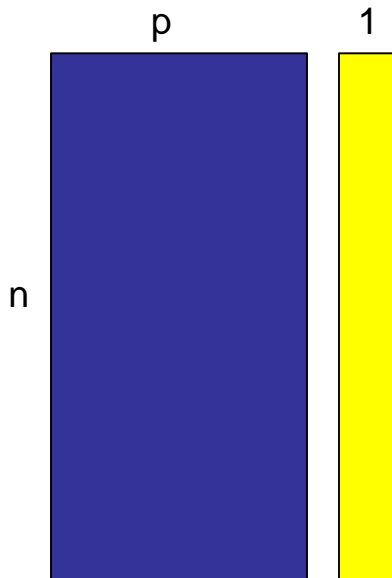
Unsupervised Learning: No outcomes / labels!



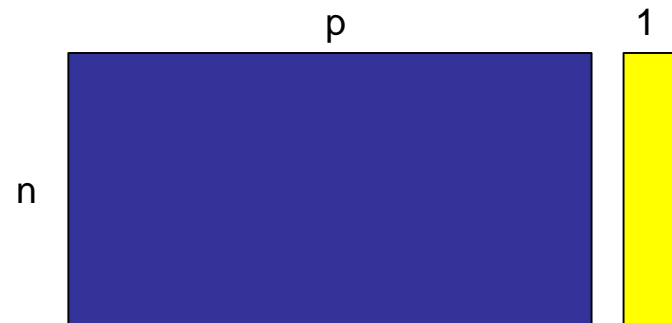


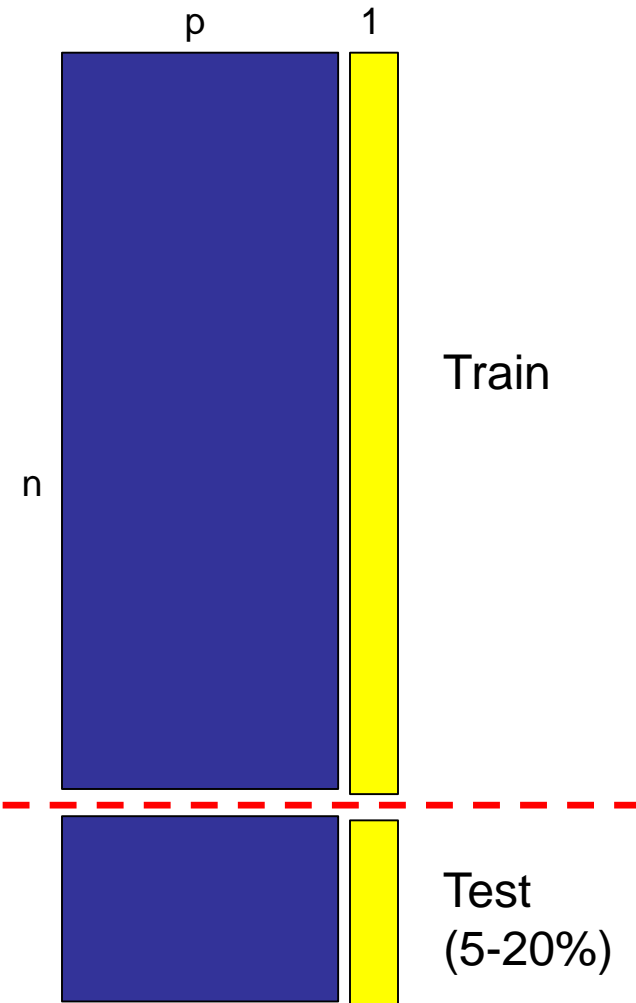
- *Supervised / Unsupervised / Both*
 - Recognizing digits in the images (S)
 - Gene clustering (U)
 - Identifying alerts in your streaming data (B)
 - Predicting stock prices (B)
 - Anomaly / fraud detection (B)
 - Ranking public speech impact from recordings (U)
 - Face recognition (S)
 - Automated text completion (Gmail) (S)
 - Voice recognition (S)

- $n \gg p$
- More samples than features
- Examples?



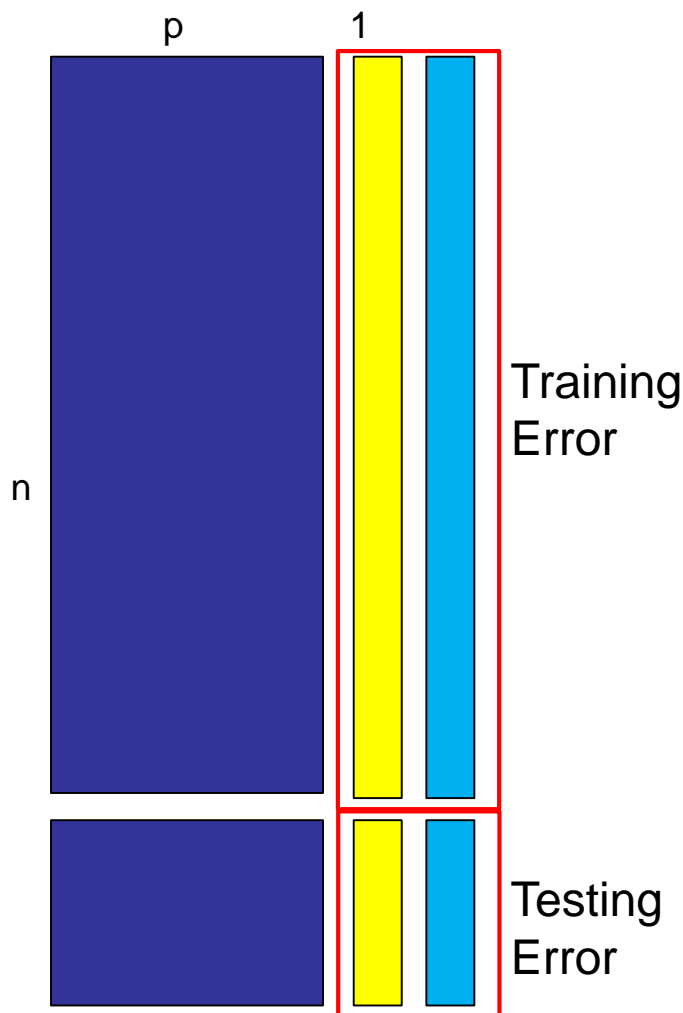
- $n \ll p$
- Fewer samples than features
- Examples?





- Train / test split
 - Hold out validation
 - Selects $x\%$ of data randomly
 - K-fold cross validation
 - Shuffles that data (or not)
 - Split the data into k equal sets
 - For each split, use $k-1$ set for training and test on the k^{th}

Assessing your model



- Training error
 - Error of how well the model generates the data that it has seen
- Testing error
 - Error on unseen data (prediction)
- Generally $E_{\text{train}} \leq E_{\text{test}}$
- What if $E_{\text{train}} \ll E_{\text{test}}$?
- What if $E_{\text{train}} > E_{\text{test}}$?

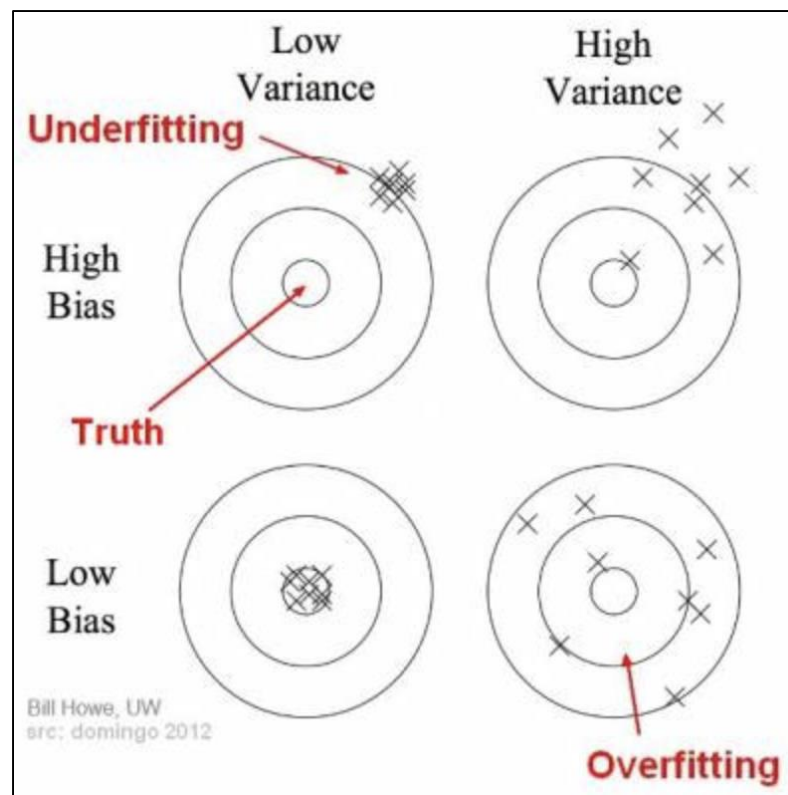


$$Err(x) = E \left[(Y - \hat{f}(x))^2 \right]$$

$$Err(x) = \left(E[\hat{f}(x)] - f(x) \right)^2 + E \left[\left(\hat{f}(x) - E[\hat{f}(x)] \right)^2 \right] + \sigma_e^2$$

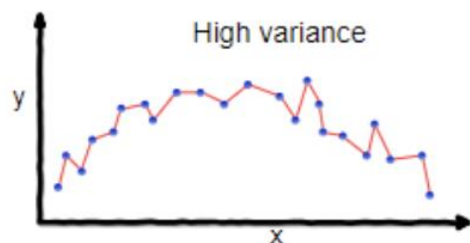
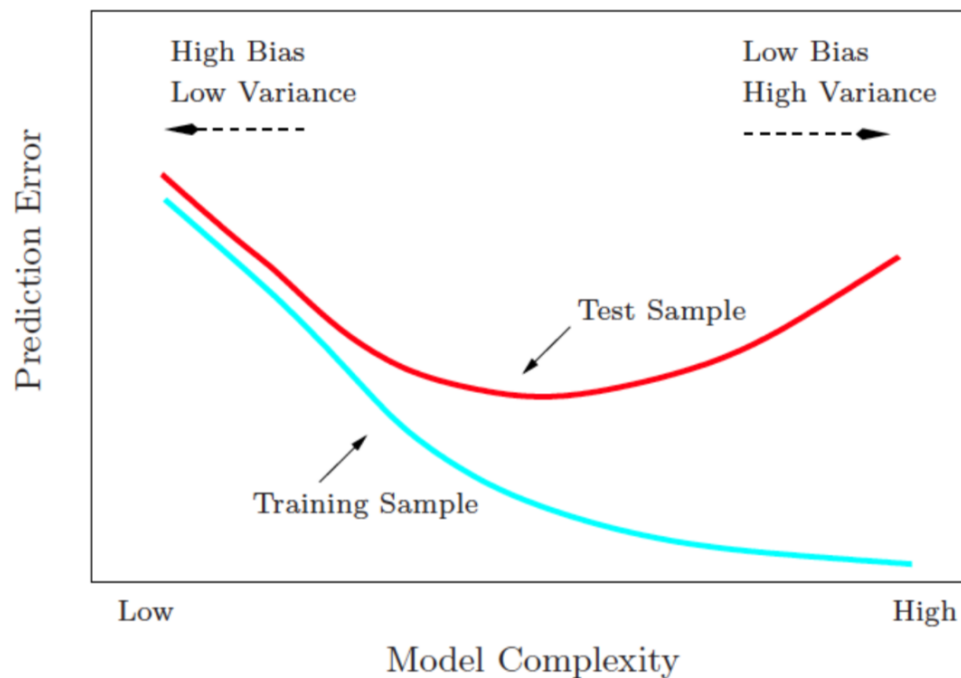
$$Err(x) = \text{Bias}^2 + \text{Variance} + \text{Irreducible Error}$$

- **Bias** – difference between predicted and true value
 - *High bias* – generalized model, high error
 - *Low bias* – fits well to the training set, low training error
- **Variance** – variability of prediction for a given data point
 - *High variance* – sensitive to training data set
 - *Low variance* – not sensitive to changes in training set
- **Irreducible error** – noise in the data

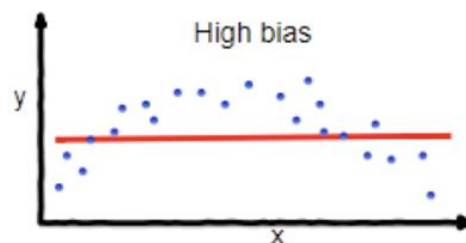




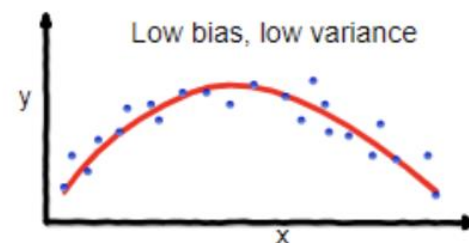
- Bias-variance tradeoff
 - Complex, many parameters
 - overfitting
 - Simple, few parameters
 - underfitting



overfitting



underfitting



Good balance



	Supervised	Unsupervised
Discrete	Classification	Clustering
	KNN Datasets: <i>Wine dataset</i>	K-Means Hierarchical clustering (Biclustering) Datasets: Synthetic data NCI60
Continuous	Regression	Dimensionality reduction
	Multilinear regression Ridge regression Lasso regression Datasets: <i>Boston housing dataset</i> <i>Synthetic data</i>	Principle Component Analysis (PCA) Datasets: Art data NCI60



`sklearn.neighbors.KNeighborsClassifier`

```
class sklearn.neighbors. KNeighborsClassifier (n_neighbors=5, weights='uniform', algorithm='auto',  
leaf_size=30, p=2, metric='minkowski', metric_params=None, n_jobs=None, **kwargs)
```

[\[source\]](#)

Classifier implementing the k-nearest neighbors vote.

Read more in the [User Guide](#).

Parameters: **n_neighbors** : *int, optional (default = 5)*

Number of neighbors to use by default for `kneighbors` queries.

weights : *str or callable, optional (default = 'uniform')*

weight function used in prediction. Possible values:

- 'uniform' : uniform weights. All points in each neighborhood are weighted equally.
- 'distance' : weight points by the inverse of their distance. in this case, closer neighbors of a query point will have a greater influence than neighbors which are further away.
- [callable] : a user-defined function which accepts an array of distances, and returns an array of the same shape containing the weights.

algorithm : *{'auto', 'ball_tree', 'kd_tree', 'brute'}, optional*

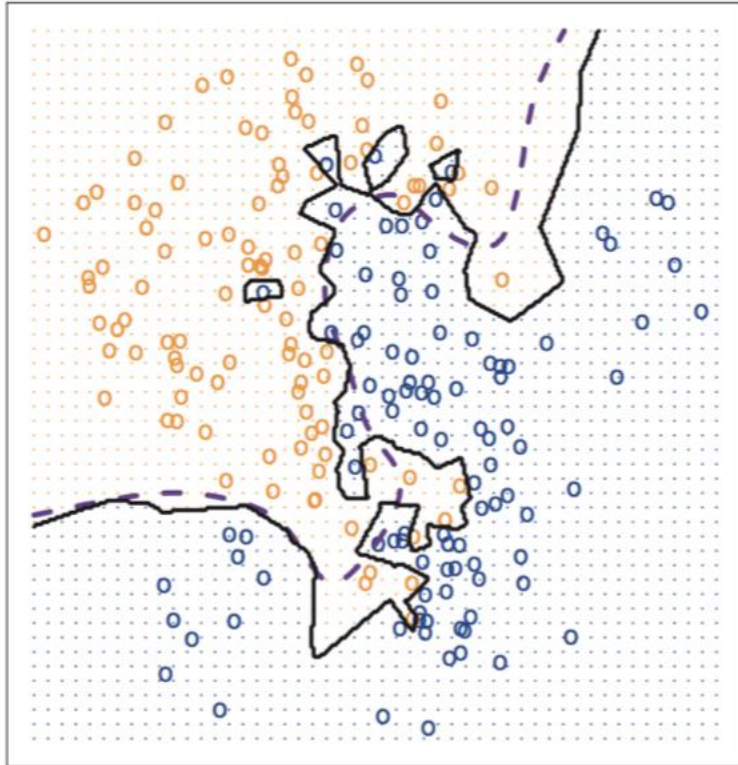
Algorithm used to compute the nearest neighbors:

- 'ball_tree' will use `BallTree`
- 'kd_tree' will use `KDTree`

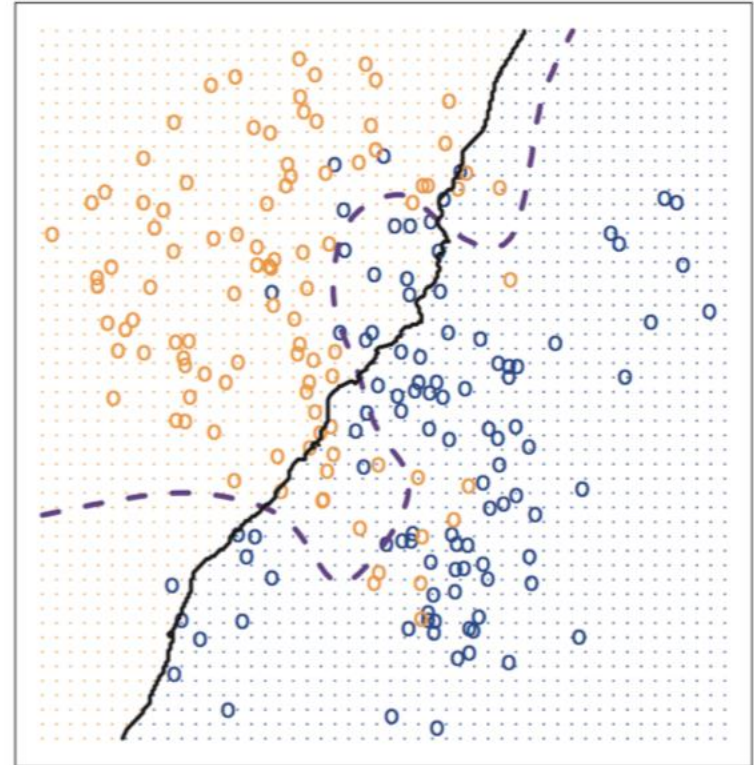


K Nearest Neighbors

KNN: K=1

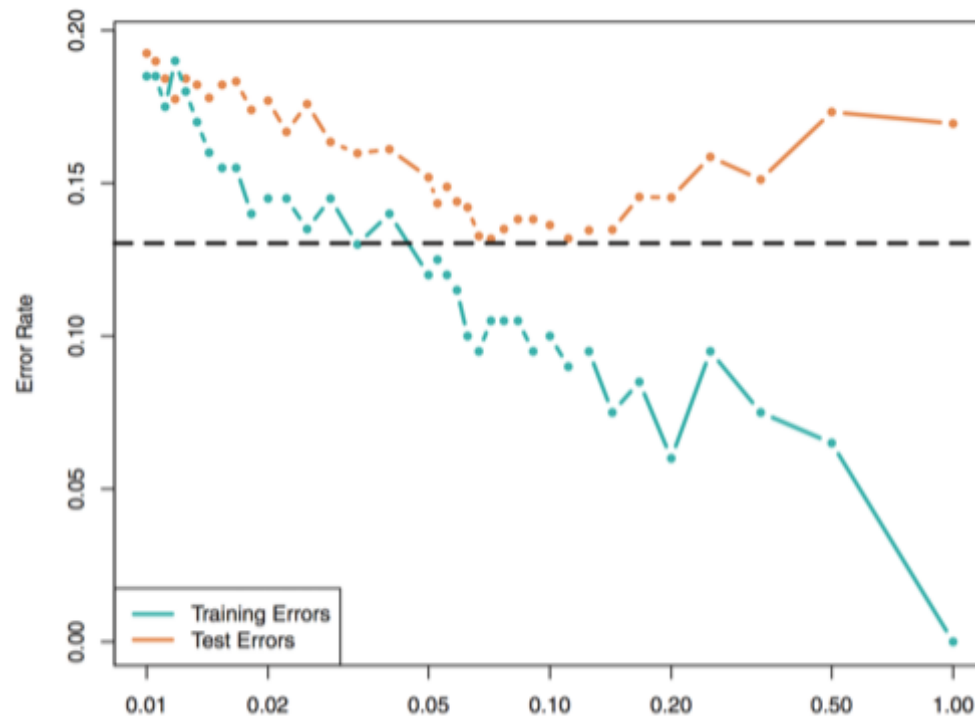
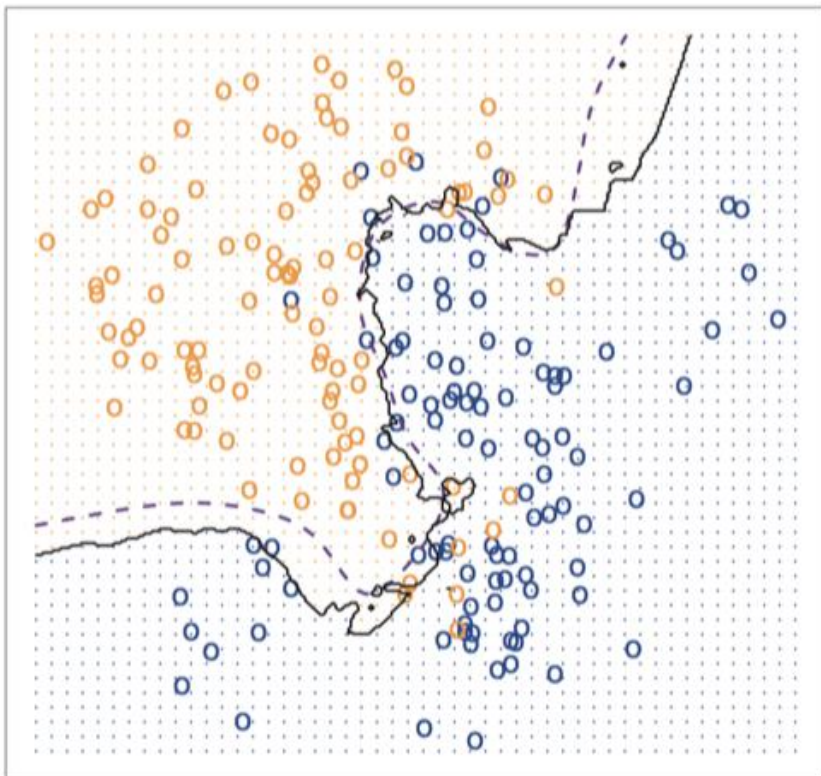


KNN: K=100





K Nearest Neighbors

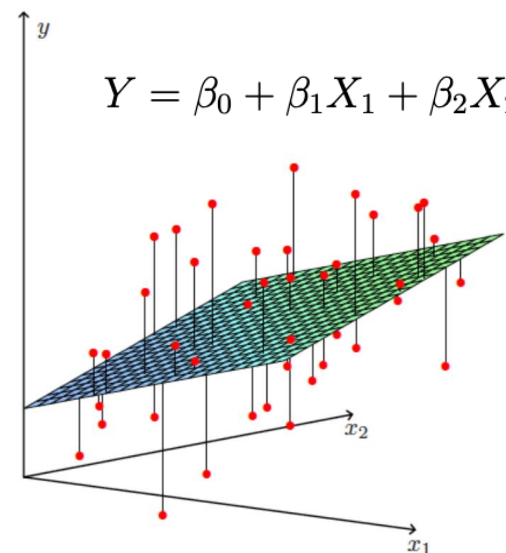
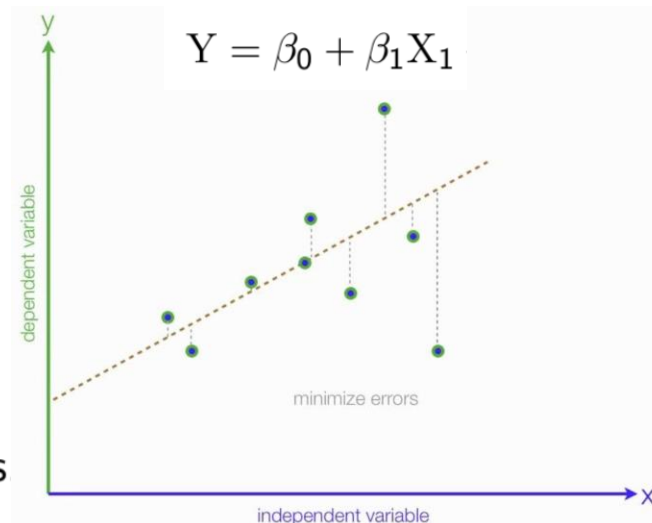


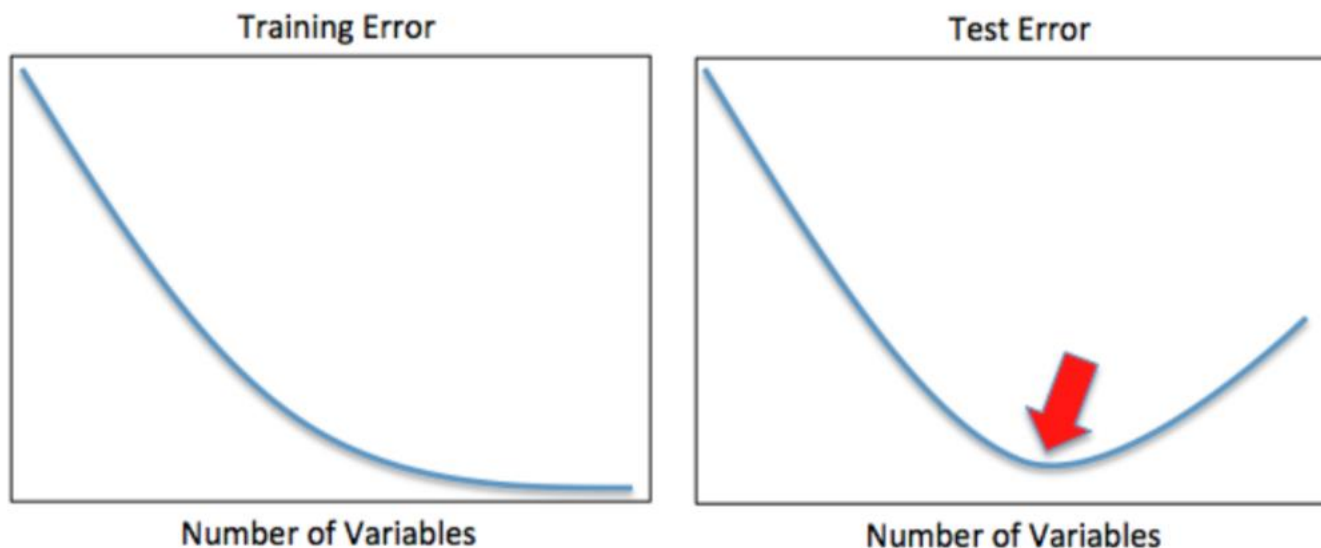


- Linear Model:

$$Y = \beta_0 + \beta_1 X_1 + \dots + \beta_p X_p + \epsilon.$$

- Y - responses associated with n observations.
- X_j - j^{th} predictor measured for each of the n observations
- ϵ measurement noise for the n observations.
- β_0 - Intercept.
- β_j - Coefficient of j^{th} predictor.
- Goal: Estimate β_0 and β_j 's to fit linear model.





- Solution:
 - Feature selection
 - Best subset regression, Forward and backward step-wise
 - Regularized linear regression
 - Ridge regression, Lasso regression



- **Ridge**

$$\underset{\beta}{\text{minimize}} \quad ||Y - \mathbf{X}\beta||_2^2 + \lambda ||\beta||_2^2$$

- It shrinks the parameters, therefore it is mostly used to prevent multicollinearity.
- L2 makes weights small

- **Lasso**

$$\underset{\beta}{\text{minimize}} \quad ||Y - \mathbf{X}\beta||_2^2 + \lambda ||\beta||_1$$

- Used when we have more number of features, because it automatically does feature selection.
- L1 removes weights

Regression = [Prediction]

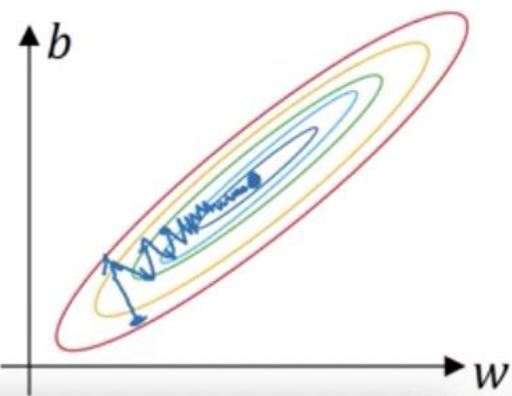
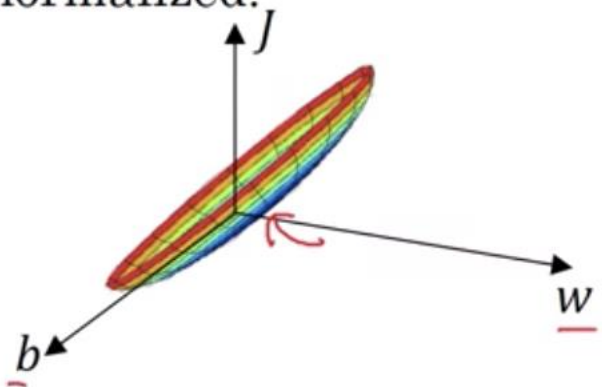
Regression + Ridge = [Prediction] + [Bias Variance Trade-off]

Regression + Lasso = [Prediction] + [Bias Variance Trade-off] + [Feature Selection]

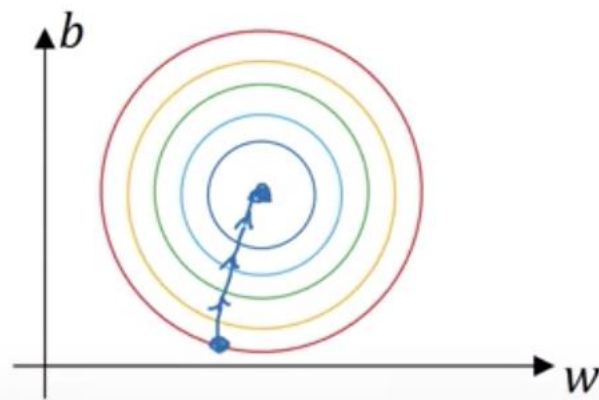
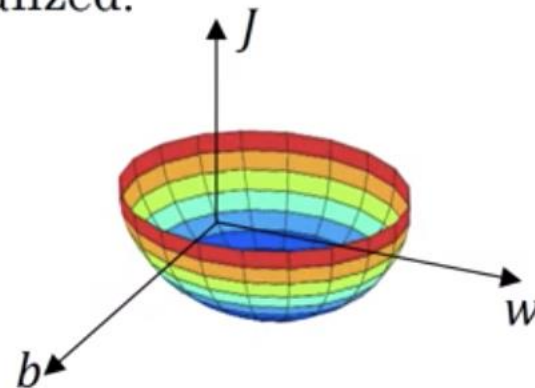


Understand contours

Unnormalized:

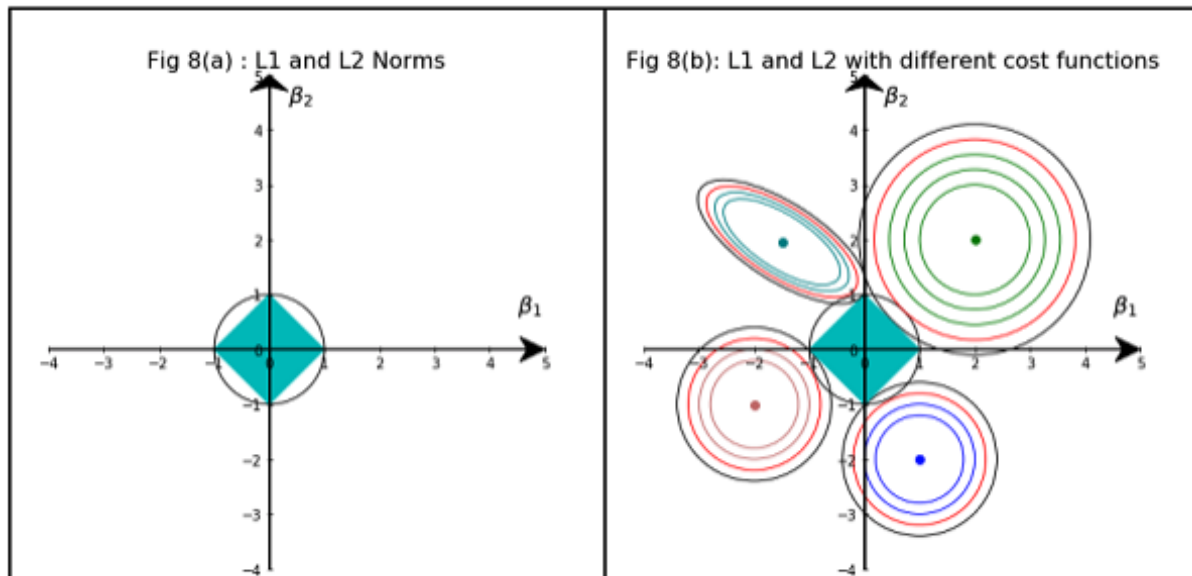


Normalized:





Ridge and Lasso regressions



- **Force 1:** Bias term pulling β_1 and β_2 to lie somewhere on the black circle only.
- **Force 2:** Gradient Descent trying to travel to the global minimum indicated by the dots.
- Each gradient descent contour corresponds to different LR
- The RED circle in each contour intersects the Ridge or L2 Norm.
- The BLACK circle in each contour intersects the Lasso or L1 Norm.

