TEXAS ADVANCED COMPUTING CENTER
WWW.TACC.UTEXAS.EDU

TEXAS
The University of Texas at Austin

Machine Learning on HPC: Supervised Learning

Scalable Computational Intelligence Group
Texas Advanced Computing Center
University of Texas at Austin

Juliana Duncan
June. 2023

Workshop Series Overview

- There are five workshops this week:
  - Workshop 1: July 15, 2024
    - Introduction on computing cluster environment, Python for machine learning and containers
  - **Workshop 2: July 16, 2024**
    - **Common supervised machine learning methods with examples and hands-on exercises**
  - Workshop 3: July 17, 2024
    - Common unsupervised machine learning methods with examples and hands-on exercises
  - Workshop 4: July 18, 2024
    - Deep learning concepts, common neural network structure and frameworks with examples and hands-on exercises.
  - Workshop 5: July 19, 2024
    - Advanced ML topics
      - Introduction to Reinforcement Learning and LLMs

Agenda of Today

- 9:00 - 11:00
  - Machine Learning Basics
  - Common Supervised Learning Methods
- 11:00-12:00
  - Introduction to Hands-on Exercises
- 1:00 - 2:30
  - More Common Supervised Learning Methods
- 2:30 - 4:00
  - Hands-on Exercises

Outline

- Machine Learning (ML) Basics
  - Define ML
  - Categories of ML
  - ML Workflow
- Common Supervised Learning Techniques
  - K Nearest Neighbors
  - Linear Regression
  - Regularization
  - Logistic Regression
  - Support vector machines
  - Probabilistic Models
  - Decision Tree Based Models

What is Machine Learning?

- A formal definition from T. Mitchell (1997). *Machine Learning book*:

> "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at the tasks improves with the experiences."
> --- (Mitchell 1997)

Recognize hand written words within images

- Task: Recognize hand written words within images
- Performance: percent of words classified correctly
- Experience: database of classified hand written words

What is Machine Learning?

- A formal definition from T. Mitchell (1997). *Machine Learning book*:

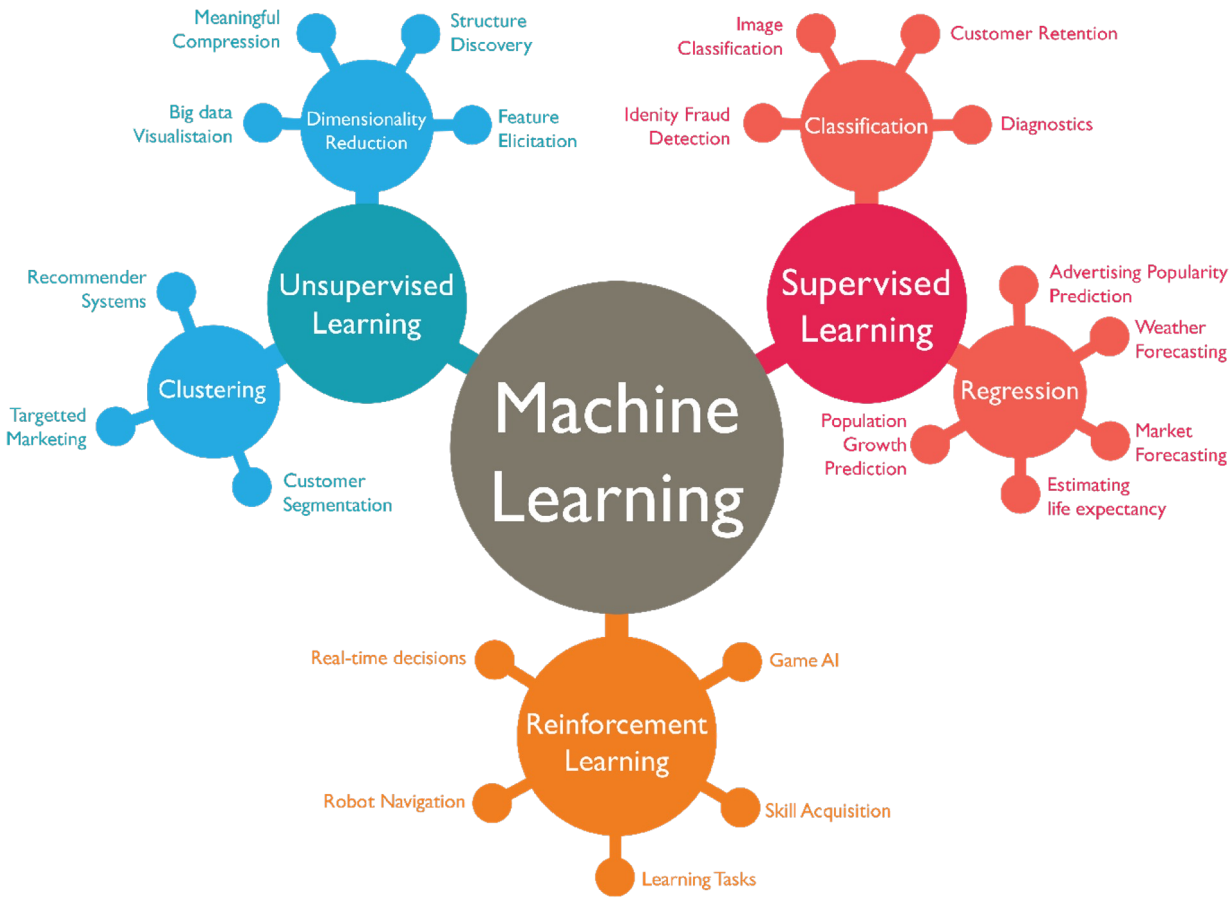
> "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at the tasks improves with the experiences."
> --- (Mitchell 1997)

Next let's take a deeper dive into:

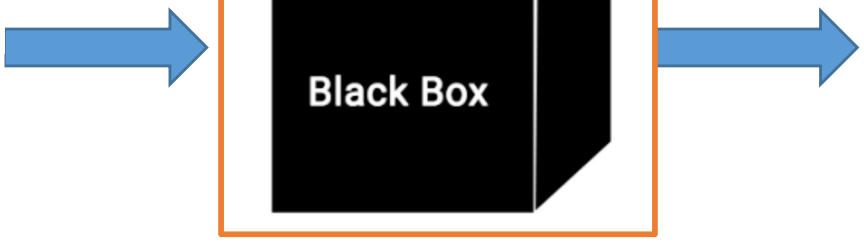
- Common ML tasks
- How do we gain experience and workflow for learning from experience
- Process for evaluating performance

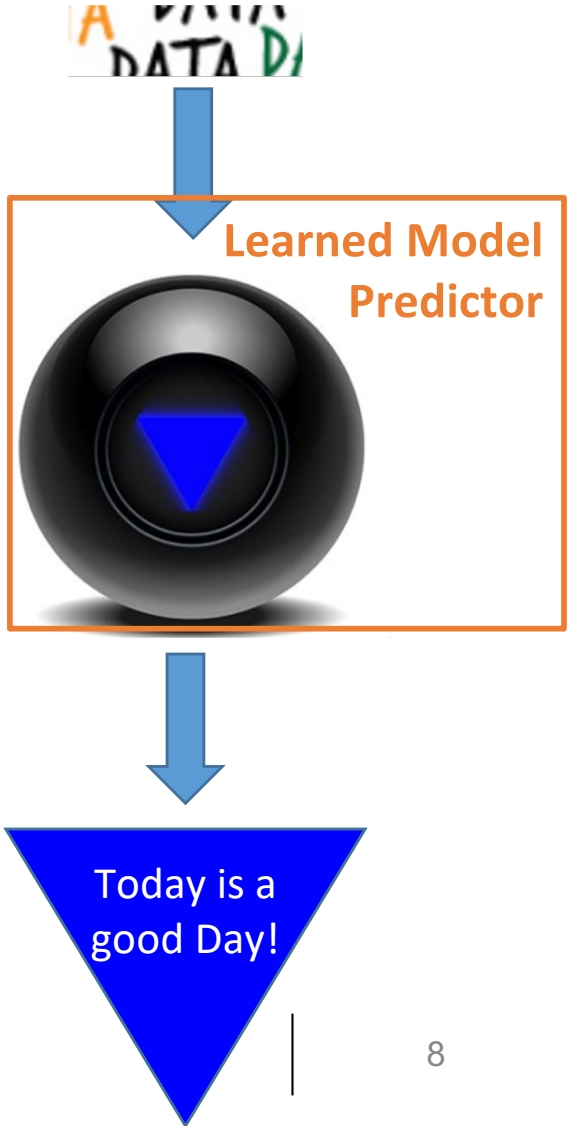
Common Task for “Learning”

- Supervised Learning
  - Data has Labels
- Unsupervised Learning
  - Data has NO Labels
- Reinforcement Learning
  - Learn from making mistakes

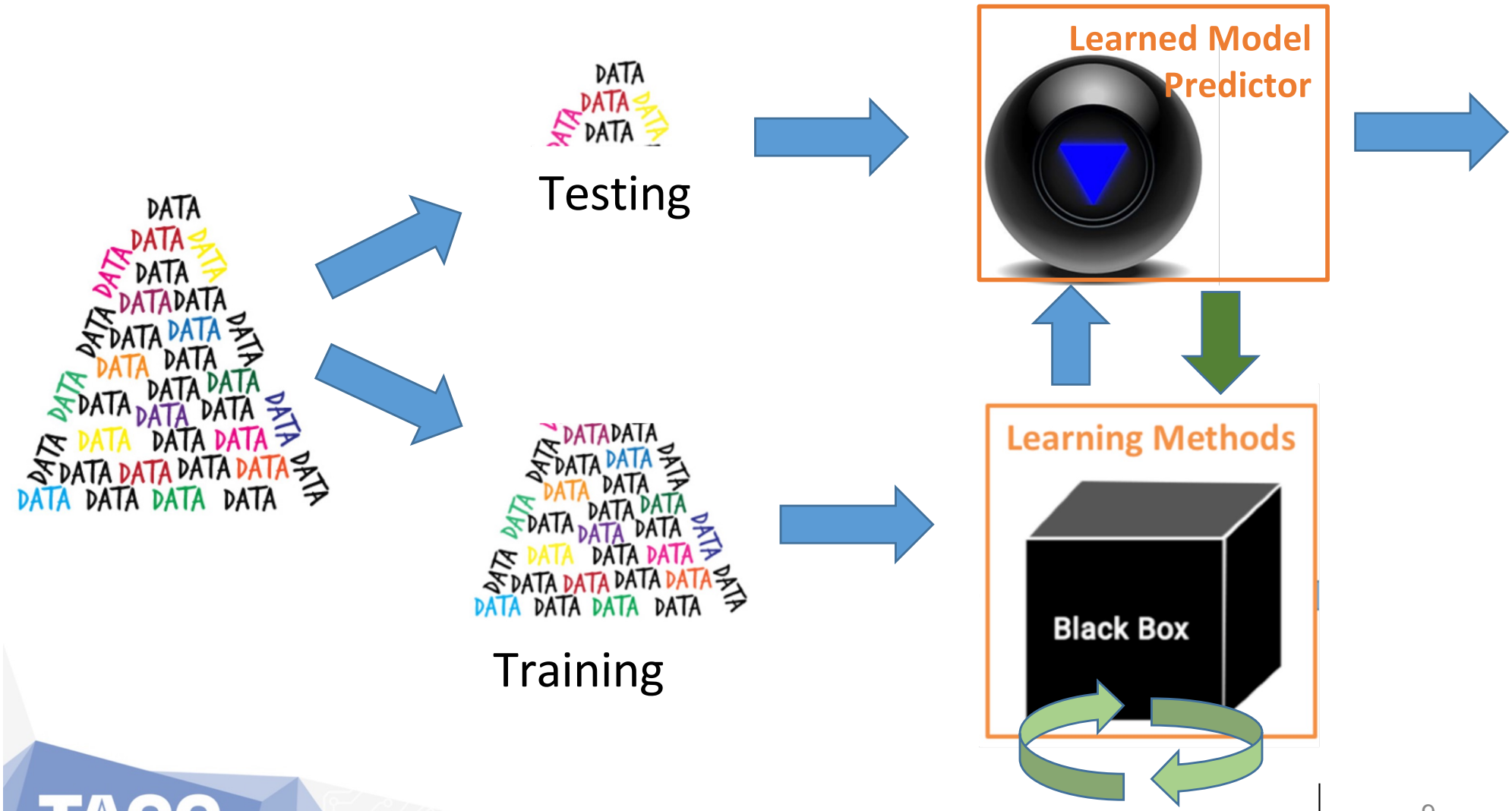


Learn from Past Experiences Data





How to Evaluate Performance of Machine Learning? --- the basic mechanism



Training and Testing

- Training
  - The process for making a model.
  - Training data is the data “observed” by the learning method
- Testing
  - Process for evaluating the performance of the model
  - Testing Data is data NOT observed by the learning model
- 80/20 split
  - 80% available data are randomly selected for training
  - The rest 20% are used for testing.
- Prediction
  - Apply model to data not in either training or testing data set.
  - Assume the input data and its prediction are from the same process producing the training data.

Performance Metrics

- Various Performance Metrics will be discussed throughout the this talk for Supervised Learning (Regression and Classification)
- Simple Example with Classification
  - Accuracy

$$\frac{number\ of\ data\ points\ classified\ correctly}{total\ number\ of\ datapoints}$$

Example: tax evasion detection

Historical data records

Data attribute/ Features

Class Label

Tid	Refund	Marital Status	Taxable Income	Evade
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Record to be inferenced

Refund	Marital Status	Taxable Income	Evade
No	Married	85k	?

Outline

- Machine Learning (ML) Basics
  - Define ML
  - Categories of ML
  - ML Workflow
- Common Supervised Learning Techniques
  - K Nearest Neighbors
  - Linear Regression
  - Regularization
  - Logistic Regression
  - Support vector machines
  - Probabilistic Models
  - Decision Tree Based Models

Classification:
K Nearest Neighbor

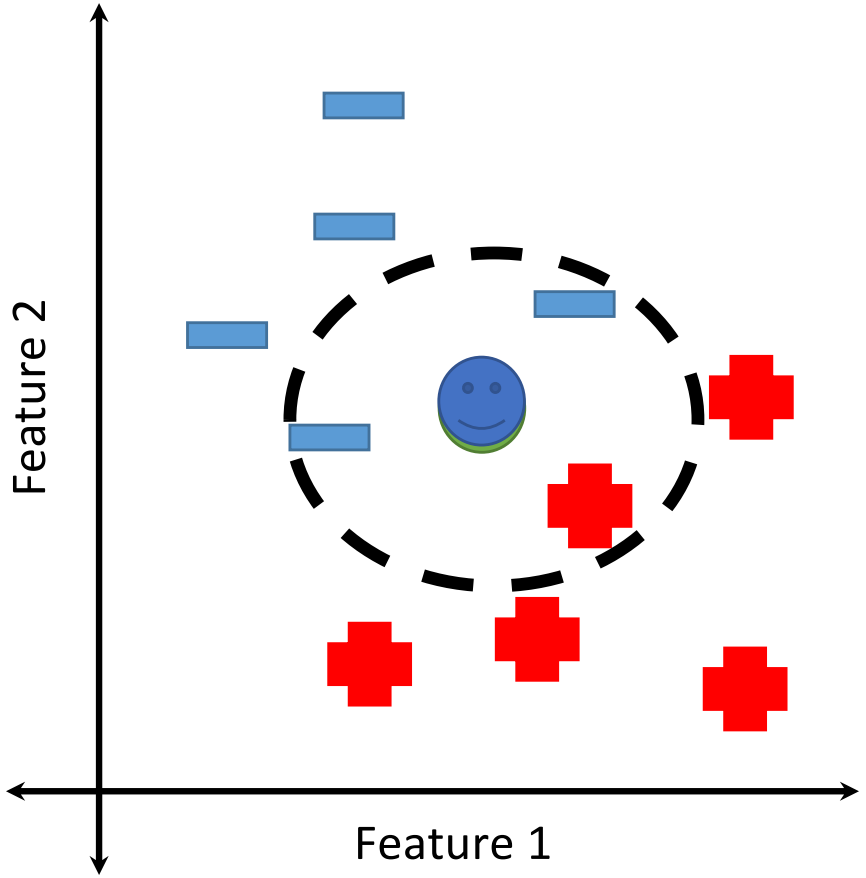
Nearest Neighbor Classification

- The basic idea:

  If two data objects are similar (close), they are likely from the same class.

- General workflow

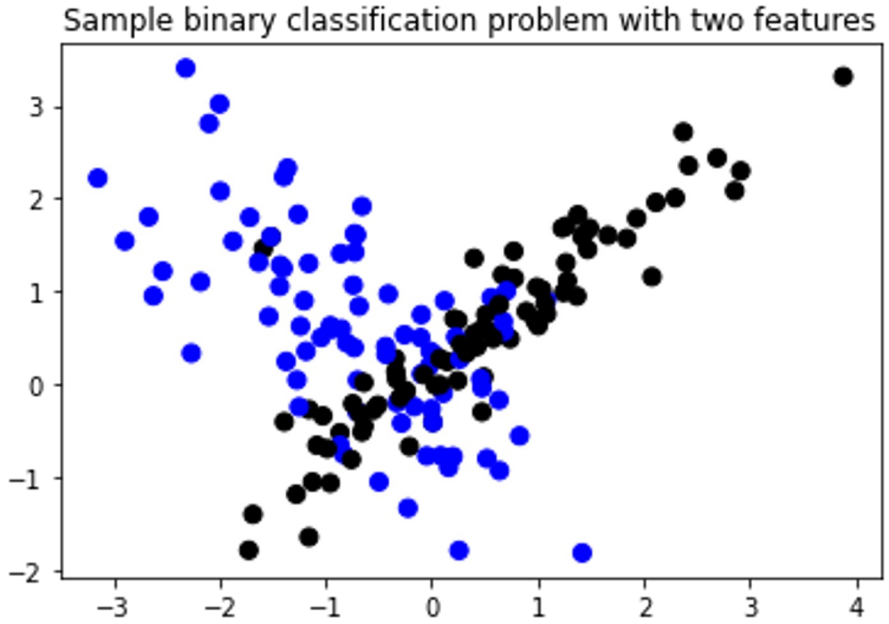
  1. Start with a set of data object with class labels
  2. Given a distance (similarity)measure of comparing data, compute distance from *unseen data point* to all data in training set
  3. Retrieve (k) nearest neighbor for the unknown data
  4. Assign class label based on the retrieved record.

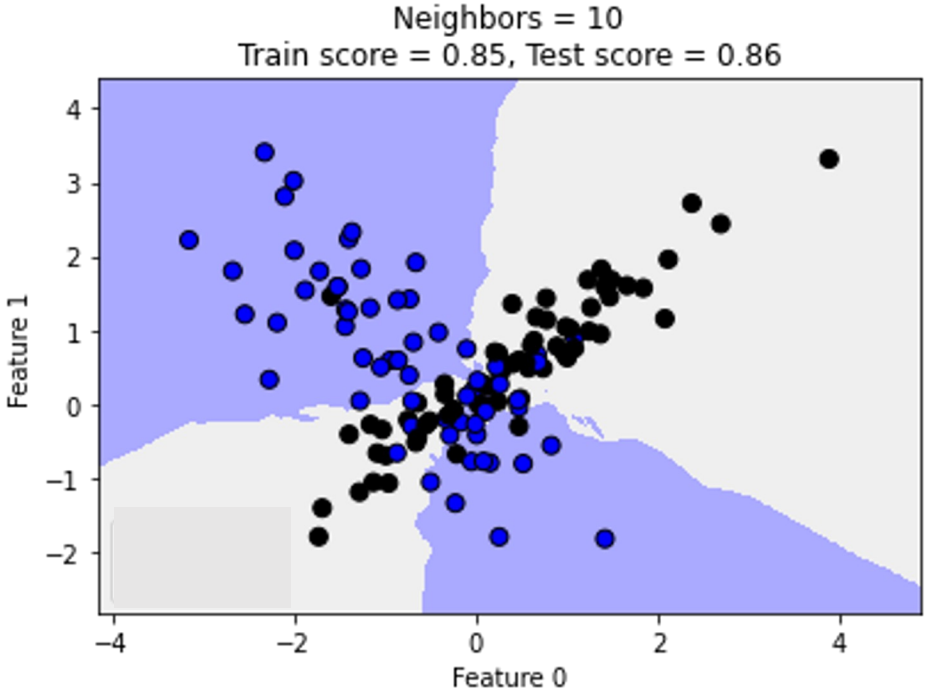


- K is a hyperparameter
- Hyperparameters are parameters users select that will impact the performance of ML models

Decision Boundary for Nearest Neighbor Classification

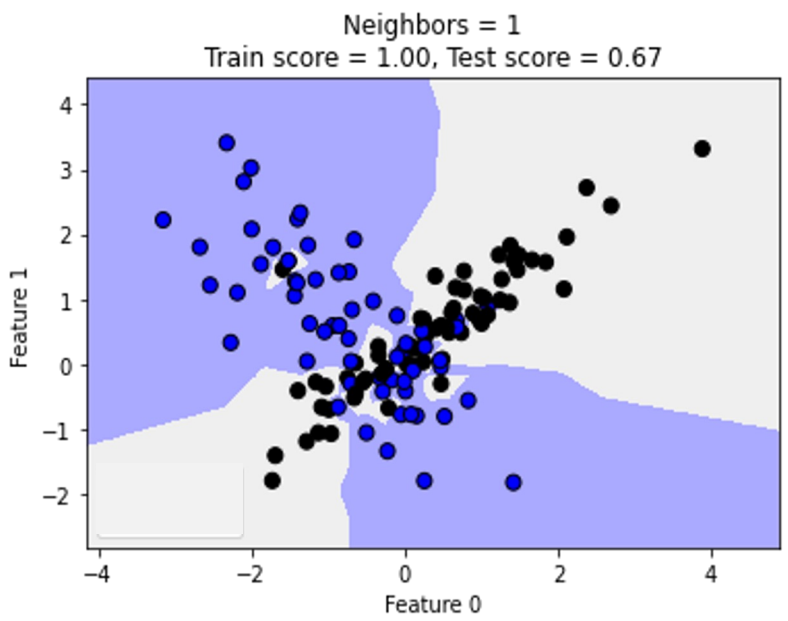
- A simple binary classification case

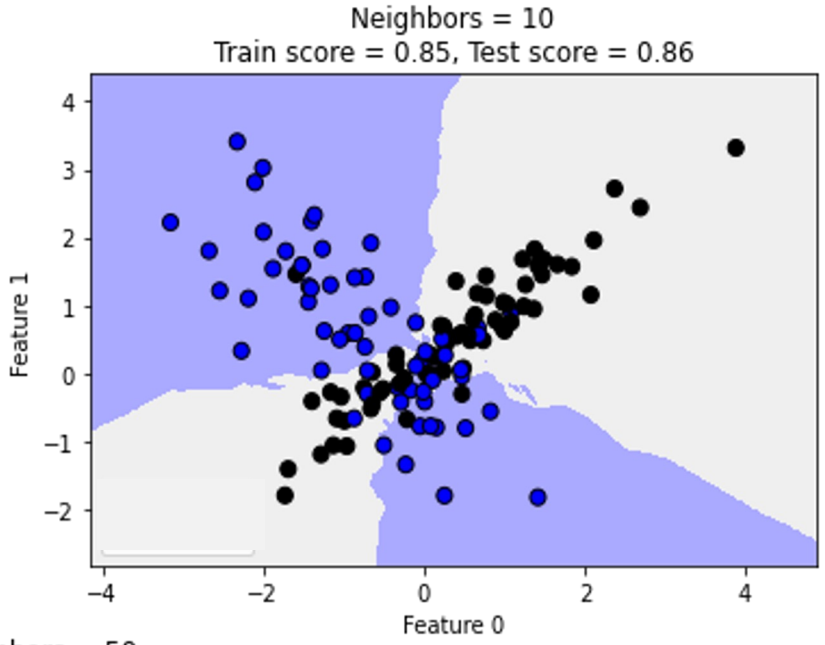


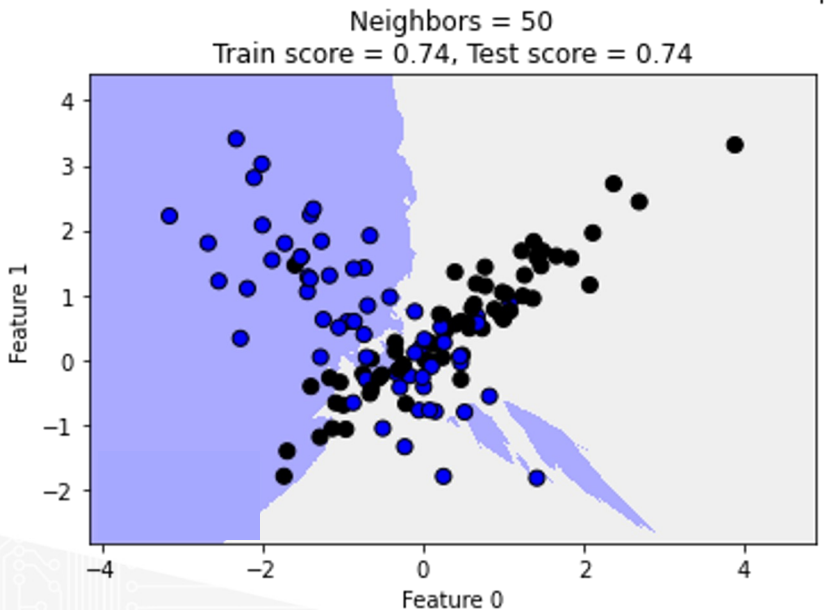


Choice of K

- What changes?

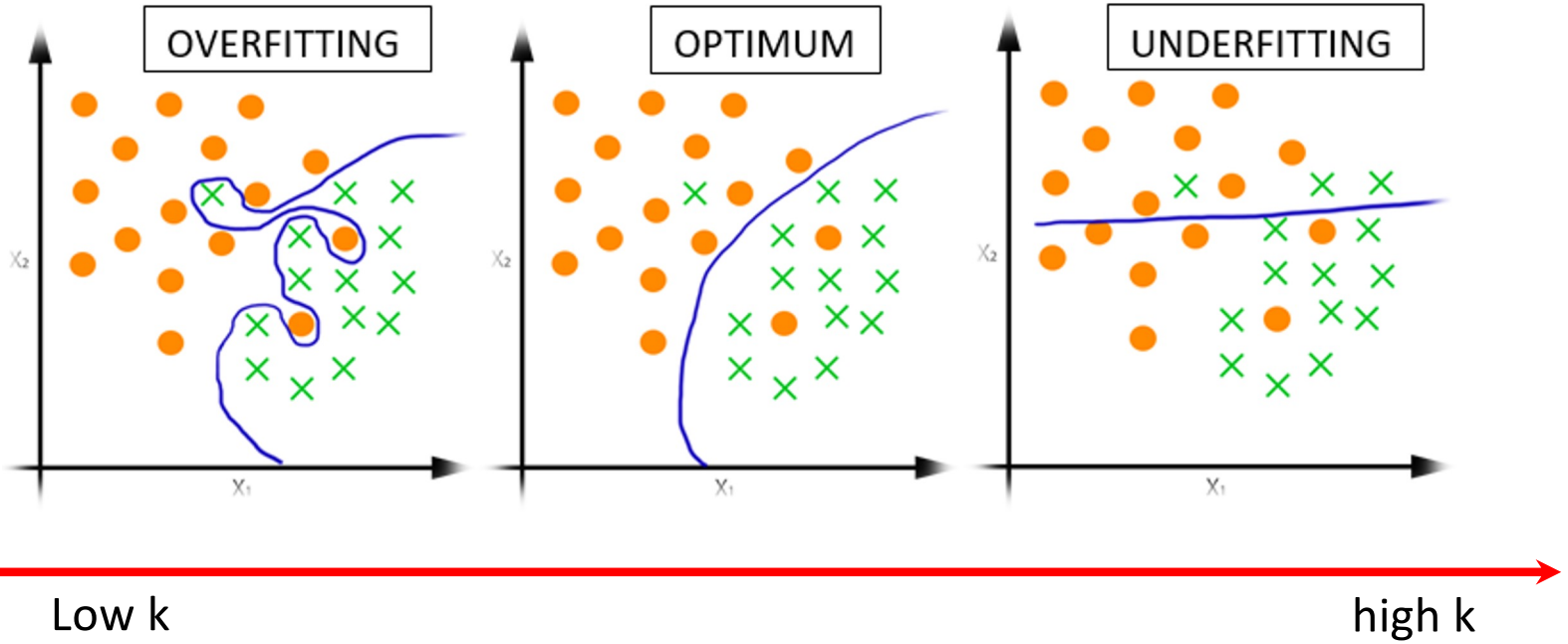






Underfitting vs Overfitting

- Underfitting,
  - model cannot reflect all the relations from training data
  - Model performs poorly on training and testing data. “failed to learn”
- Overfitting,
  - Model may lead to poor generalization on new testing data.
  - Performs very well on training data but poorly on testing data
- Hyperparameters impact under or over fitting



Classification: Additional Evaluation Metrics

Let's say we built a model predicting whether a patient has cancer:

Confusion Matrix

	Predicted - Cancer	Predicted – NOT Cancer
Actual - Cancer	0	30
Actual – NOT Cancer	0	270

This is a bad model, but what is the accuracy …

Accuracy = 270/300= 90%

Accuracy alone should not be used to evaluate a model's quality when the data is imbalanced.

Confusion Matrix, Recall, and Precision

- True or False refers to whether we classified data point correctly
- Positive or Negative refers to the **predicted** class label

Confusion Matrix

	Predicted - Positive	Predicted – Negative
Actual - Positive	True Positive (TP)	False Negative (FN)
Actual – Negative	False Positive (FP)	True Negative (TN)

$$\text{Recall} = \frac{TP}{TP + FN}$$

$$\text{Precision} = \frac{TP}{TP + FP}$$

Classification: Additional Evaluation Metrics

Let's say we built a model predicting whether a patient has cancer:

Confusion Matrix

	Predicted - Cancer	Predicted – NOT Cancer
Actual - Cancer	0	30
Actual – NOT Cancer	0	270

An Alternative to this metric is Recall

Recall = Predicted-Cancer / Data Points that are Actually Cancer = 0%

Classification: Additional Evaluation Metrics

Let's say we built a model predicting whether an email is spam:

Confusion Matrix

	Predicted - Spam	Predicted – NOT Spam
Actual - Spam	30	90
Actual – NOT Spam	30	270

Precision may make more sense since we do not want to accuse legit emails of being spam (or both recall and precision!)

Precision = Predicted-Spam / Data Points that are Predicted Spam = 50%

Regression:
Linear Regression

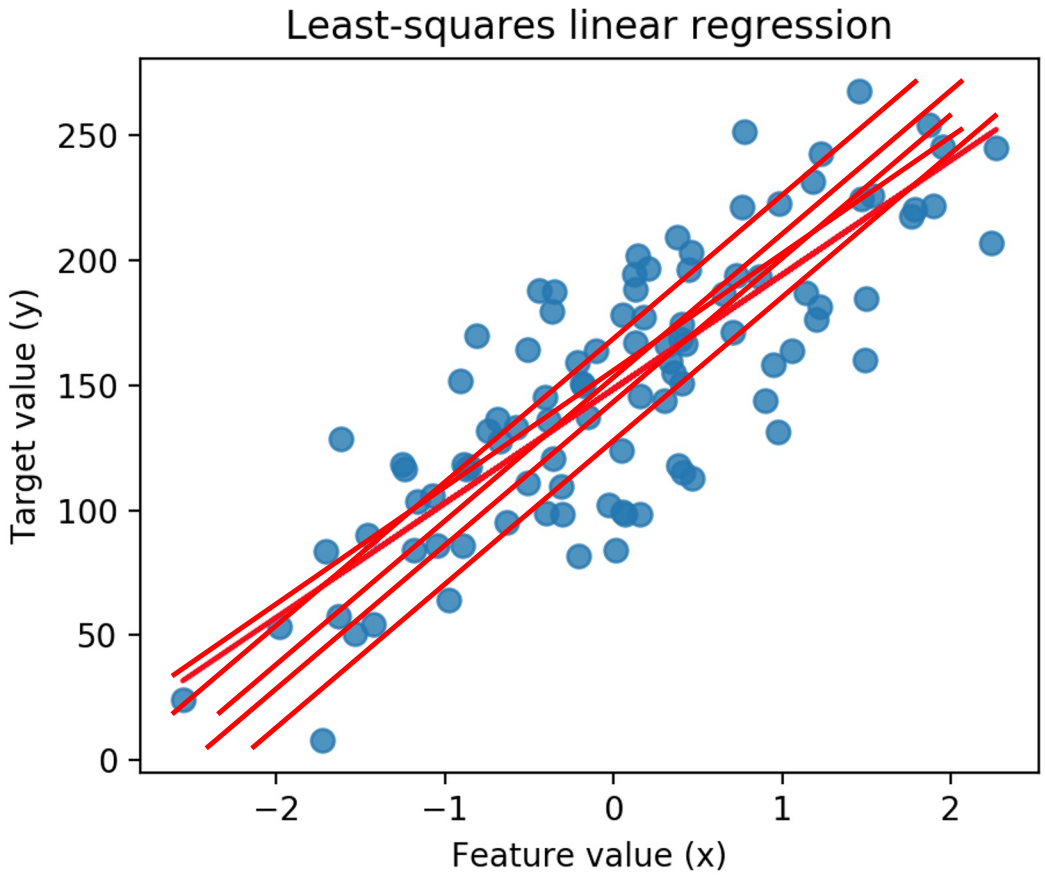
Regression Model

- The independent variables X
- The dependent variables Y
  outcome, target, or criterion variable
- The unknown parameters $\boldsymbol{\theta}$
- Predict/estimate Y with f(X, $\boldsymbol{\theta}$)

Years experience	Salary
1.1	41,000
1.7	52,000
3	53,000
4.2	70,000

Simple Linear Regression

- Only one independent variable (x) and one dependent variable (y)
- Assuming label and feature are connected through a function e.g. y=ax+b
- $\theta = \langle a, b \rangle$
- $f(x, \theta) = ax + b$



Common Goal of the Regression

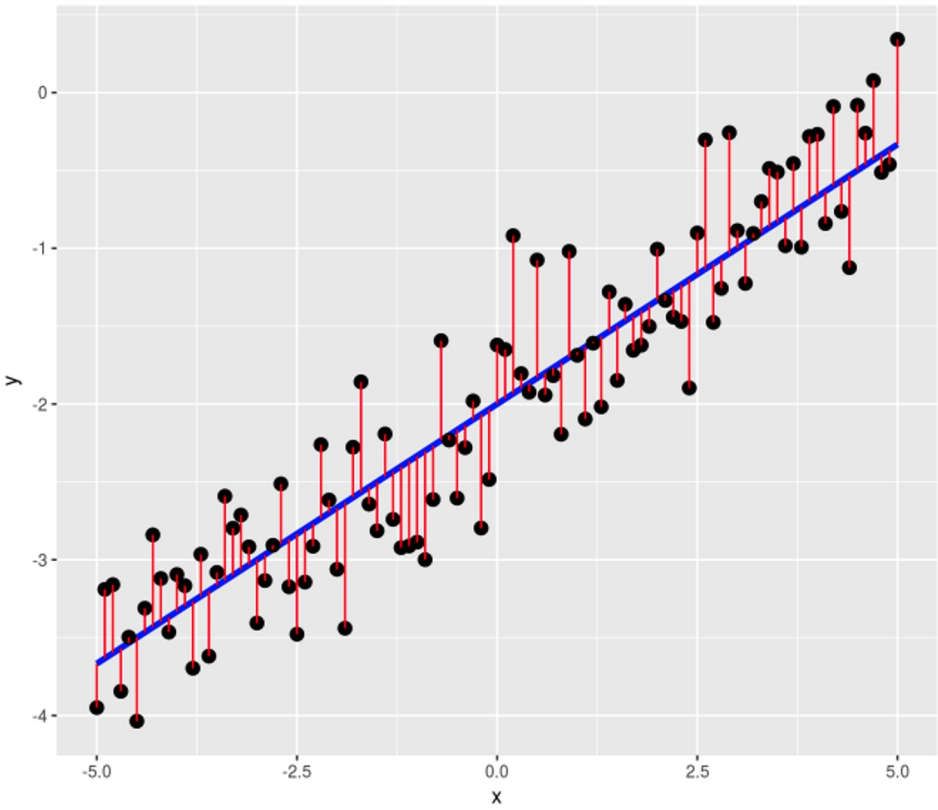
- Learn the parameters by minimize the cost or objective function

- Least absolute deviations: to minimize

$$\sum_{i=1}^{N} |f(x_i, \theta) - y_i|$$

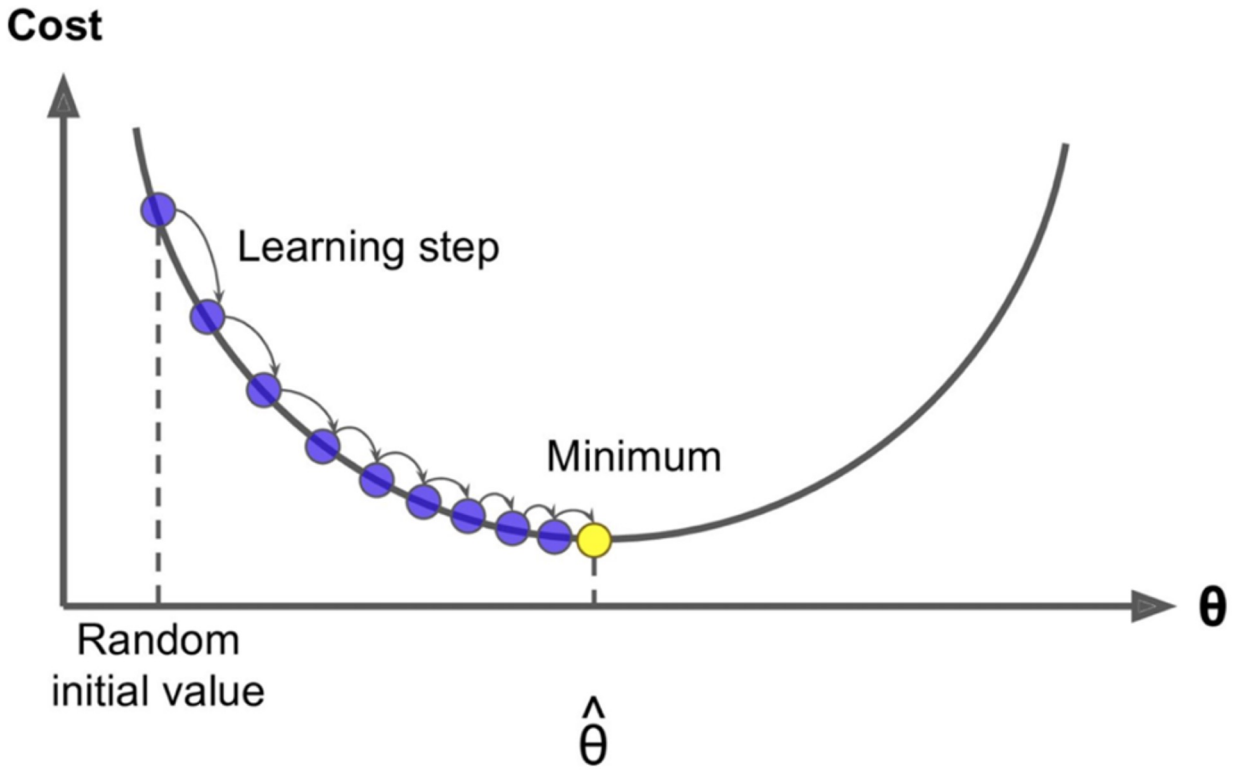
- Ordinary least squares (OLS): to minimize

$$\sum_{i=1}^{N} (f(x_i, \theta) - y_i)^2$$



Training Models by Minimize “errors”

- A basic idea in many ML is to minimize loss function
- Numerical optimization methods like gradient descent are commonly used to find optimal values for θ

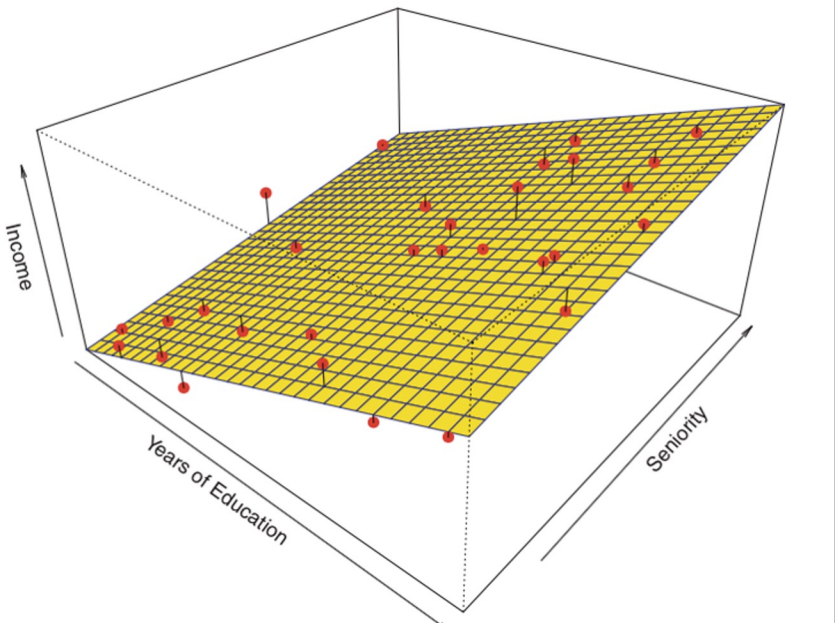


Multiple Linear Regression

- More than one independent variable
- x is a vector of features, $x \in R^N$
- $\boldsymbol{\theta}$ is a vector of weights, $\boldsymbol{\theta} \in R^N$
- $\theta = \langle b, a_0, a_1, \ldots a_N \rangle$

$$f(x, \theta) = b + a_0 x_0 + a_1 x_1 + \cdots + a_N x$$

Years of Education	Years experience	Salary
13	1.1	41,000
17	1.7	52,000
19	3	53,000
22	4.2	70,000



Common Metrics of Performance

- Mean Squared Error

$$\frac{1}{N}\sum_{i=1}^{N}(f(x_i,\theta)-y_i)^2$$

- Mean Absolute Error

$$\frac{1}{N}\sum_{i=1}^{N}|f(x_i,\theta)-y_i|$$

Common Metrics of Performance

- From OLS, Residue sum of squares

$$SS_{\text{res}} = \sum_i (y_i - f_i)^2$$

- Total sum of square

$$SS_{\text{tot}} = \sum_i (y_i - \bar{y})^2.$$

- Coefficient of determination, aka, R^2
  - The proportion of the variance in the dependent variable that is predictable from the independent variable(s)

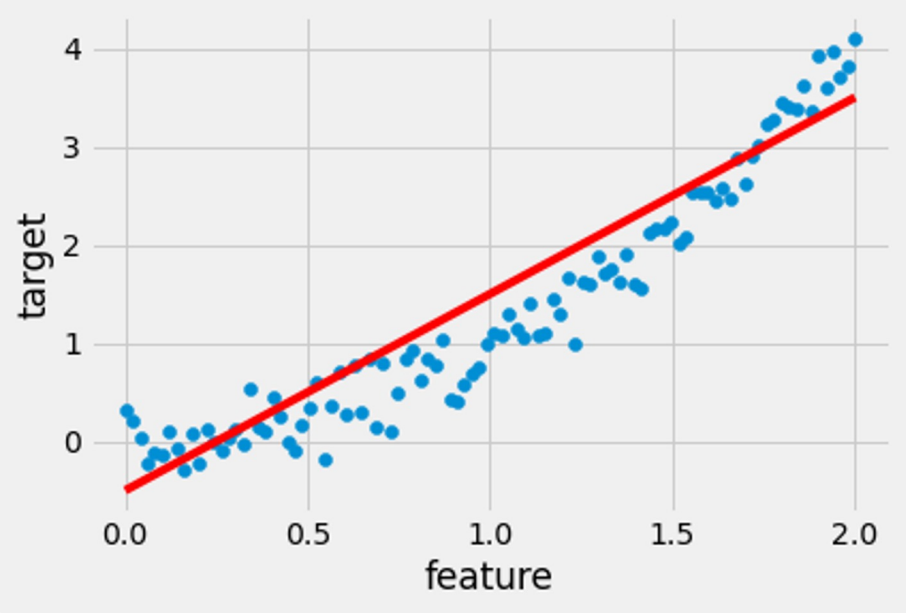
$$R^2 \equiv 1 - \frac{SS_{\text{res}}}{SS_{\text{tot}}}.$$

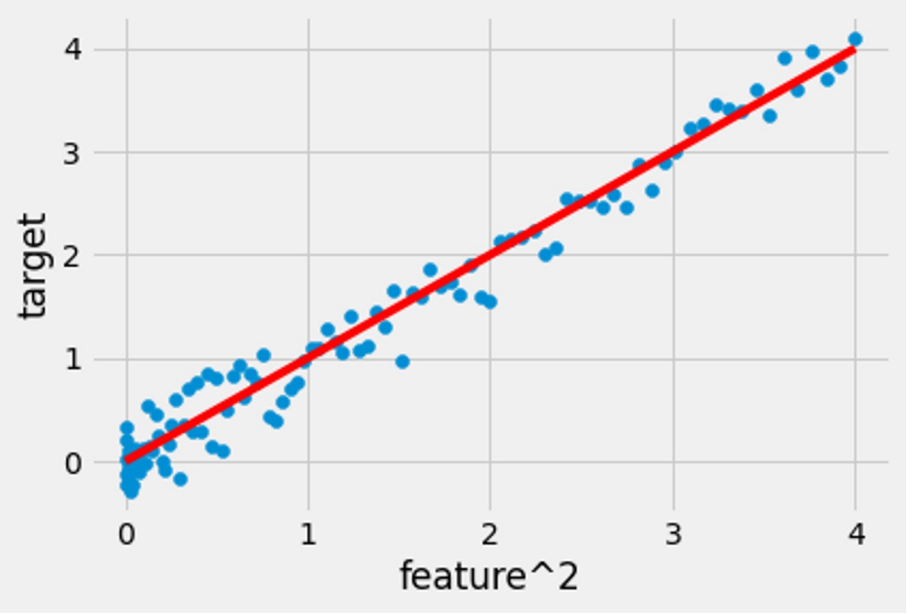
Practical Considerations of Linear Regression

- Pros
  - Linear regression is simple
  - It is very interpretable
    - Slopes, a, can be defined as
    - Holding all else constant, for 1 unit increase in feature x_i the target variable increases by a_i
- Cons
  - not all features in your data have linear relationship with the target
  - need to transform features such that a linear relationship exists to get good results

Feature Engineering

You may need to change your features so that there is a linear relationship between the features and target



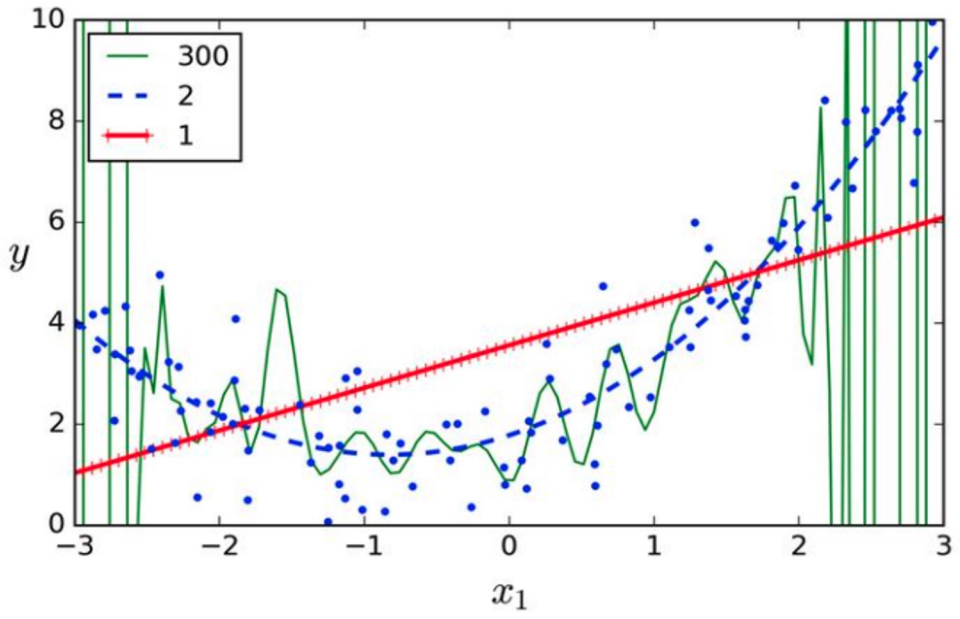


Polynomial regression

- a special case of **multiple linear regression**

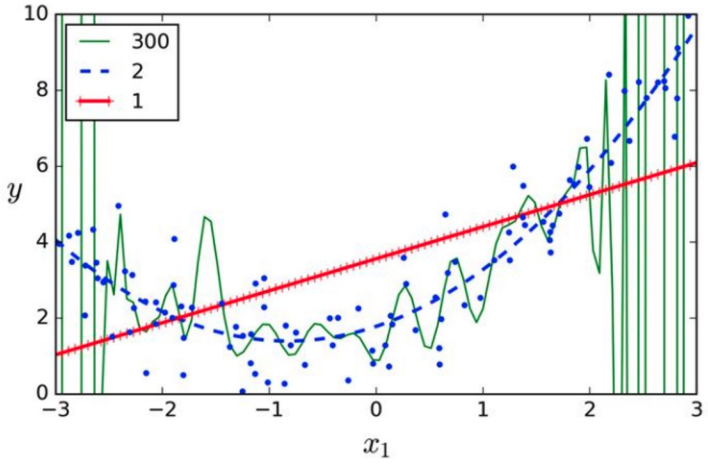
$$y = a_0 + a_1 x + a_2 x^2 + a_3 x^3 + \cdots + a_n x^n$$

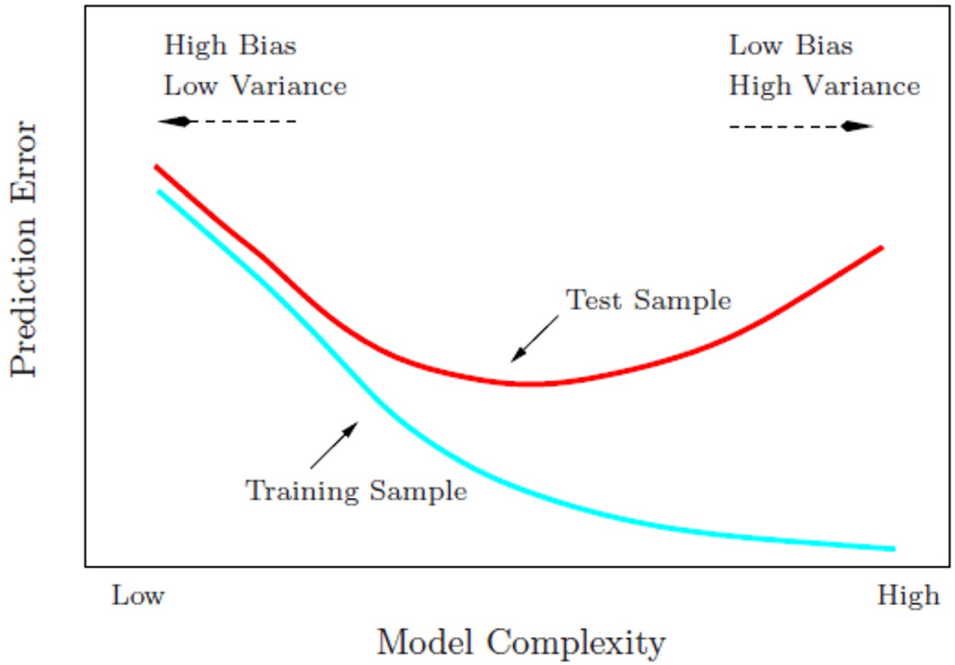
- Which is better?
  - Red line: linear regression
  - Blue line: 2-degree polynomial regression
  - Green line: 300-degree polynomial regression



Underfitting and Overfitting

- Model Complexity ≈ number of parameters in a model
- High degree polynomial or using too many features can lead to over fitting
- How do we know what the appropriate polynomial degree is?





Ridge Regression

- One way to combat this overfitting is to use **regularization**. An example is Ridge regression.
- Alters cost or objective function to avoid overfitting
- OLS tries to minimize

$$\sum_{i=1}^{N}(f(x_i,\theta)-y_i)^2$$

- Ridge regression adds a regularization term

$$\sum_{i=1}^{N}(f(x_i,\theta)-y_i)^2+\alpha\sum_{k=1}^{p}a_k^2$$

- Where N is the number of training points, p is the number of features, and a_i is the slopes associated with feature i
- α – regularization hyperparameter – is a positive number, when α =0 ridge regression is effectively the same as OLS.

Standardizing Data

- When using Ridge regression (or the similar Lasso regression) it is important to standardize your data.
- Why? Let's say we are predicting MPG of vehicles given the features
  - weight of vehicle in lbs
  - number of cylinders
- Which beta coefficients will be larger?
  - even though weight is probably the most important, we may see that it has a smaller slope associated with due to the large magnitudes of weight of the vehicles.
- In Ridge regression, we want all our features beta coefficients to have a fair comparison
  - Solution: Standardize data!

Standardizing Data

There are other ML algorithms where it is important to standardize data. For example,

- KNN
- Any time using a numerical optimizer of a loss function; speeds up gradient descent-like algorithms
- Any time adding regularization term to loss function

Hands on Exercise:sklearn

- Free open source machine learning library for Python programming language
- Has implementations of all ML algorithms covered today

Sklearn

```
# import needed objects
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split

# get data
data = get_data(...)
X = data.features
y = data.targets

# test train split
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.6, random_state = 0)

# instantiante then fit model with training data
knn = KNeighborsClassifier(n_neighbors = 10)
knn.fit(X_train, y_train)

# evaluate model
test_accuracy = knn.score(X_test, y_test)
train_accuracy = knn.score(X_train, y_train)

# make predictions
y_pred=knn.predict(X_test)
```

Sklearn

```
# import needed objects
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split

# get data
data = get_data(...)
X = data.features
y = data.targets

# test train split
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.6, random_state = 0)

# instantiante then fit model with training data
knn = KNeighborsClassifier(n_neighbors = 10)
knn.fit(X_train, y_train)

# evaluate model
test_accuracy = knn.score(X_test, y_test)
train_accuracy = knn.score(X_train, y_train)

# make predictions
y_pred=knn.predict(X_test)
```

Sklearn

```
# import needed objects
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split

# get data
data = get_data(...)
X = data.features
y = data.targets

# test train split
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.6, random_state = 0)

# instantiante then fit model with training data
knn = KNeighborsClassifier(n_neighbors = 10)
knn.fit(X_train, y_train)

# evaluate model
test_accuracy = knn.score(X_test, y_test)
train_accuracy = knn.score(X_train, y_train)

# make predictions
y_pred=knn.predict(X_test)
```

Sklearn

```
# import needed objects
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split

# get data
data = get_data(...)
X = data.features
y = data.targets

# test train split
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.6, random_state = 0)

# instantiante then fit model with training data
knn = KNeighborsClassifier(n_neighbors = 10)
knn.fit(X_train, y_train)

# evaluate model
test_accuracy = knn.score(X_test, y_test)
train_accuracy = knn.score(X_train, y_train)

# make predictions
y_pred=knn.predict(X_test)
```

Sklearn

```
# import needed objects
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split

# get data
data = get_data(...)
X = data.features
y = data.targets

# test train split
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.6, random_state = 0)

# instantiante then fit model with training data
knn = KNeighborsClassifier(n_neighbors = 10)
knn.fit(X_train, y_train)

# evaluate model
test_accuracy = knn.score(X_test, y_test)
train_accuracy = knn.score(X_train, y_train)

# make predictions
y_pred=knn.predict(X_test)
```

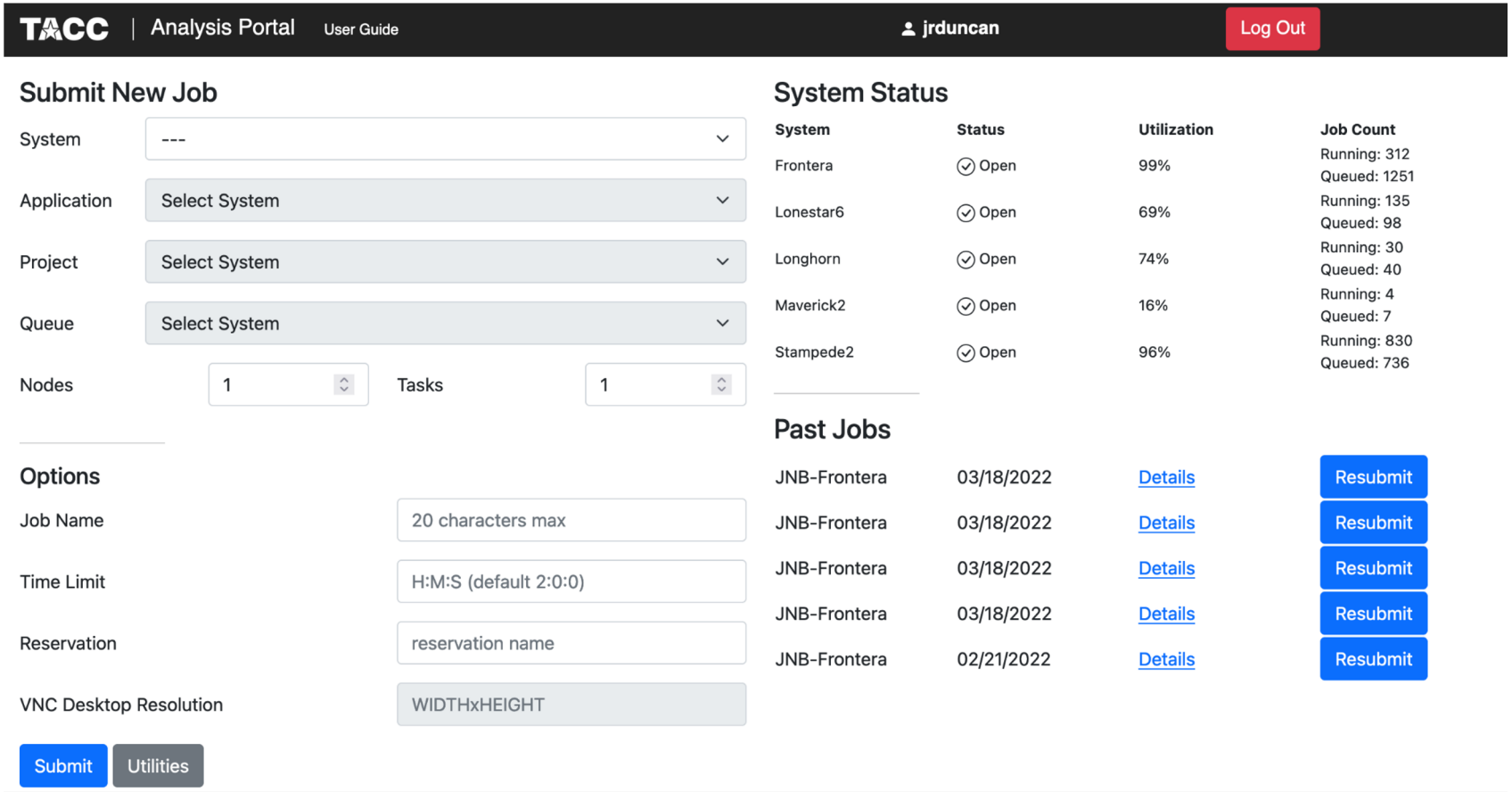
Sklearn

```
# import needed objects
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split

# get data
data = get_data(...)
X = data.features
y = data.targets

# test train split
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.6, random_state = 0)

# instantiante then fit model with training data
knn = KNeighborsClassifier(n_neighbors = 10)
knn.fit(X_train, y_train)

# evaluate model
test_accuracy = knn.score(X_test, y_test)
train_accuracy = knn.score(X_train, y_train)

# make predictions
y_pred=knn.predict(X_test)
```

Jupyter notebook

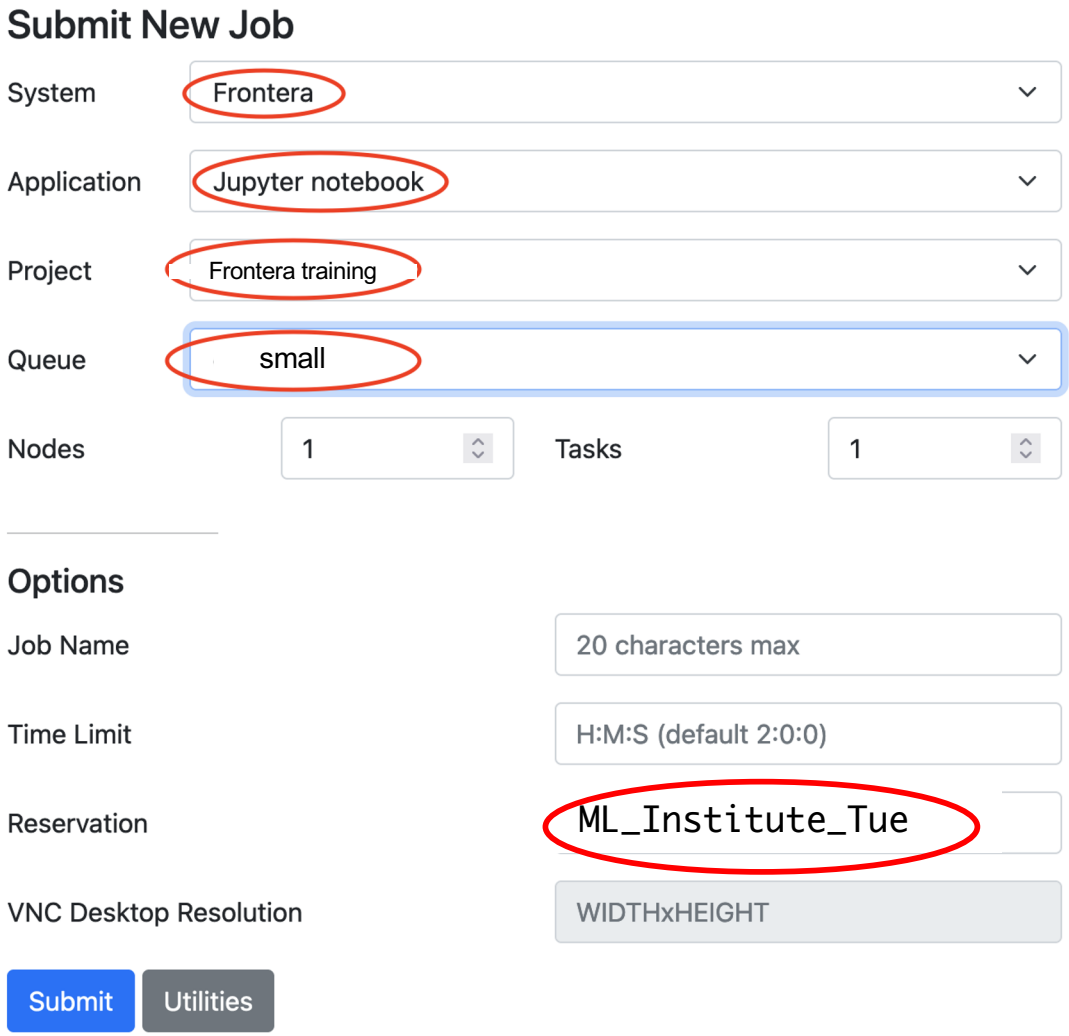
- TACC Analysis Portal:

  Go to https://vis.tacc.utexas.edu; Login with your training account credentials

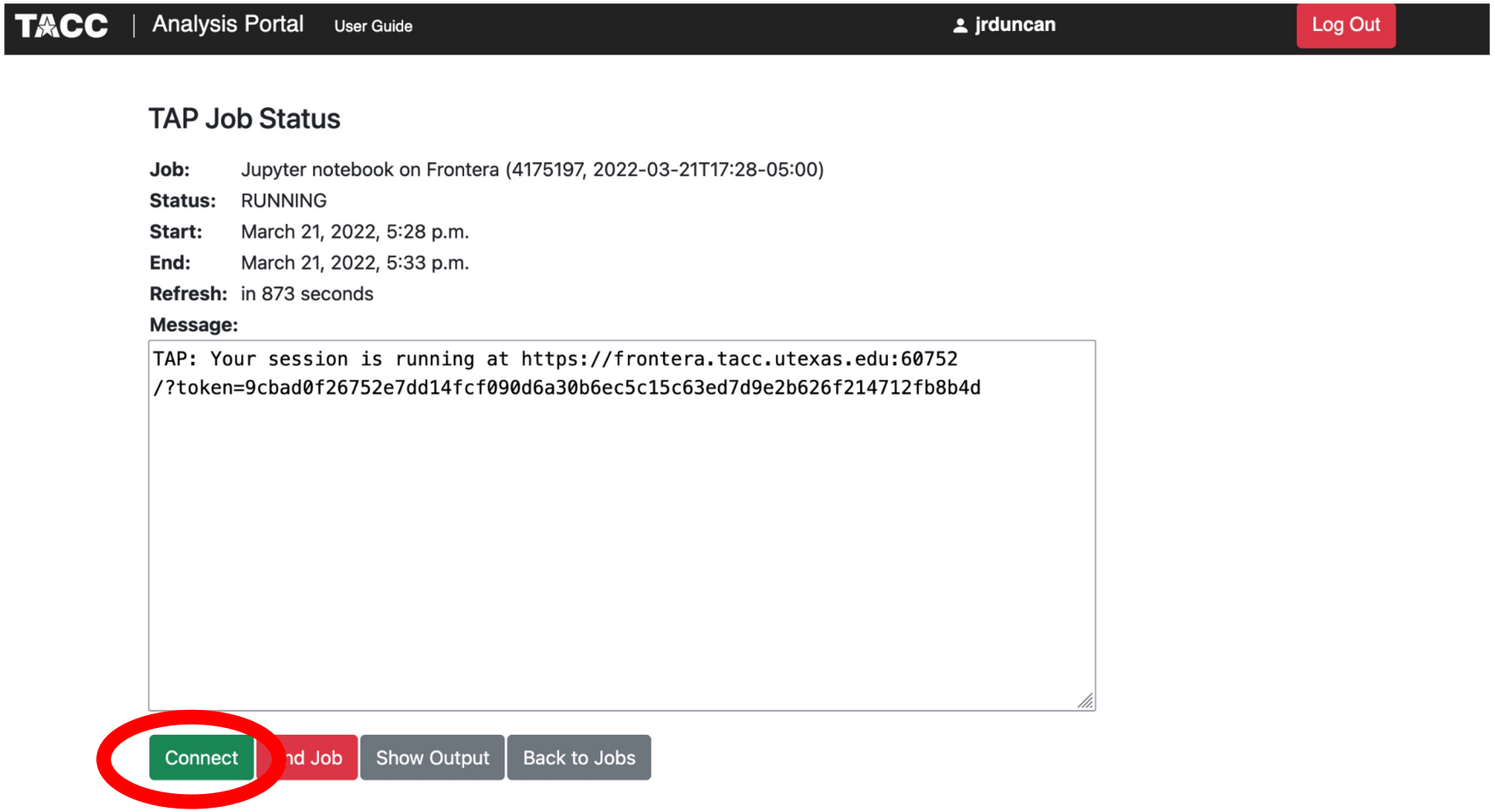


Jupyter notebook

- TACC Visualization Portal:
  Select the following options



Jupyter notebook



Morning Exercise

- You can find today's hand on exercises in `ml_institute_summer_24/day2`
- In this directory is a jupyter notebook which will use KNN, linear regression, and the algorithms that will be introduced this afternoon in the notebooks called **SupervisedLearning.ipynb**
- This notebook contains several optional exercises

Hands-on Exercise / Break

Outline

- Machine Learning (ML) Basics
  - Define ML
  - Categories of ML
  - ML Workflow
- Common Supervised Learning Techniques
  - K Nearest Neighbors
  - Linear Regression
  - Regularization
  - Logistic Regression
  - Support vector machines
  - Probabilistic Models
  - Decision Tree Based Models

Logistic Regression

(Binary) Logistic Regression

- Used for classification problems with a (Binary) dependent variable
- ”y” can only take two values (usually 0 and 1)

  Positive/Negative:

  Pass/fail

  Heathy/sick
- One or more independent variables x
- Predict the probability of a binary response

How Logistic Regression Works

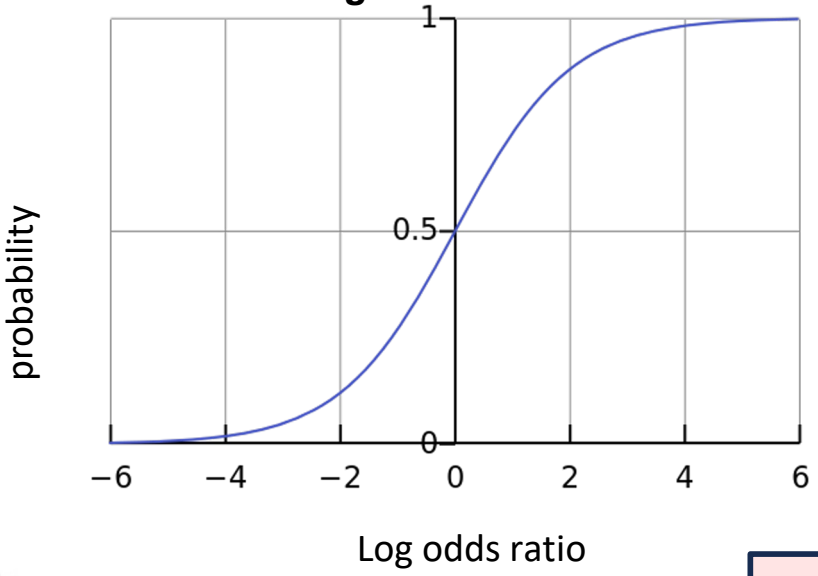
Linear Regression for classification?

$$pr\ldots y = f(x,\theta) = b + a_0x_0 + a_1x_1 + \cdots + a_Nx_N$$

Instead linear model predicts the log odds ratio

$$t = ln\left(\frac{p}{1-p}\right) = f(x,\theta) = b + a_0x_0 + a_1x_1 + \cdots + a_Nx_N$$

Logistic Function



Logistic Function

Converts log odds ratio to probabilities

$$\sigma(t) = \frac{1}{1+e^{-t}}$$

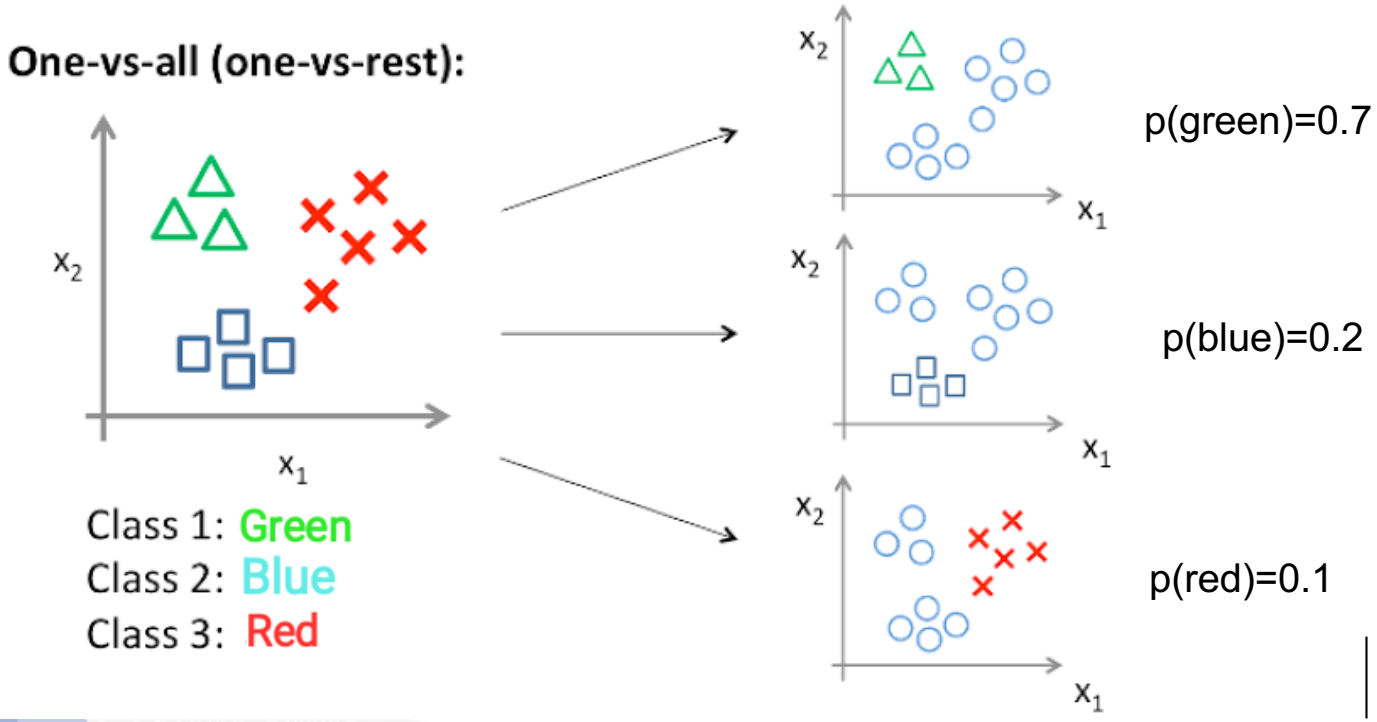
Loss Function for Classification

$$loss = \sum_{i=1}^{n} y_i \log(p_i) + (1-y_i)\log(1-p_i)$$

$$p(x,\theta) = \sigma(b + a_0x_0 + a_1x_1 + \cdots + a_Nx_N)$$

Multiclass Logistic Regression

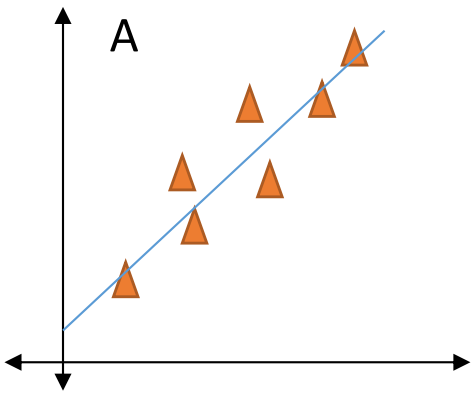
- Logistic regression can be applied to multiclass classification problems (more than 2 categories (i.e. green, blue, red))
- **One-v-rest**:
  - Build binary logistic regression for each class label vs all other labels; this produces probability of that label
  - Selects highest confidence model



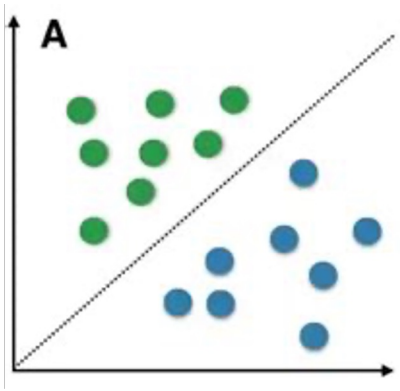
Practical Usage

- Linear regression and Logistic regression create linear models (A below)

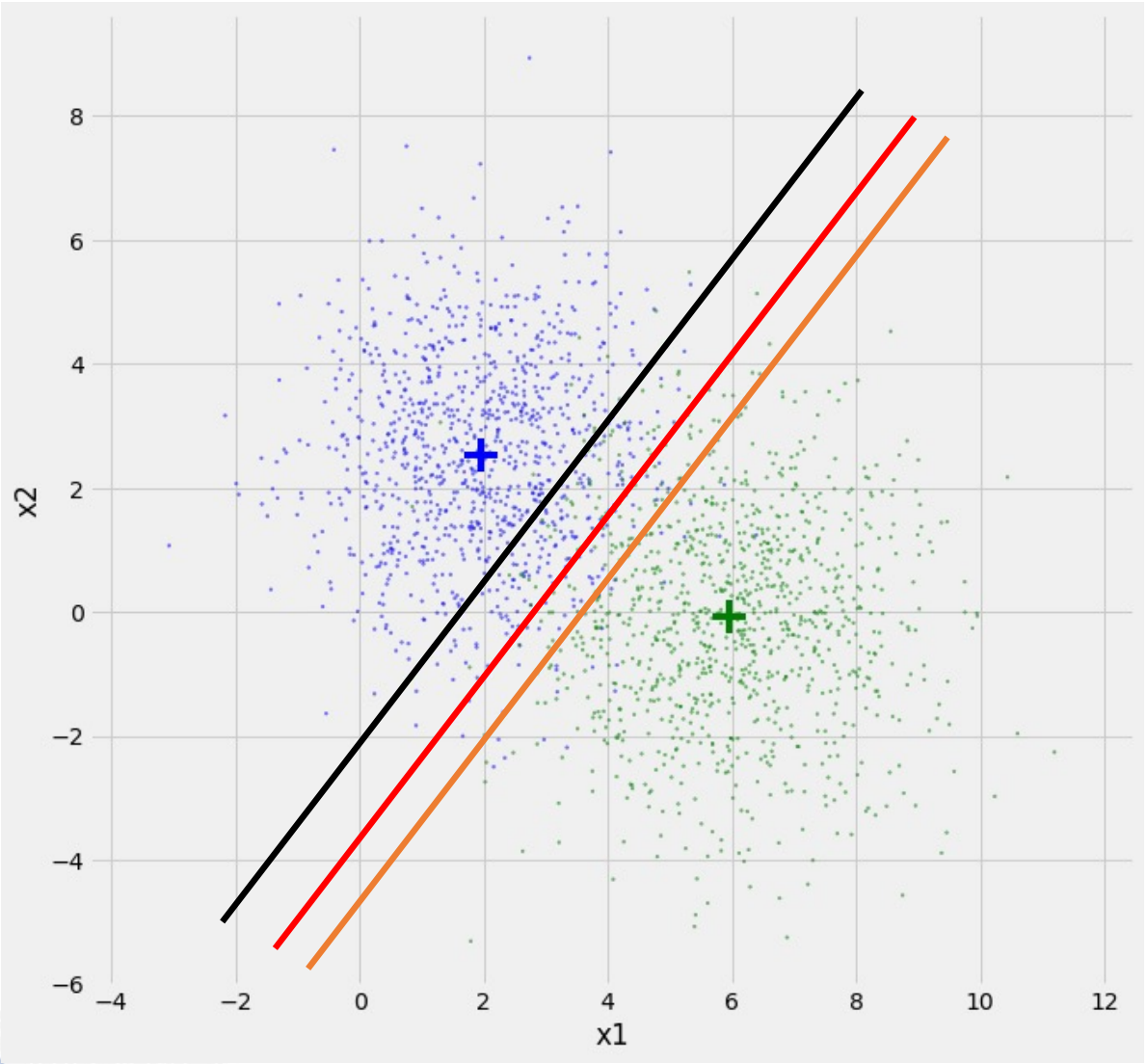
Regression



Classification

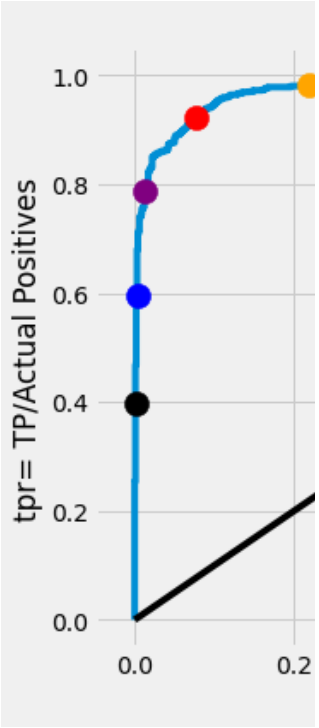


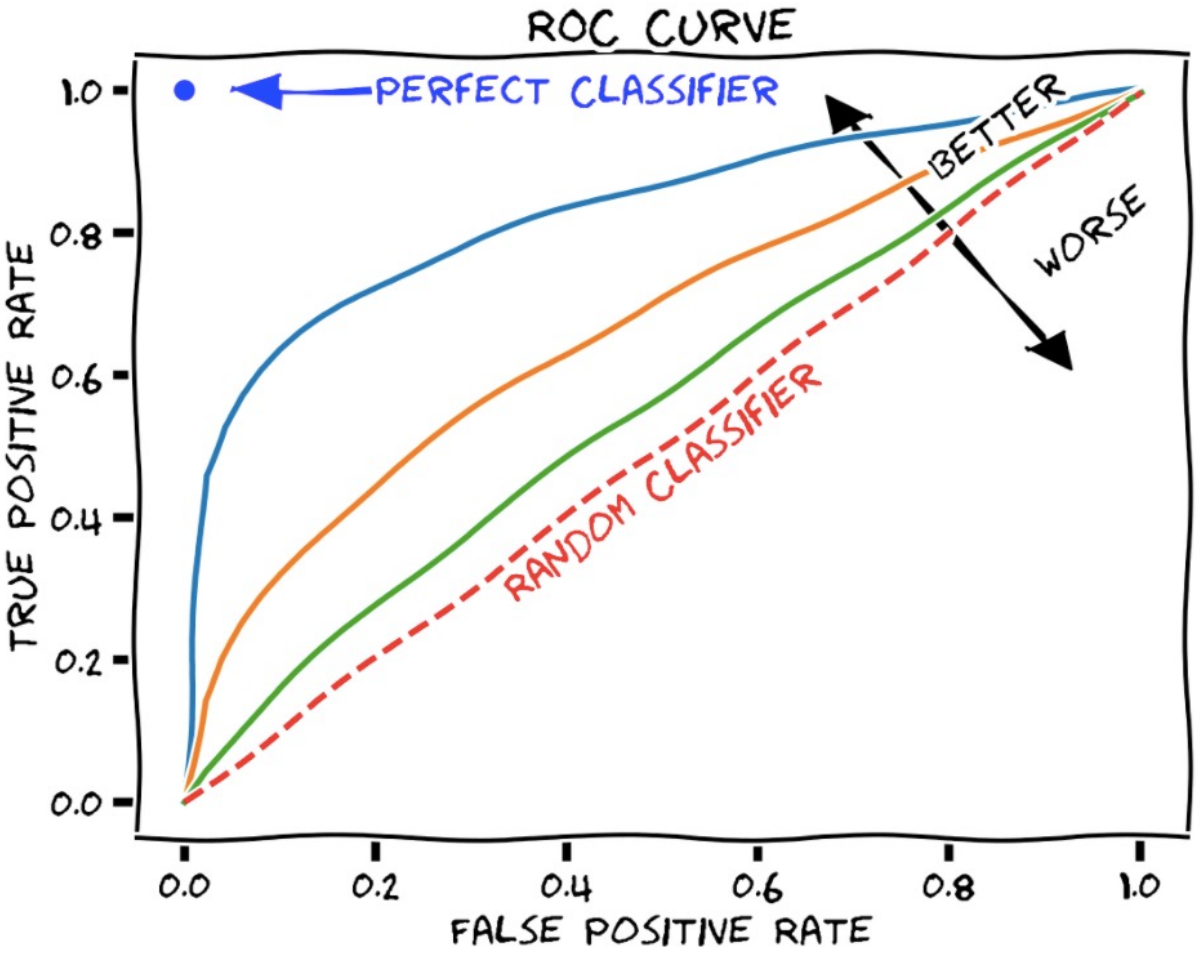
Thresholding and ROC curves

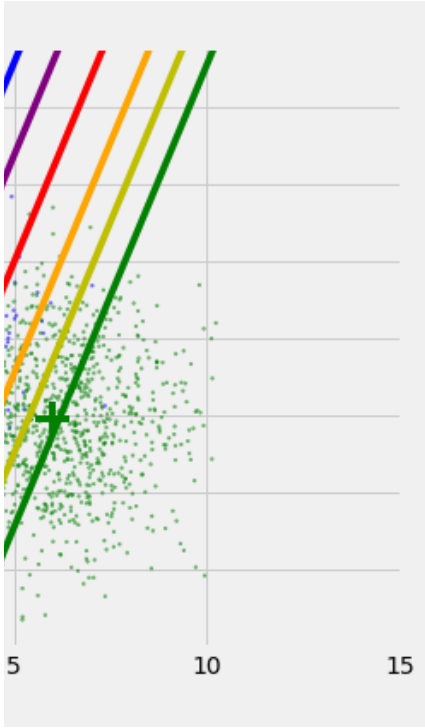


p(blue)=0.5
p(blue)=0.95
p(blue)=0.3

Model Evaluation: ROC curves







Pros and Cons of Linear Models

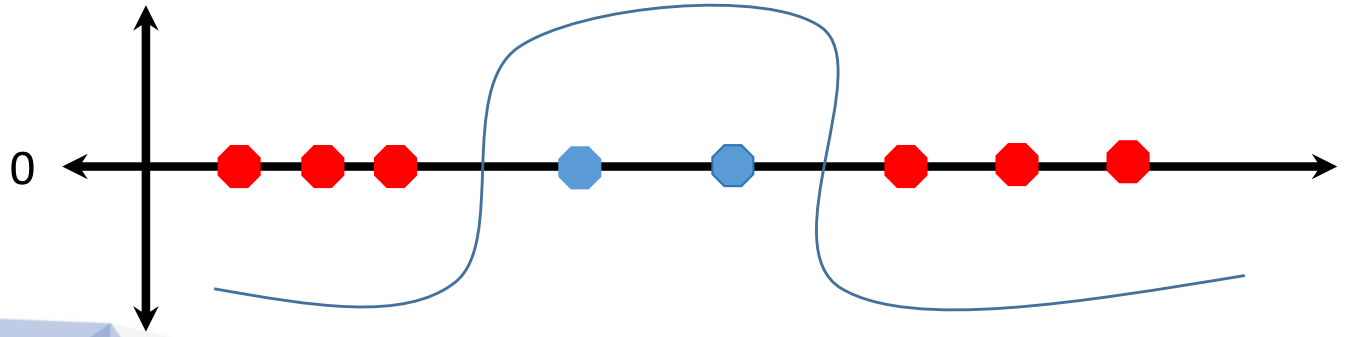
- Pros
  - Simple
  - Interpretable
    - Holding all else constant, for 1 unit increase in feature x_i the response variable y increases by a_i
  - If linear relationship exist, use linear models
- Cons
  - Very Difficult to engineer linear relationships

Outline

- Machine Learning (ML) Basics
  - Define ML
  - Categories of ML
  - ML Workflow
- Common Supervised Learning Techniques
  - K Nearest Neighbors
  - Linear Regression
  - Logistic Regression
  - Support vector machines
  - Probabilistic Models
  - Decision Tree Based Models

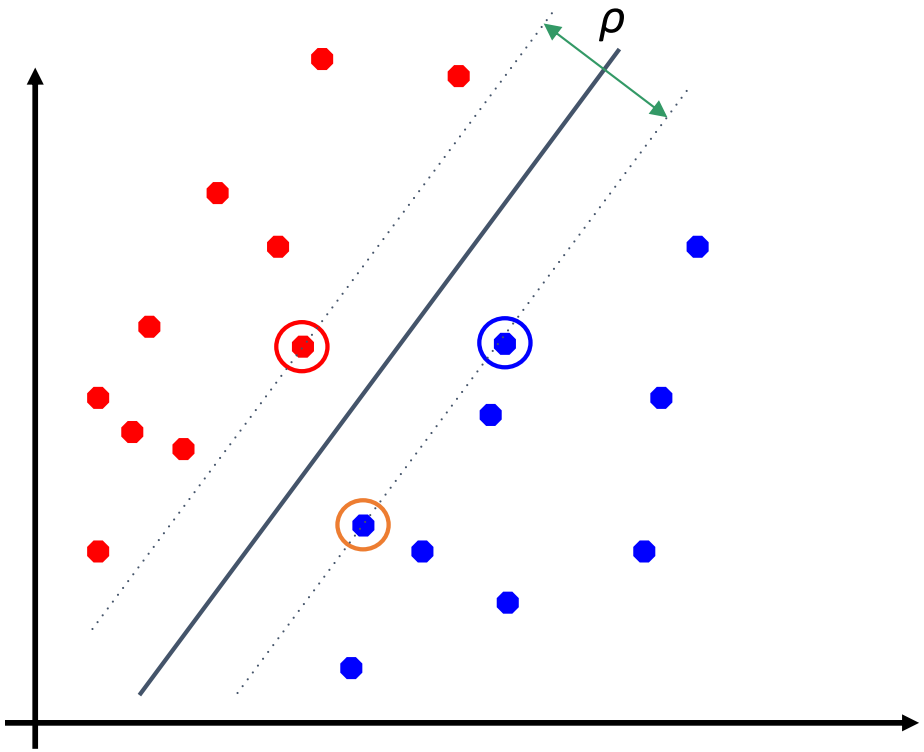
Support Vector Machine (SVM)

- Used for classification
- Given labeled training data, SVM generates an optimal hyperplane to separate classes
- Creates a decision function



Binary Classification with Linear Separator

- A line is used as the hyperplane separating two classes
- The closets data to the hyperplane are the **support vectors**
- Hyperplane is selected such that the margin is as big as possible
- A line is bad if it passes too close to some points (noise sensitive)



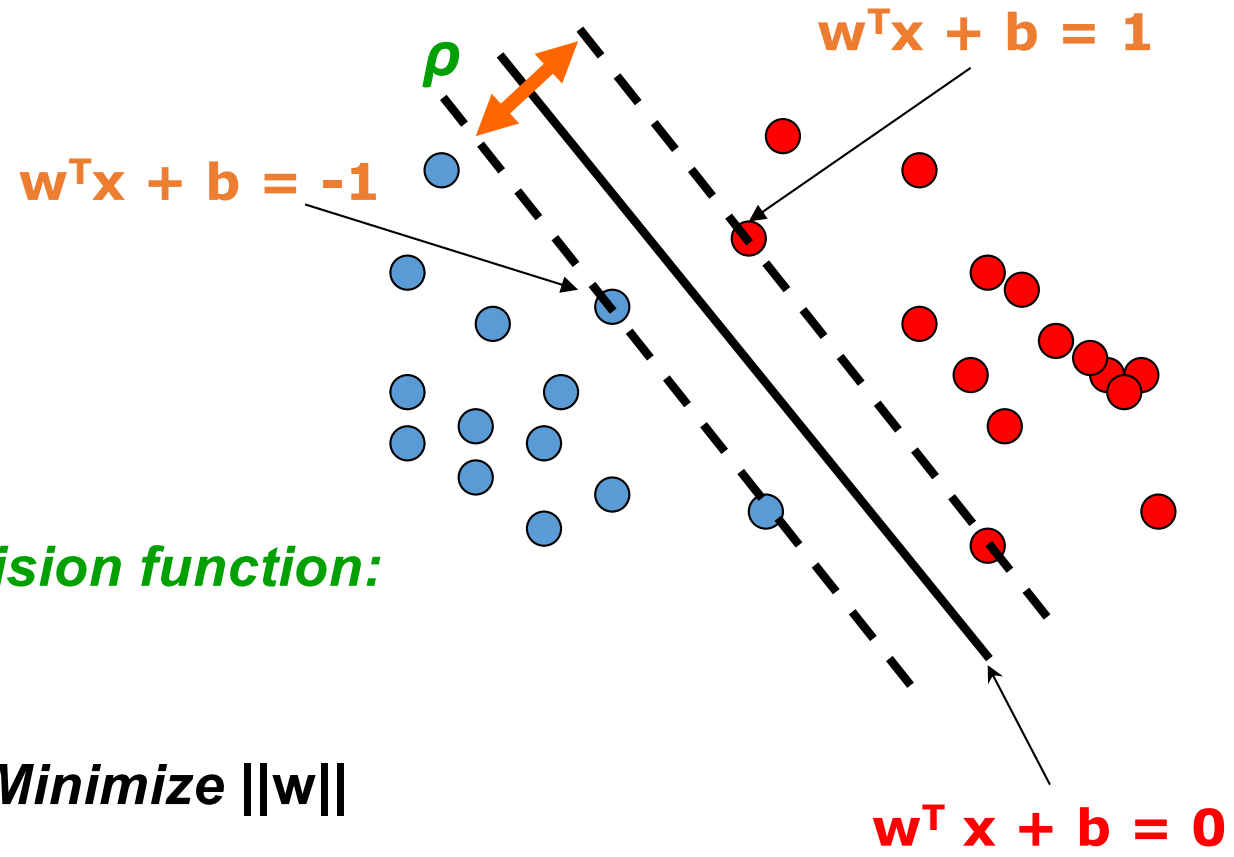
Linear Support Vector Machine

Decision Function

$\mathbf{w}^T \mathbf{x} + b$

Hyperplane

$\mathbf{w}^T \mathbf{x} + b = 0$



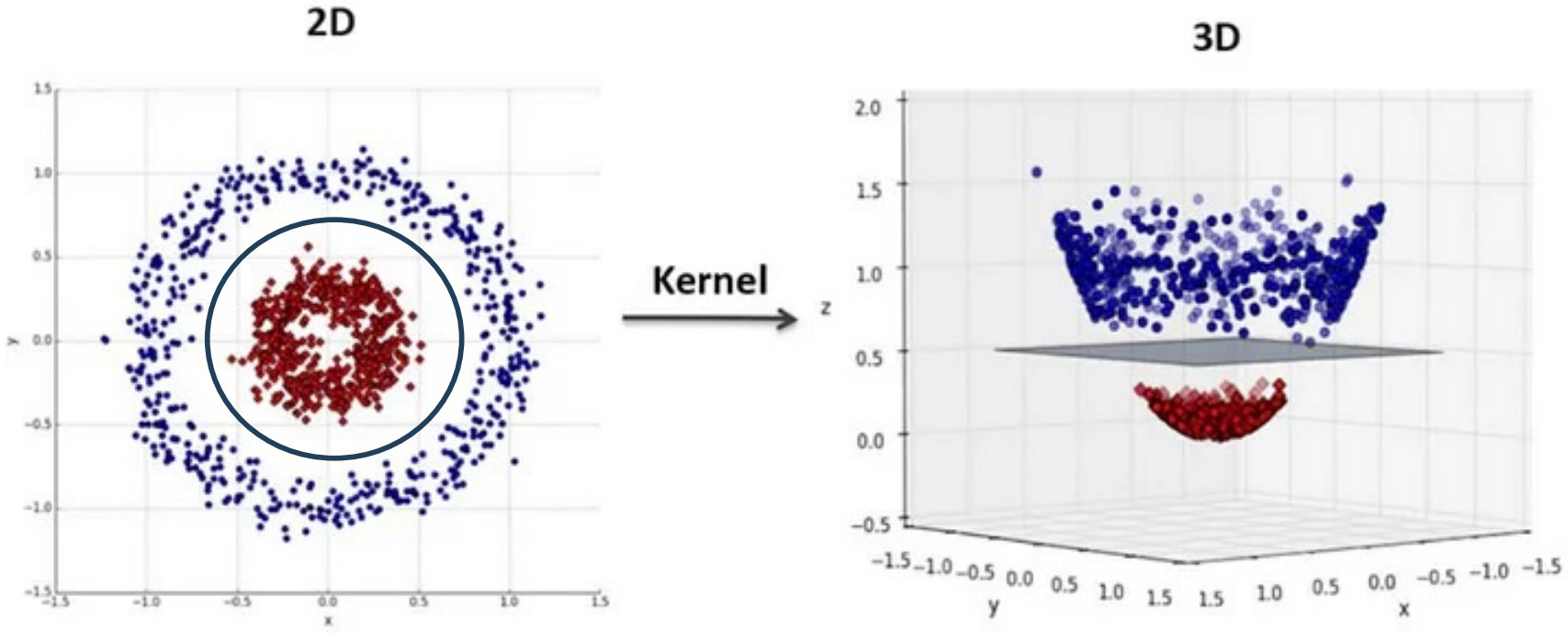
Given constraints to decision function:

$\rho = 2/\|\mathbf{w}\|_2$

Maximize the margin *ρ*, *Minimize* ||w||

Non-linear SVM

Kernels functions: Mapping the original feature space to some higher-dimensional feature space where the training set is separable



You will try out the rbf kernel and a polynomial kernel in the hand's on exercise.

Image Source: doi: 10.1002/9781118705810.ch10.

Support Vector Machines

- Pros
  - Performs well in high dimensional spaces
    - # features > # training points
  - Can find non linear relationships by using kernel trick
- Cons
  - Struggles with noisy data
  - Not as interpretable as logistic regression
    - No probabilities
    - No slopes

Naïve Bayes Classifier

Naïve Bayes Classifier

- Up until this point we have mostly made predictions by computing distances
  - KNN – distance to nearest neighbors
  - Linear Regression – Residuals
  - SVM – maximum margin
- Naïve Bayes is a **probabilistic model**
  - Similar to Logistic Regression, Naïve Bayes estimates probabilities of outputs

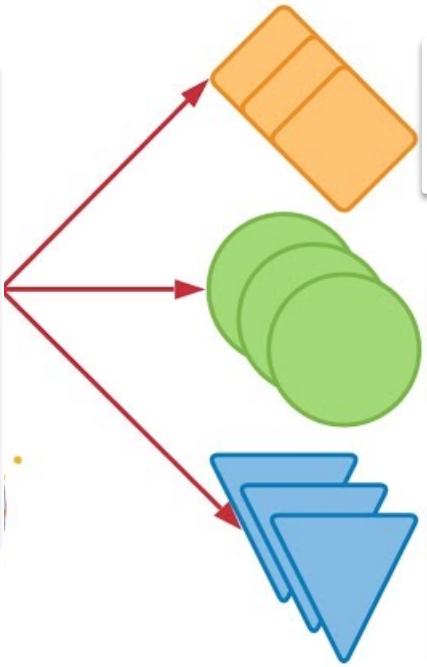
Naïve Bayes Classifier

Estimate probabilities of class given training data

Labelled Data

Bayes Theorem

$$P(C|D) = \frac{P(D|C)P(C)}{P(D)}$$



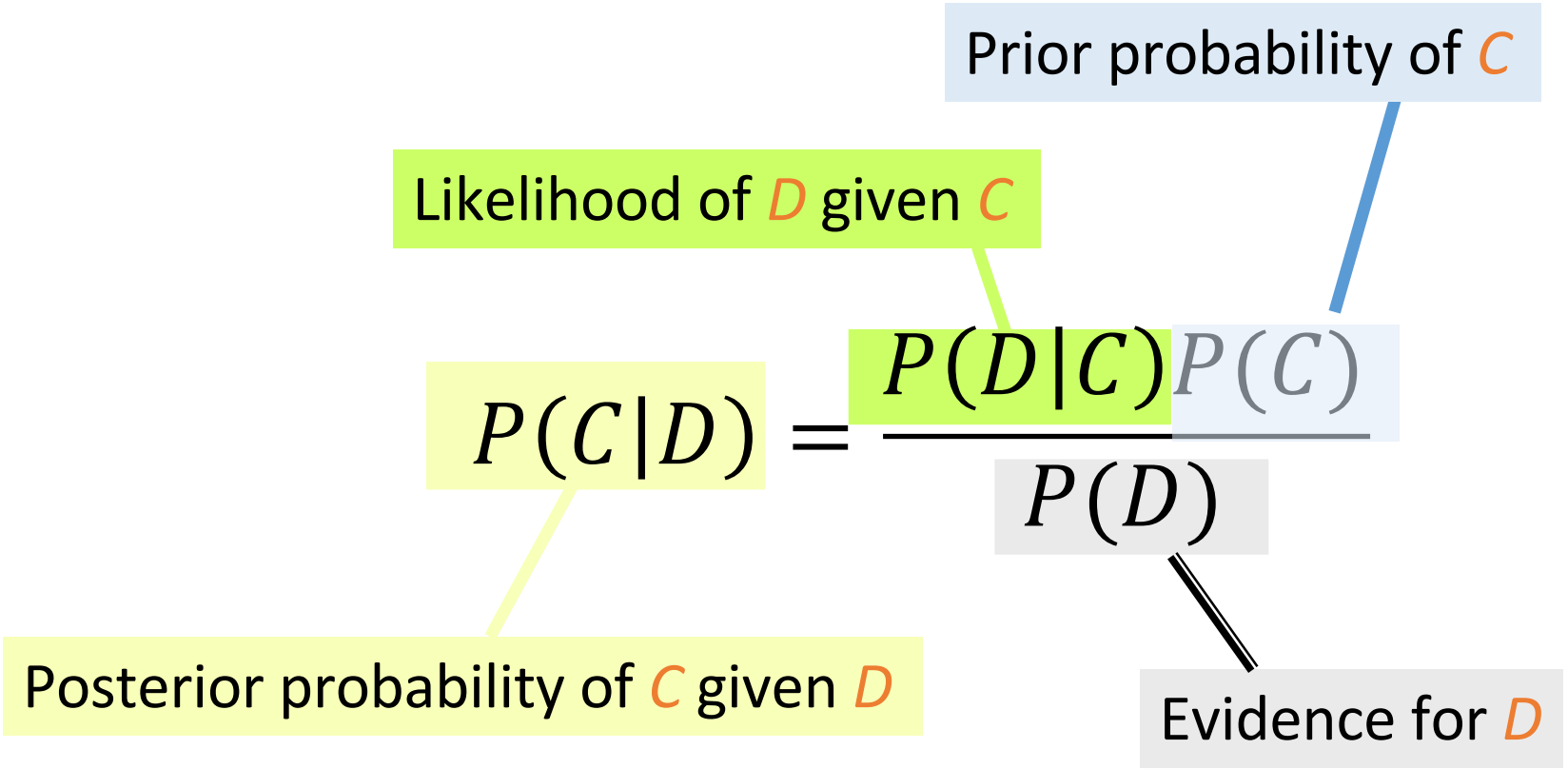
P(Orange | Data)

P(Green| Data)

P(Blue| Data)

Bayes Theorem for Inference

Let's use C (Class) and D (Data) instead of A, B



Bayes Theorem: update probability of class based on the data

Example Calculation

Refund	Marital Status	Evade
Yes	Single	No
No	Married	No
No	Single	No
Yes	Married	No
No	Divorced	Yes
No	Married	No
Yes	Divorced	No
No	Single	Yes
No	Married	No
No	Single	Yes

Compute P(C|D) where

Classes (C):

1. Yes – Evade Taxes
2. No – Do not Evade

How to Estimate Probabilities from Data?

Refund	Marital Status	Evade
Yes	Single	No
No	Married	No
No	Single	No
Yes	Married	No
No	Divorced	Yes
No	Married	No
Yes	Divorced	No
No	Single	Yes
No	Married	No
No	Single	Yes

$$P(C|D) = \frac{P(D|C)P(C)}{P(D)}$$

We do not need to compute this.

How to Estimate Probabilities from Data?

Refund	Marital Status	Evade
Yes	Single	No
No	Married	No
No	Single	No
Yes	Married	No
No	Divorced	Yes
No	Married	No
Yes	Divorced	No
No	Single	Yes
No	Married	No
No	Single	Yes

$$P(C|D) = \frac{P(D|C)P(C)}{P(D)}$$

- $P(C) = N_c/N$
- N_C is total number of data points of class C
- N is total number of data points

e.g., P(No) = 7/10,
P(Yes) = 3/10

How to Estimate Probabilities from Data?

Refund	Marital Status	Evade
Yes	Single	No
No	Married	No
No	Single	No
Yes	Married	No
No	Divorced	Yes
No	Married	No
Yes	Divorced	No
No	Single	Yes
No	Married	No
No	Single	Yes

$$P(C|D) = \frac{P(D|C)P(C)}{P(D)}$$

- **Assumption 1**
- Assumes Features are independent
- This allows us to treat each column independently and multiply computed probabilities

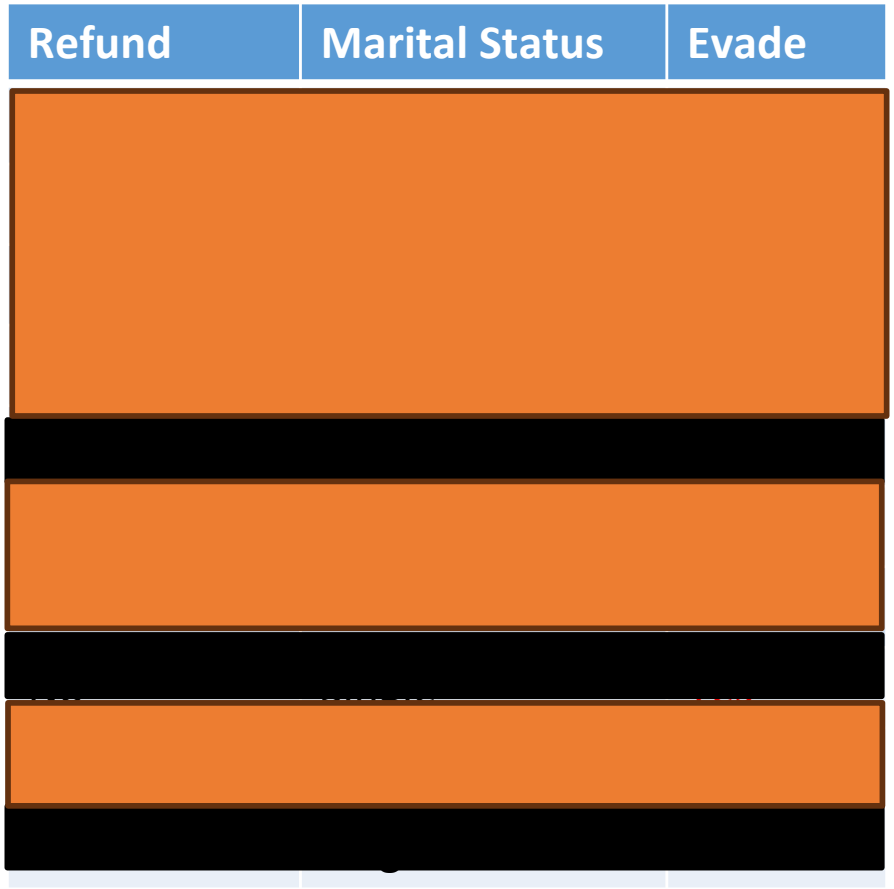
Given a Test Record:

$$X = (Refund = No, Married)$$

Compute:

$$P(D|No) = P(Refund = No|No)P(Married|No)$$

How to Estimate Probabilities from Data?



$$P(C|D) = \frac{P(D|C)P(C)}{P(D)}$$

- Example of computing Likelihood
- P(Status=Married|Evade=No)
  - only consider training data where class is no
  - compute the percent of data points where marital status = Married
- P(Status=Married|No) = 4/7
- P(Refund=Yes|Yes)=0

How to Estimate Probabilities from Data?

Refund	Marital Status	Evade
Yes	Single	No
No	Married	No
No	Single	No
Yes	Married	No
No	Divorced	Yes
No	Married	No
Yes	Divorced	No
No	Single	Yes
No	Married	No
No	Single	Yes

$$P(C|D) = \frac{P(D|C)P(C)}{P(D)}$$

- **Assumption 2:**
- Assume multinomial probability distribution for each feature
- There are different types of Naïve Bayes depending on the assumed underlying distribution per feature
- Multinomial, Gaussian, etc.

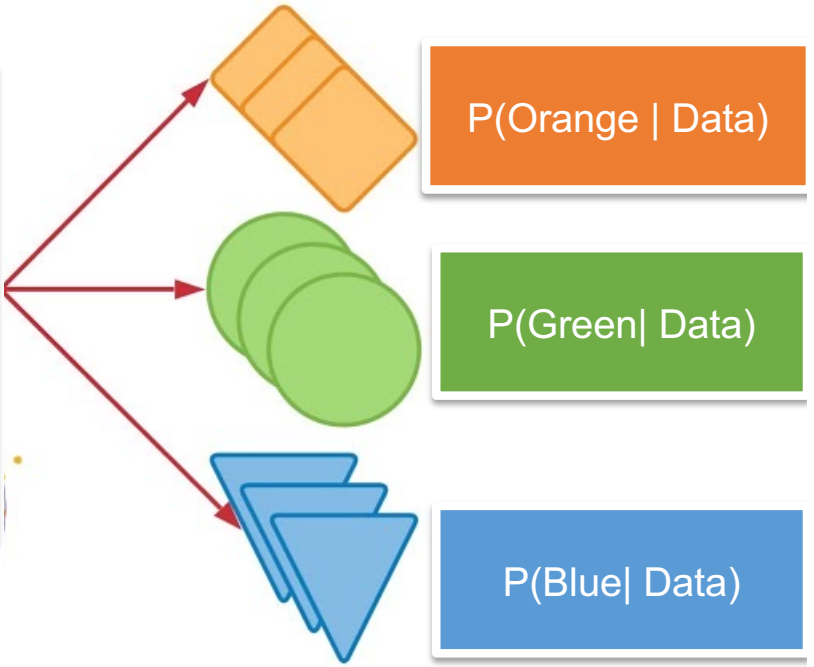
Naïve Bayes Classifier

Estimate probabilities of class given training data

Labelled Data

Bayes Theorem

$$P(C|D) = \frac{P(D|C)P(C)}{P(D)}$$



Highest posterior computed is the predicted class

Naïve Bayes (Assumptions)

Why is Naïve Bayes Naïve?

- Assumes features independent from one another
- Assumes variables come from a specific distribution

Despite assumptions, naïve bayes can still make good predictions in some use cases and is efficient at making predictions

- Text classification

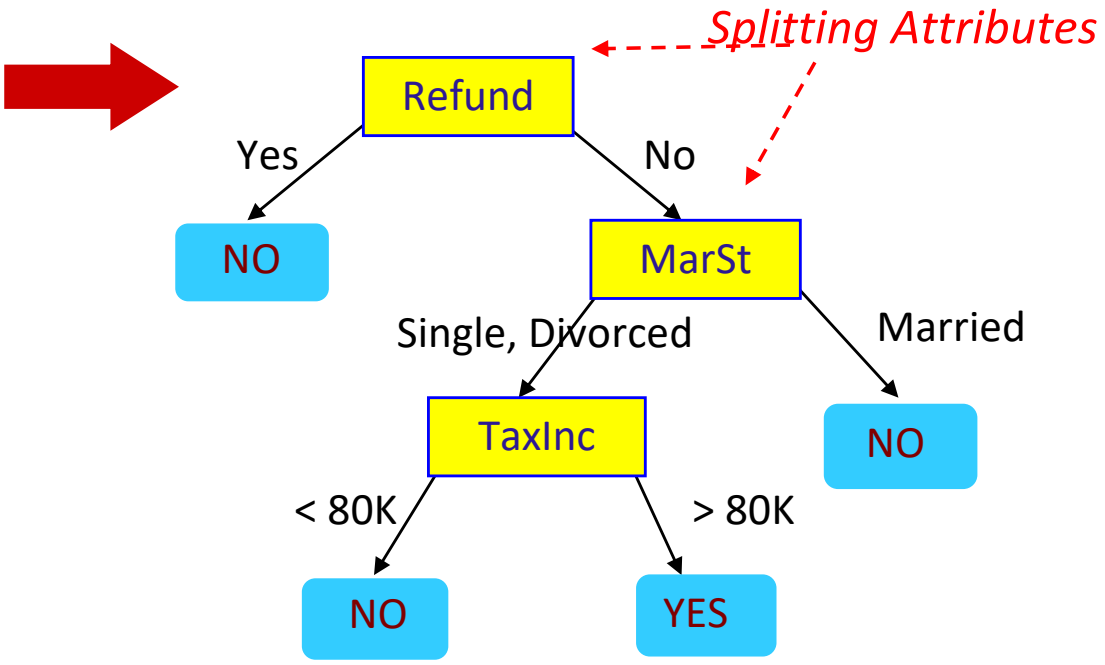
Tree-based Classifier

Decision Tree

- A decision tree is a (binary) tree with each branch having a left and right node.
- Each branch node specifies a feature to split along with information on which values to pass to the next node or leaf.
- Leaf nodes contain the predicted class label and are terminal.

	categorical	categorical	continuous	class
Tid	**Refund**	**Marital Status**	**Taxable Income**	**Cheat**
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Training Data



Model: Decision Tree

Decision Trees

Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Class 1: Don't cheat (default class)
Class 2: Cheat

When building a decision tree:

1. Considering all possible binary splits based on a single feature
2. Calculate the information gain for all possible splits
3. Pick the best split
4. Repeat with children nodes until pure node is created or some other stopping criteria

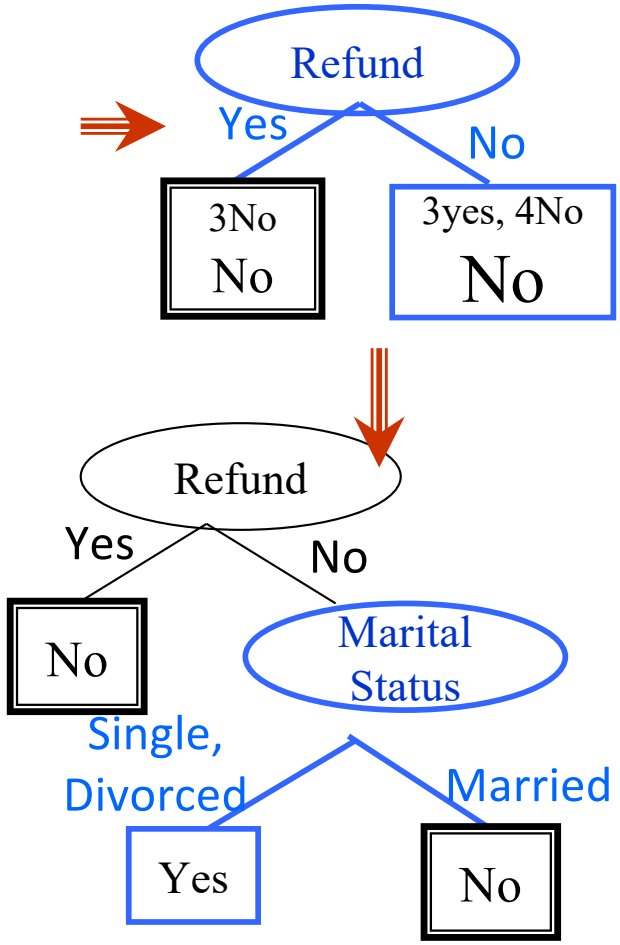
Decision Trees

Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Class 1: Don't cheat (default class)
Class 2: Cheat

When building a decision tree:

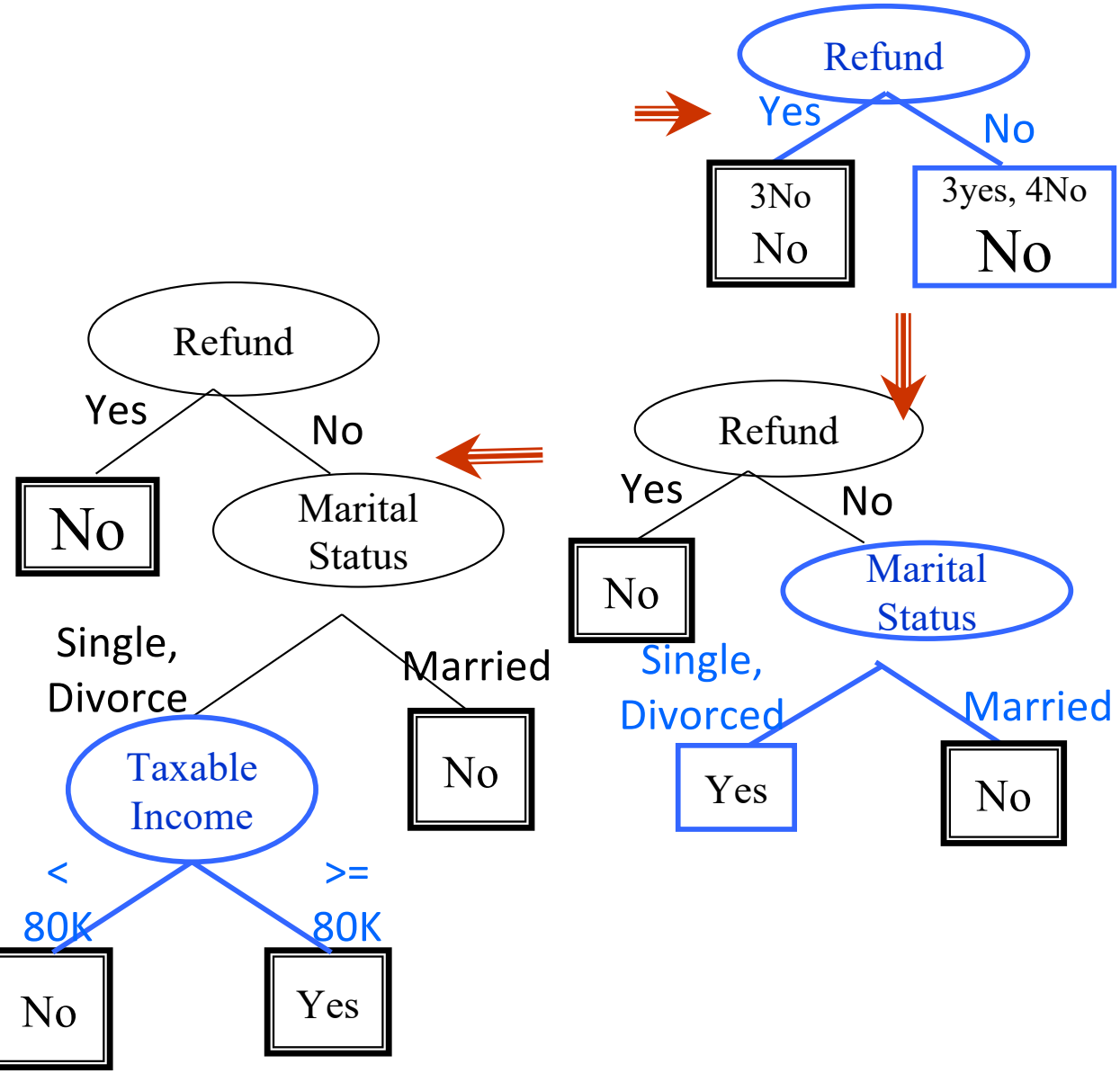
1. Considering all possible binary splits based on a single feature
2. Calculate the information gain for all possible splits
3. Pick the best split
4. Repeat with children nodes until pure node is created or some other stopping criteria



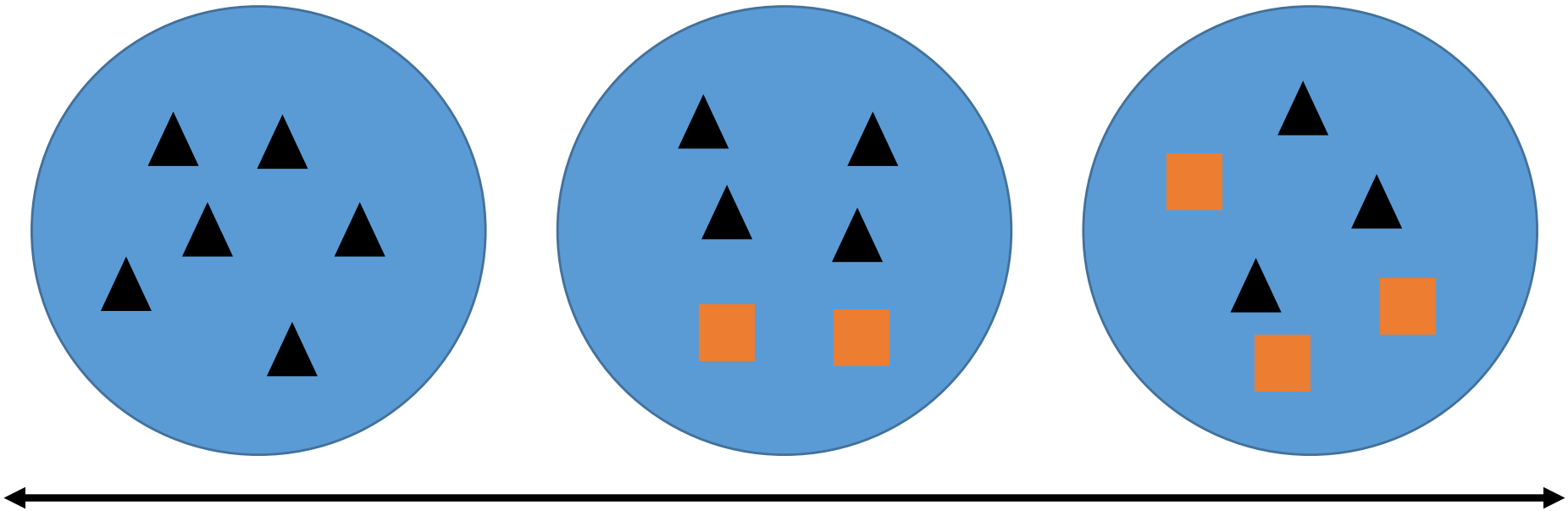
Decision Trees

Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Class 1: Don't cheat (default class)
Class 2: Cheat



Which split is the best? -- measure impurity



High purity

Low purity

$P(black) = 1$

$P(orange) = 0$

$P(black) = 0.5$

$P(orange) = 0.5$

Measuring Node Impurity

- *Let $p(j \mid t)$* is the relative frequency of class j at node t).
  - Entropy

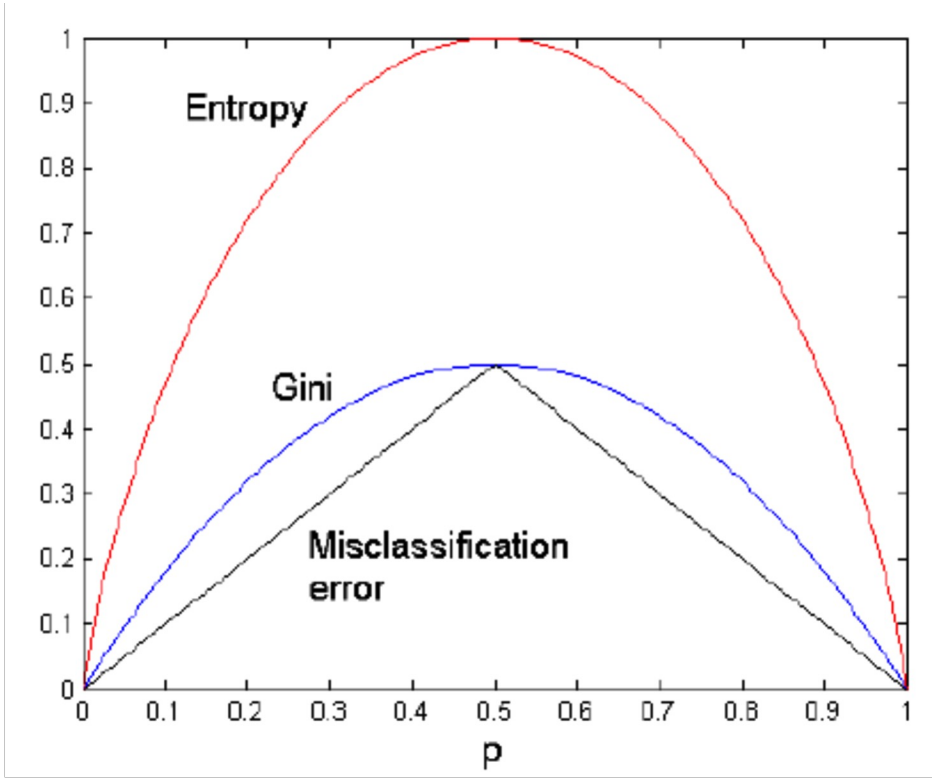
$$Entropy(t) = -\sum_j p(j \mid t) \log p(j \mid t)$$

  - Gini Index

$$GINI(t) = 1 - \sum_j [p(j \mid t)]^2$$

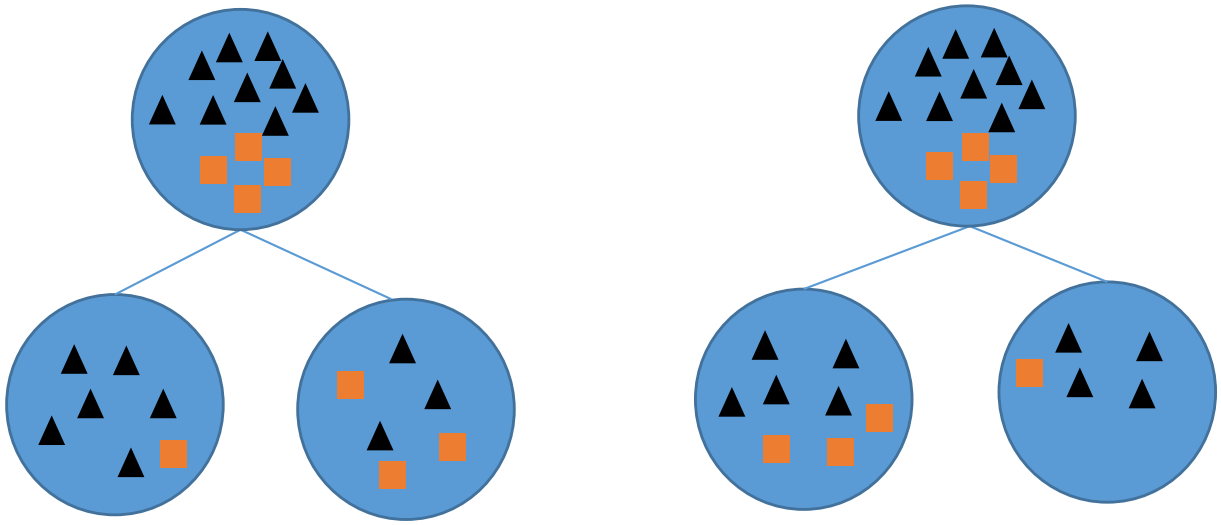
  - Misclassification Error

$$Error(t) = 1 - \max_i P(i \mid t)$$



Which split is the best? -- information gain

Which split is better?



1. Compute purity of all nodes
2. Compare the purity of the parent node to the weighted average of the children

$$IG(P,C) = H(P) - \frac{|C_1|}{|P|}H(C_1) - \frac{|C_2|}{|P|}H(C_2)$$

Where P is the parent node
C_i is one of the two children nodes
H is the impurity metric
|| refers to the data points in that node

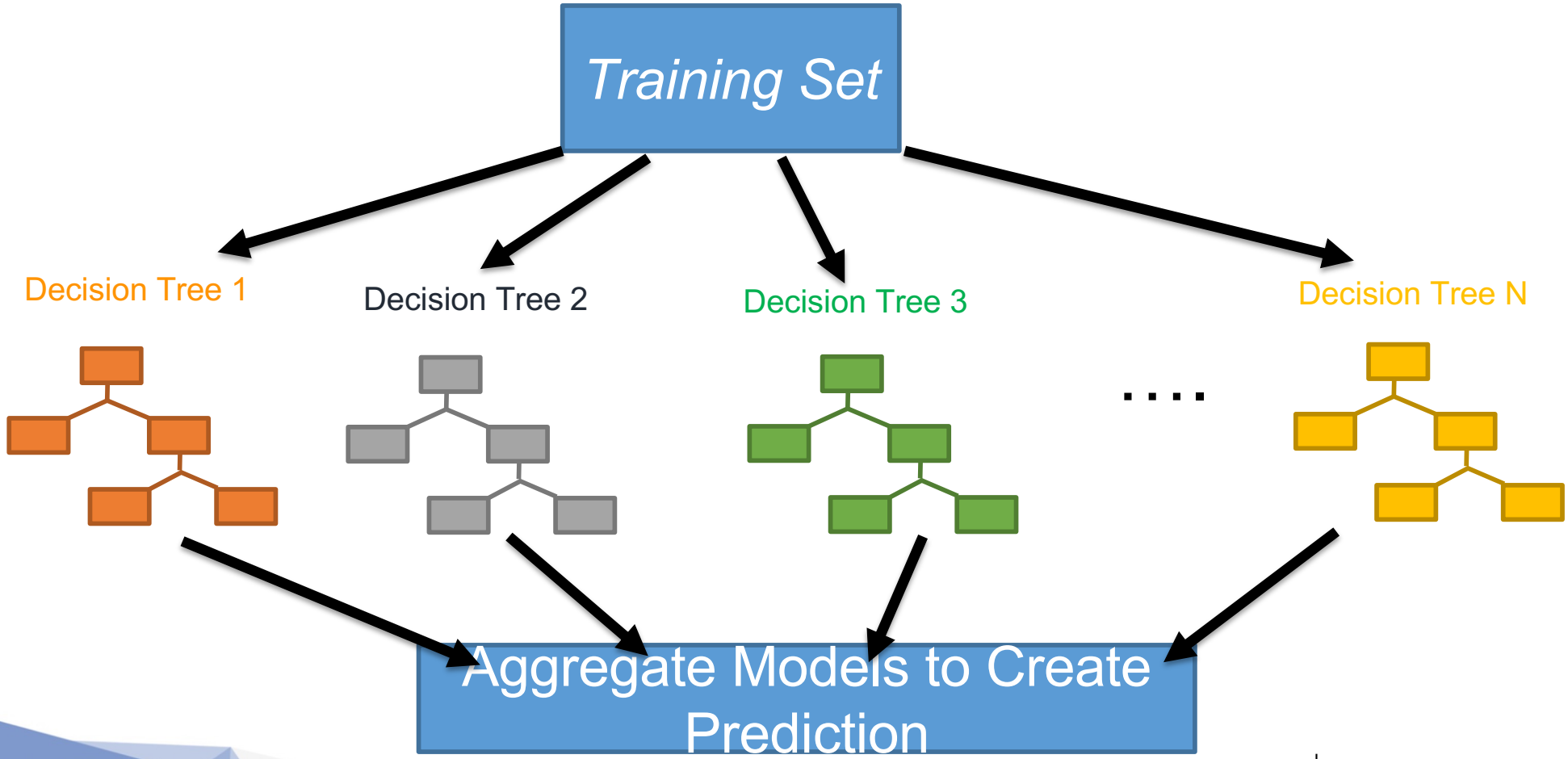
Decision Tree Based Classification

- Advantages:
  - Inexpensive to construct
  - Extremely fast at classifying unknown records
  - Easy to interpret for small-sized trees
  - Accuracy is comparable to other classification techniques for many simple data sets

- Less effective when
  - High background noise.
  - Large scale data
  - Data with high dimension

Ensemble Methods

procedure that combines the outputs of many “weak” models to produce a ”strong” model



Random Forest

A collection of decorrelated decision trees

Building stage

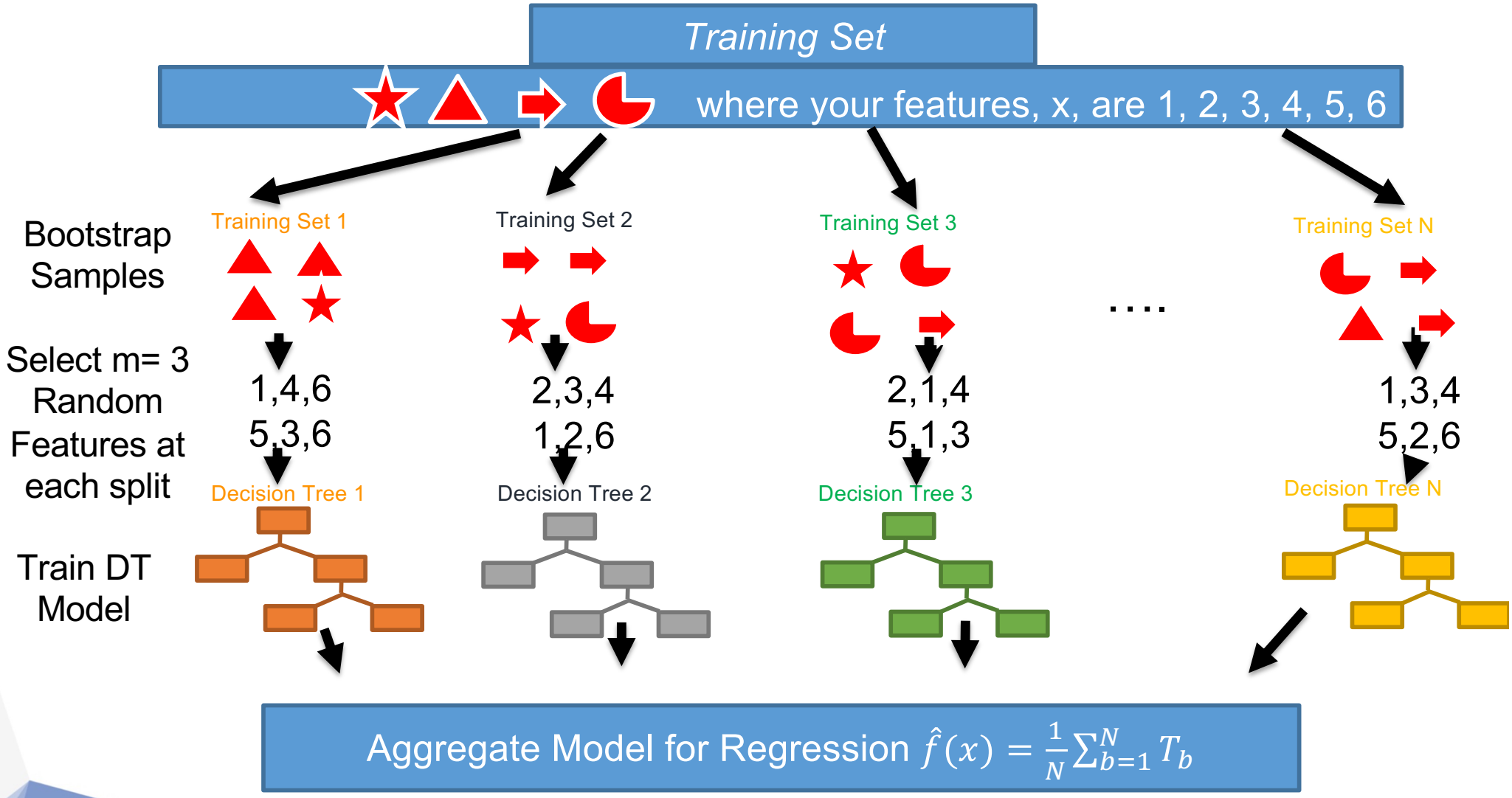
- Randomly sample data with replacement
- Only a subset of attributes are chosen to make decision at each node of the tree
- Select best split from selected subset of attributes.

Classifying

All trees are used to make a decision.

The final class label is determined statistically e.g. majority rule.

Random Forest



Feature Importance

- When building random forest models we can get a since for what features were most important to making predictions
- For example
  - Gini importance
    - Total decrease in node impurity for a particular feature weighted by the probability of reaching a particular node
  - Permutation importance
    - Quanitfy how much the performance of your model decrease if we randomly permute a feature.
- While the ability to quantify feature importance is great, it is not as interpretable as for example slopes in linear regression.

Random Forest

Advantage

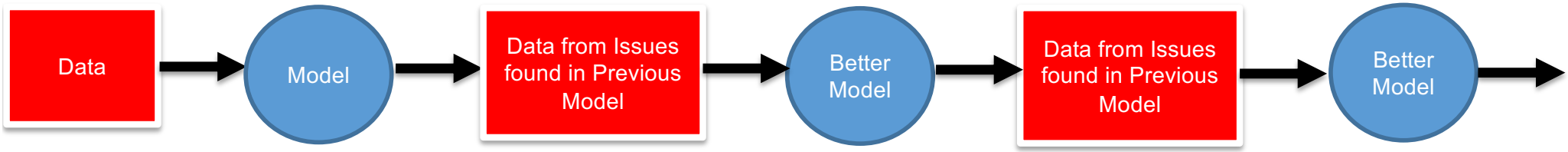
- Accurate
- Good for high dimensional data
- Reduce the need of feature detection and selection.
- Give estimate on important features.

Disadvantage

- More computational requirement
- Less effective with noisy training data.
- The result is an statistical ensemble and might be hard to interpret.

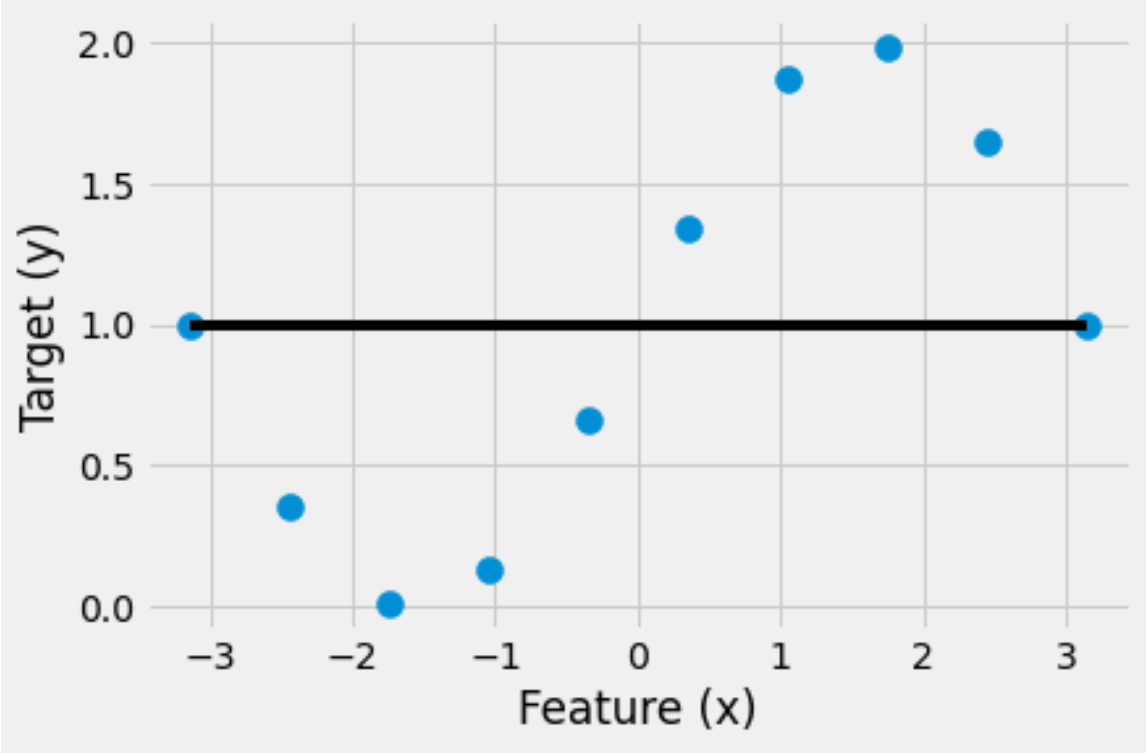
Gradient Boosting

Different Approach: sequentially add weak models to repeatedly modified versions of the data to improve model



Gradient Boosting

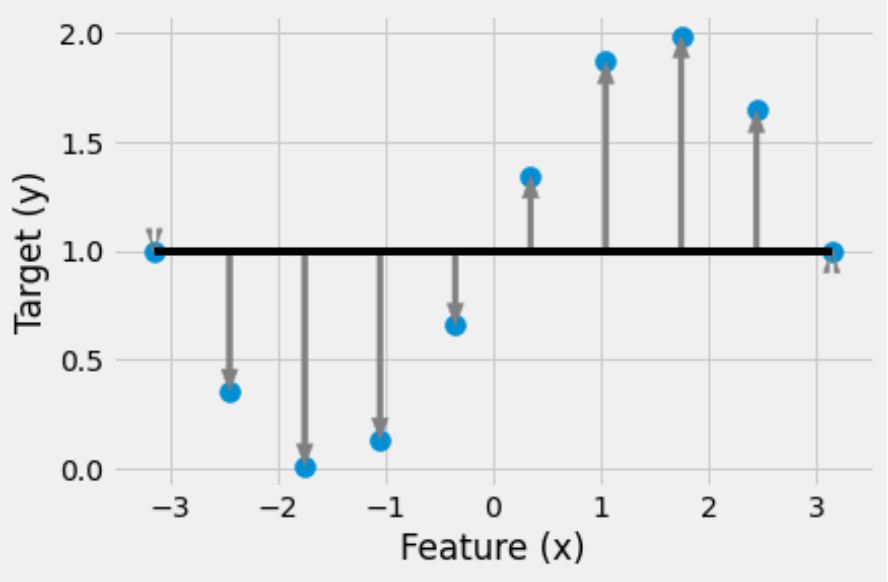
Start by creating a weak model; e.g. mean regressor



$$f_0(x) = \bar{y}$$

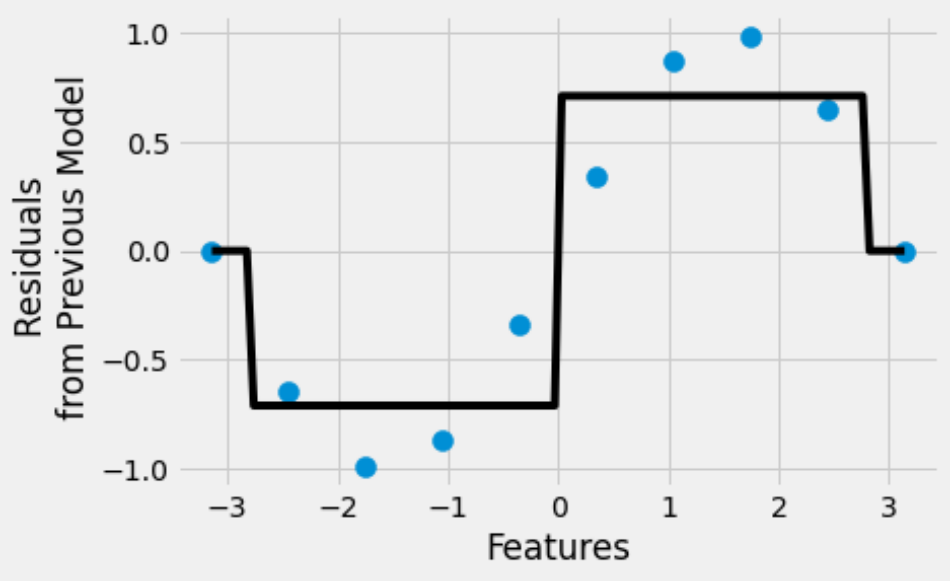
Gradient Boosting

Quantify errors by computing residuals



$y_{true} - f_0(x)$

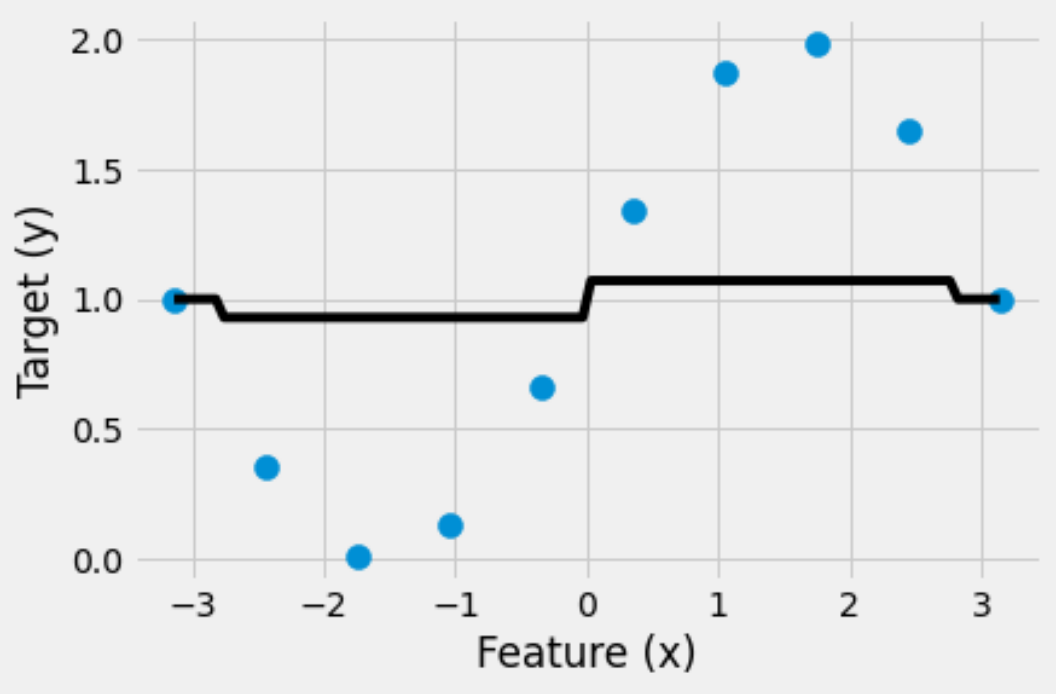
Build a decision tree to learn the residual



$h_1(x)$

Gradient Boosting

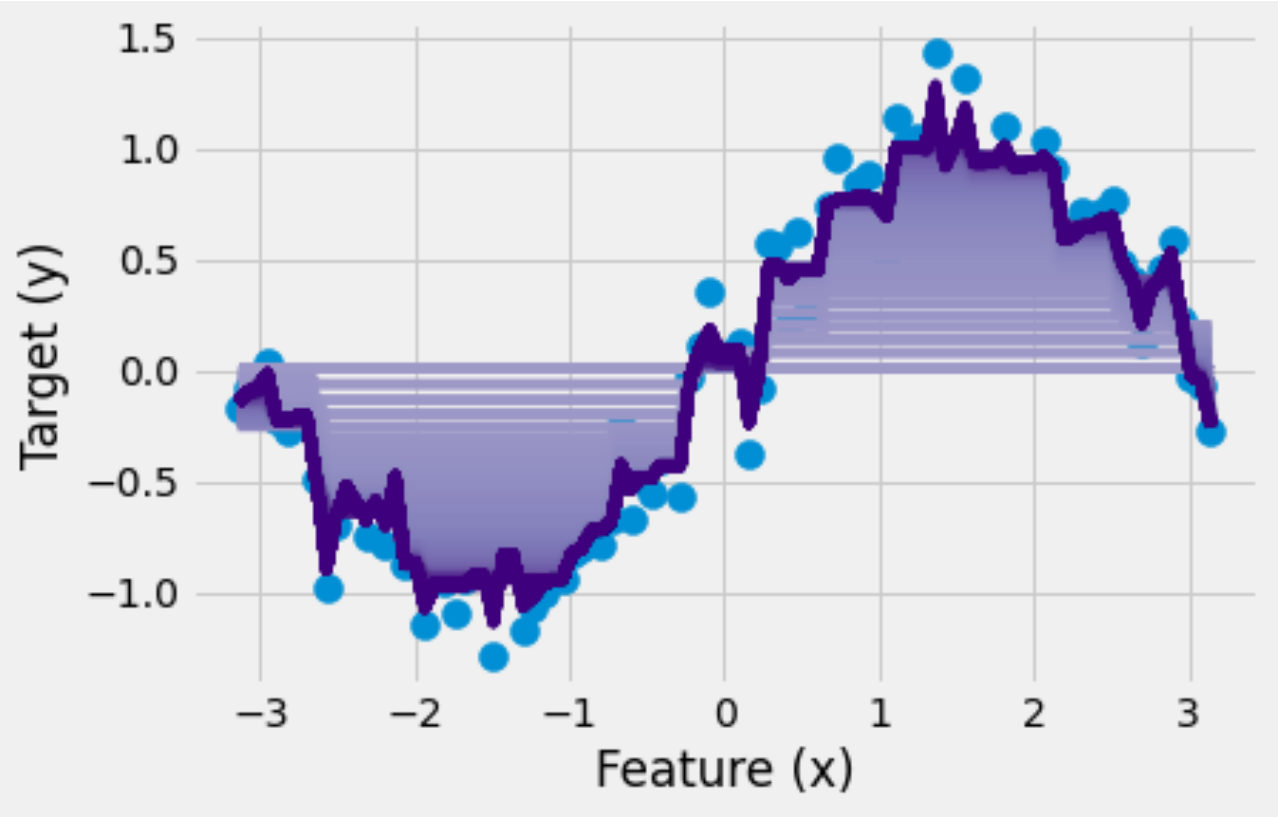
Add decision tree approximating residuals to mean regressor



$$f_1(x) = f_0(x) + \lambda h_1(x)$$

Gradient Boosting

Repeat



Gradient Boosting vs Random Forest

- Both are powerful models
- Random Forest tends to get better out of the box performance and requires less hyperparameter tuning
- If tuned correctly, Gradient Boosting tends to match or out perform random forest.

Summary

- Supervised learning methods model the connections between training data with their known outcome value/label.

- By distance
  - E.g. knn, regression, SVM
  - Predictions are based on position in the space.
  - Data can be projected into new spaces

- By probability distribution
  - E.g. Naïve Bayes Classifier
  - A statistical distribution is “learned”.
  - Predictions are based on likelihood of observation.

- By feature space
  - E.g. decision trees, Random forest.
  - Data are divided according to their feature spaces.
  - Predictions are based on group of features that shared.

Summary

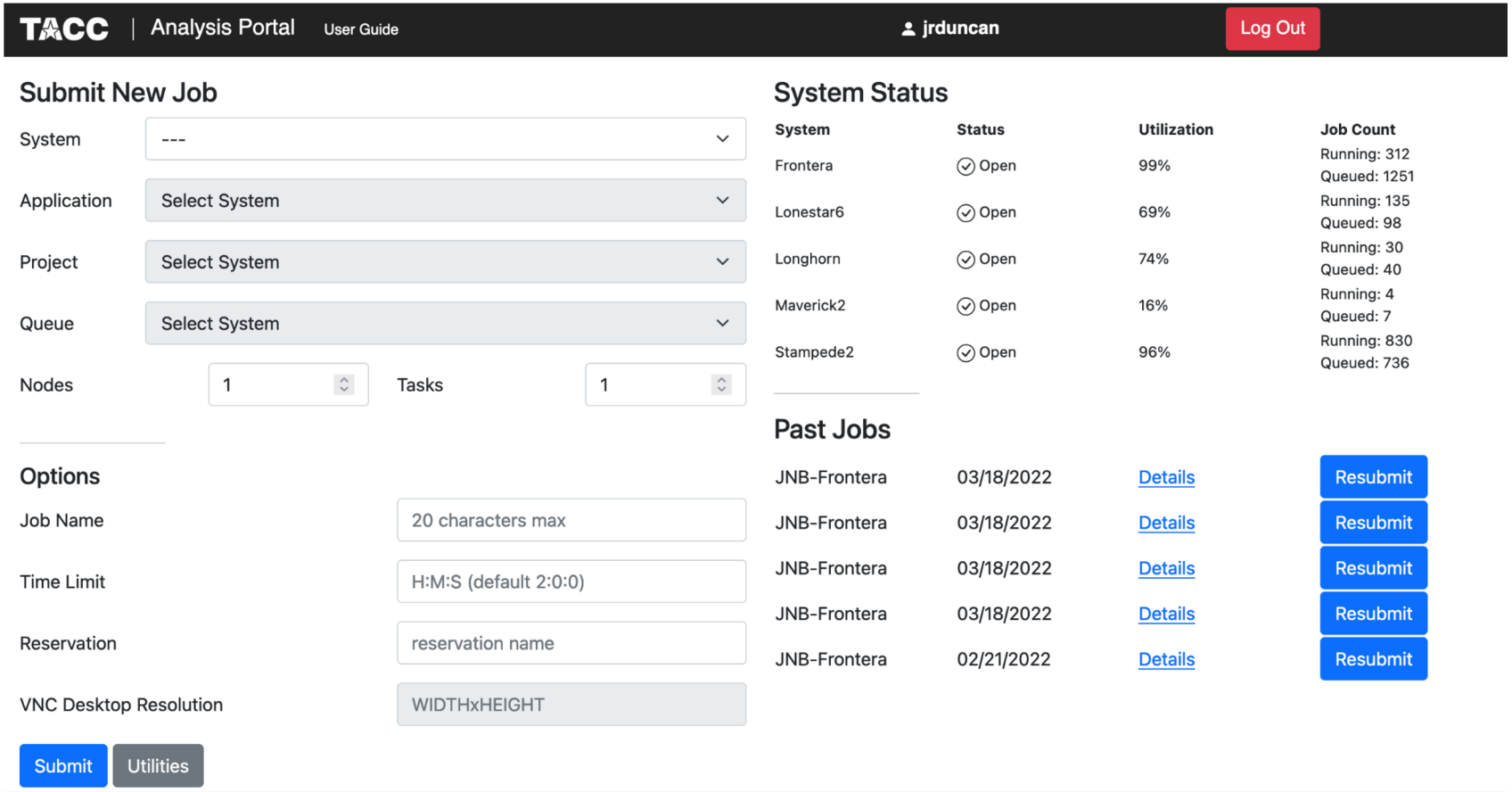
- Machine Learning (ML) Basics
  - Define ML, Categories of ML, ML Workflow
- Common Supervised Learning Techniques
  - K Nearest Neighbors, Linear Regression , Logistic Regression, Support vector machines , Probabilistic Models , Decision Tree Based Models
- Evaluation Metrics
  - Classification: Confusion Matrix, Precision, Recall, ROC curve
  - Regression: MSE, MAE, R^2
- Preparing Data for Training
  - Standardizing Data
  - Feature Engineering / Selection

Machine Learning on HPC: Supervised Learning Excercises and Case Study

Jupyter notebook

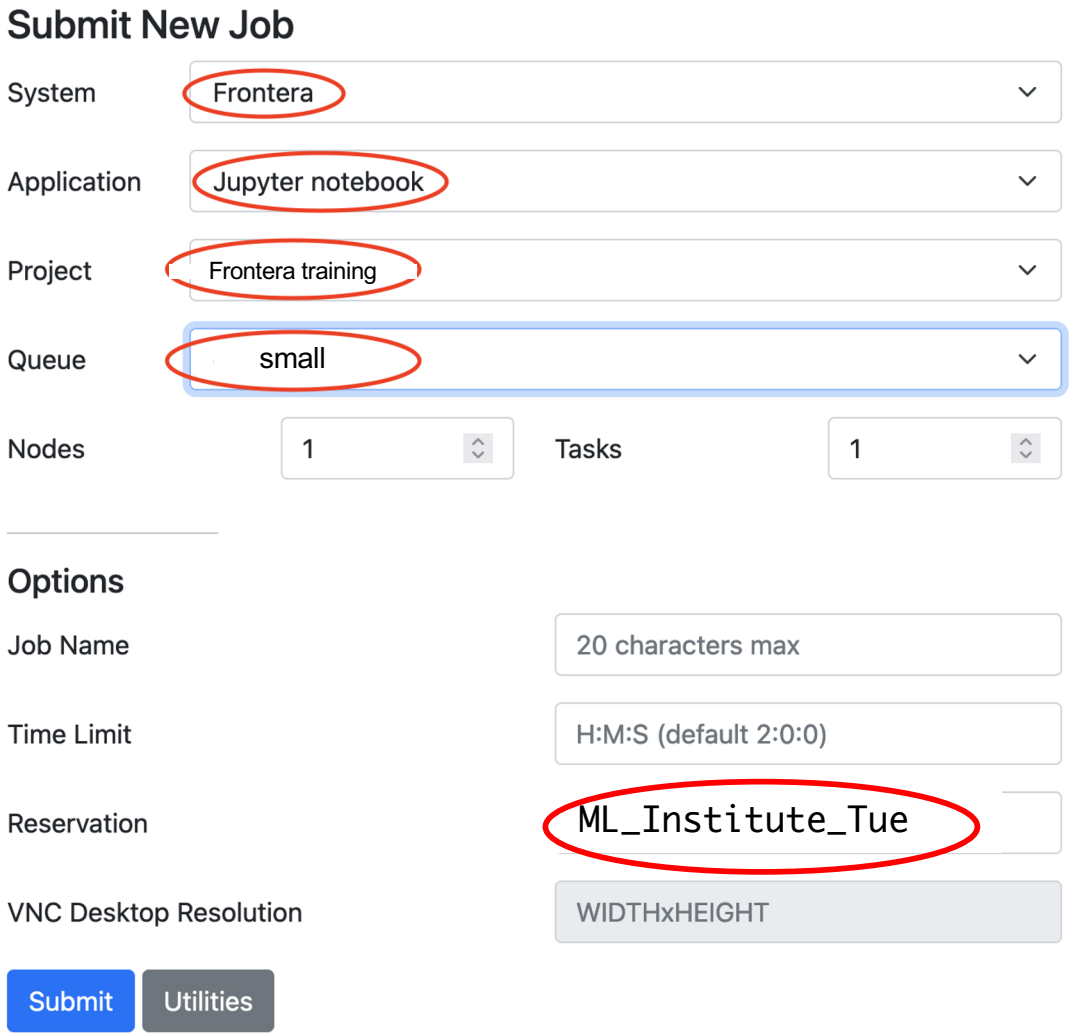
- TACC Analysis Portal:

  Go to https://vis.tacc.utexas.edu; Login with your training account credentials

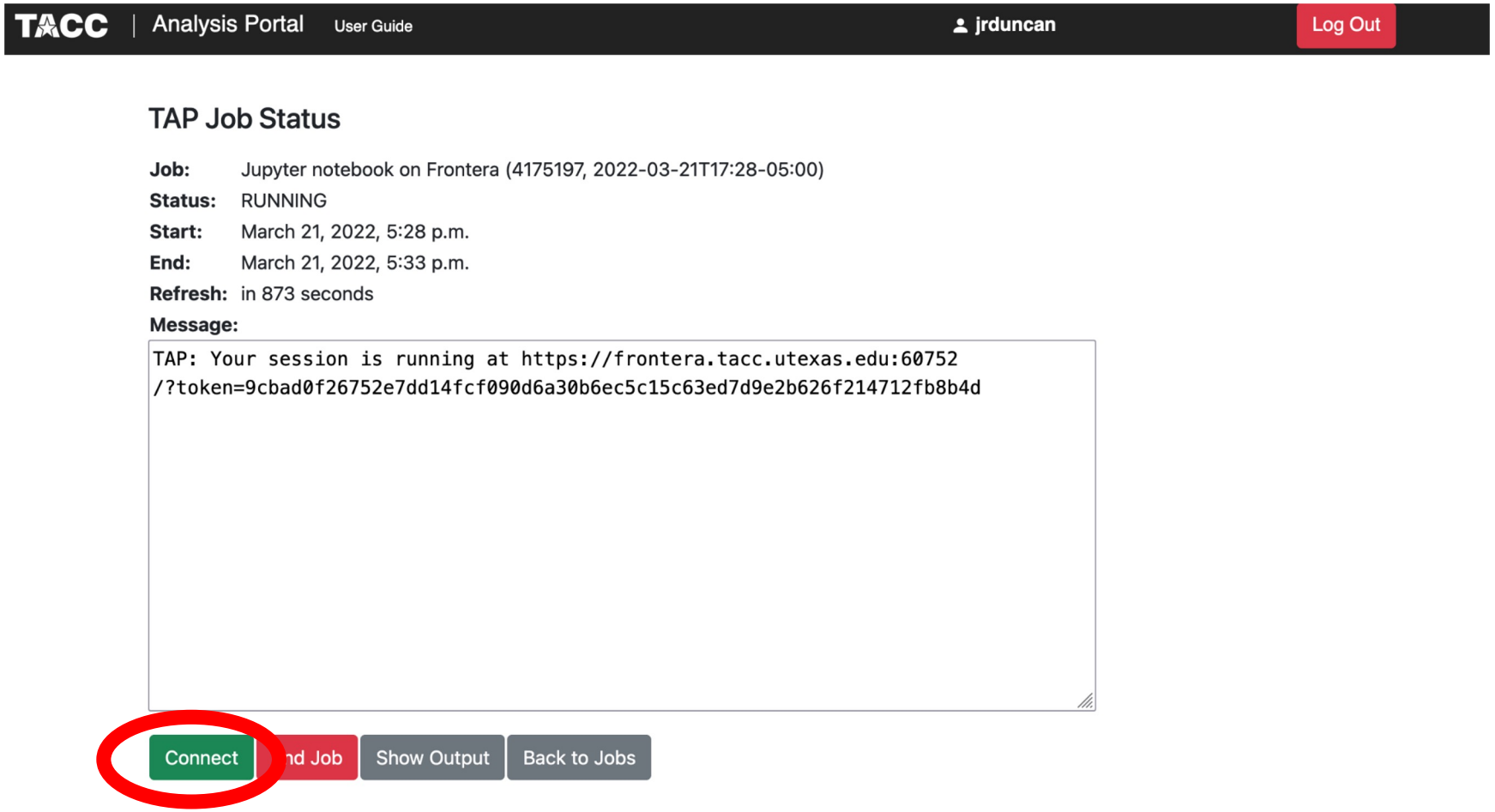


Jupyter notebook

- TACC Visualization Portal:
  Select the following options



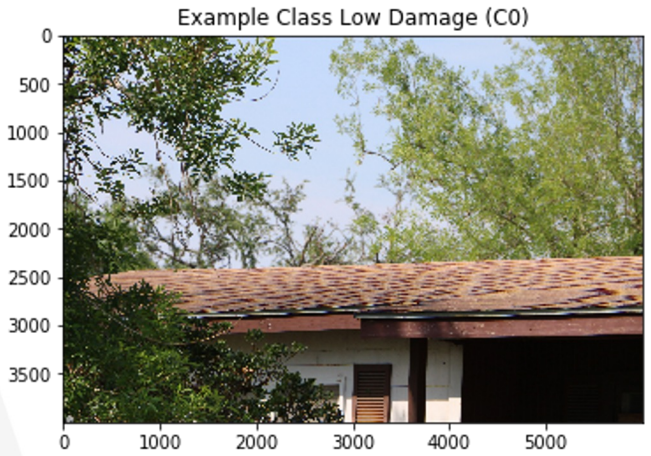
Jupyter notebook

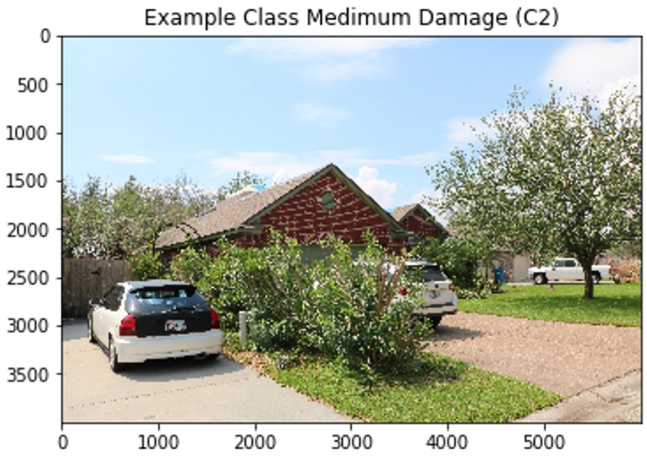


Exercise

DESIGNSAFE-CI
A NATURAL HAZARDS ENGINEERING COMMUNITY

Image Classification with Hurricane Harvey Dataset





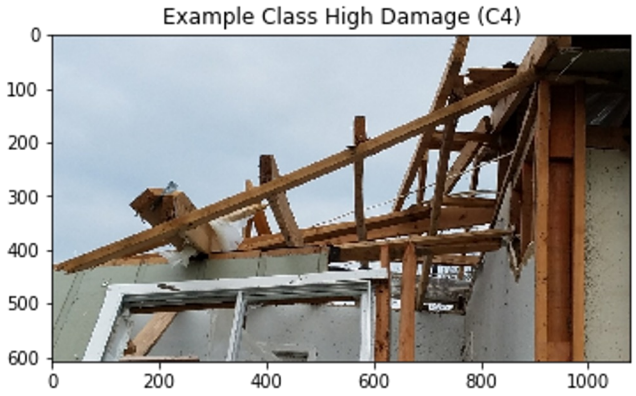
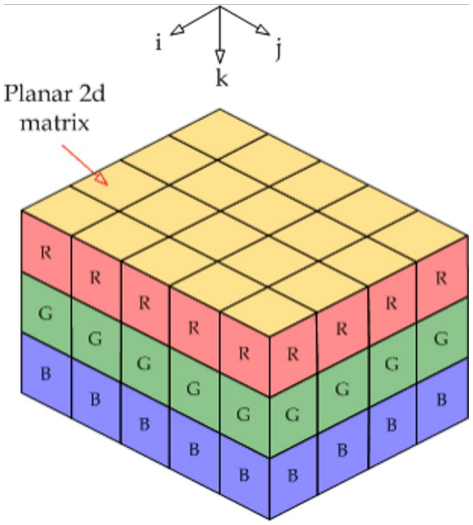


Image Processing

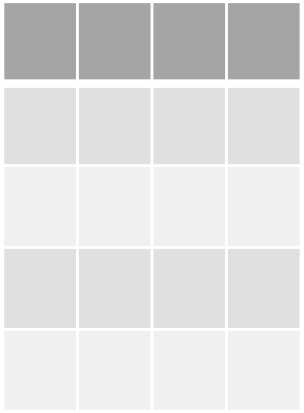
- Process of manipulating images
- Common image processing tasks:
  - Gray-scaling images
  - Resizing images
  - Reshaping data
- Image processing is used to for feature selection in supervised learning

Image Processing

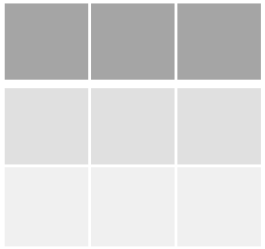
Color Image



Grayscale



Resize



Vectorize

Today:
Supervised Learning

Image processing

Vectorized Data

									low
			pixels						medium
									high

Train Model

Evaluate Model

Building and Evaluating Classification Models

- Take vectorized images from pipeline and use them to build supervised learning models
  - Two different datasets:
    - City streets versus Forest
    - Design Safe dataset
  - Decision tree, Logistic Regression, Naïve Bayes, Support Vector Machine, Random Forest
- Model Evaluation
  - Ok, so I built some models, now what?
  - Showcases taking a deeper dive into the models you built so that you can learn how to improve performance

Instructions

In this exercise you will run two notebooks which build and evaluate image classification models

- Run the **city_vs_forest_classifier.ipynb**

  This exercise will build models to distinguish between images of city landscapes vs forest landscapes

- Run the **designsafe_classifier.ipynb**

  This exercise will build models using the DesignSafe Dataset you were introduced to yesterday. We will build classification models which predict whether the damage level is low, medium or high.