UNIVERSITY of WEST FLORIDA

Machine Learning

Module 3
Scikit-learn ML Classifiers

Brian Jalaian, Ph.D.
Associate Professor
Intelligent Systems & Robotics Department

Overview of Topics in Module 3

UNIVERSITY of WEST FLORIDA

Introduction to Supervised Machine Learning

- Training Perceptron via Scikit-learn
- Logistic Regression and Conditional Probability
- Logistic Function and Cost Function
- Likelihood Function, Loss Function, and Regularization
- Training Logistic Regression with Scikit-learn

Support Vector Machines (SVM)

- Introducing Slack Variables
- Kernel Methods

Decision Trees and Random Forests

- Building a Decision Tree
- Advantages and Disadvantages of Decision Trees
- Building a Random Forest

K-Nearest Neighbors (KNN)

Choosing a classification algorithm

UNIVERSITY of WEST FLORIDA

- Requires practice & experience
- No free lunch theorem by David H. Walport (No single classifier works best across all possible scenarios)
- Recommended to compare the performance of a handful of different learning algorithms to select the best model for a particular problem.

5 Steps for Training Supervised Machine Learning

UNIVERSITY of WEST FLORIDA

1. **Selecting features and collecting labeled training examples:** This step involves identifying the relevant features of the data that will be used to make predictions. The features should be selected based on their relevance to the problem being solved, and the data should be labeled with the correct output for each input.
2. **Choosing a performance metric:** This step involves selecting a metric that will be used to evaluate the performance of the model. Common metrics for classification problems include accuracy, precision, recall, and F1 score.
3. **Choosing a learning algorithm and training a model:** This step involves selecting a machine learning algorithm that is appropriate for the problem being solved and the data being used. The model is then trained on the labeled training data.
4. **Evaluating the performance of a model:** This step involves evaluating the performance of the model on unseen data, using the performance metric chosen in step 2.
5. **Changing the settings of the algorithm and tuning the model:** This step involves adjusting the settings of the algorithm or changing the features or labeled data used to train the model in order to improve its performance. This process is known as model tuning or hyperparameter optimization.

Training Perceptron via Scikit-learn

UNIVERSITY of WEST FLORIDA

- Scikit-learn API: Provides a user-friendly and consistent interface with a highly optimized implementation of several classification algorithms
- Several function to pre-process data, fine-tune, and evaluate the models

Logistic Regression

UNIVERSITY of WEST FLORIDA

- Logistic regression is a widely used classification model that can be used to predict a binary outcome (e.g. success/failure, yes/no, etc.) based on one or more predictor variables.
- Unlike linear regression, which is used to predict a continuous outcome, logistic regression models the probability of the outcome being a certain value.
- Logistic regression is a linear model that is suitable for linearly separable classes, which means the decision boundary is a straight line.
- It is similar to the perceptron and Adaline models, but it is typically used for binary classification.
- It can be extended to handle multi-class classification problems through the use of multinomial logistic regression or the one-vs-rest (OvR) technique.
- Logistic Regression is widely used in industry and business, particularly in the field of marketing, finance, and healthcare.

Example Code:

UNIVERSITY of WEST FLORIDA

```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

# Load the data
X, y = ...

# Split the data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=1)

# Create and fit the model
logreg = LogisticRegression(solver='lbfgs', multi_class='auto')
logreg.fit(X_train, y_train)

# Make predictions on the test set
y_pred = logreg.predict(X_test)

# Evaluate the model
acc = accuracy_score(y_test, y_pred)
print(f'Accuracy: {acc:.3f}')
```

Logistic Regression & Conditional Probabilities - 1

UNIVERSITY of WEST FLORIDA

- Odds: in favor of a particular event
- $\frac{p}{1-p}$, where p stands for the probability of positive event
- $p = p(y = 1 \mid x)$, positive event class label $y = 1$ and the symptoms as feature x
- Logit function: logarithm of the odds

$$\text{Logit}(p) = \log \frac{p}{1-p}$$

Takes input value in the range of 0 to 1 and transforms them to entire real-number range

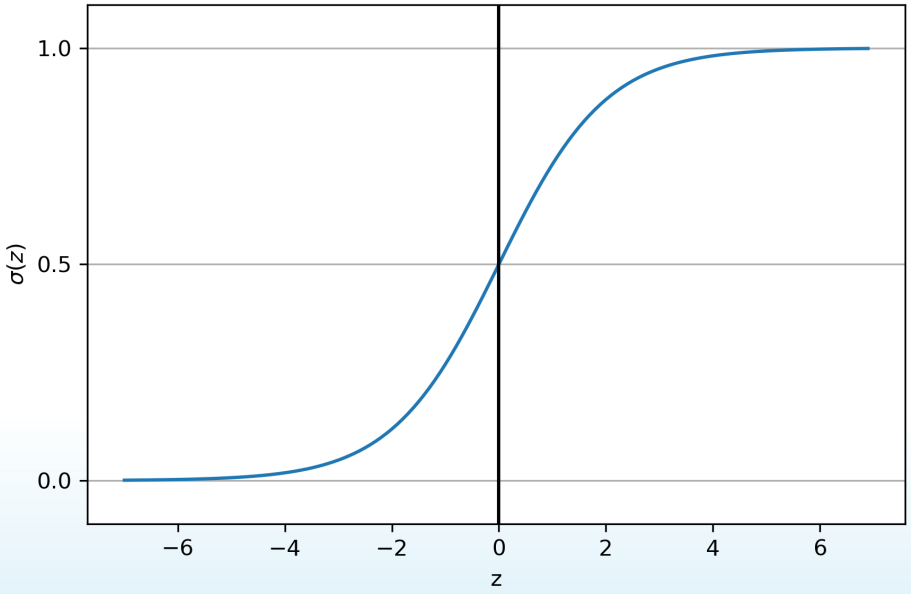
Logistic Regression & Conditional Probabilities - 2

- Under the logistic model, we assume there is a relationship between the weighted input (net input) and the logit:

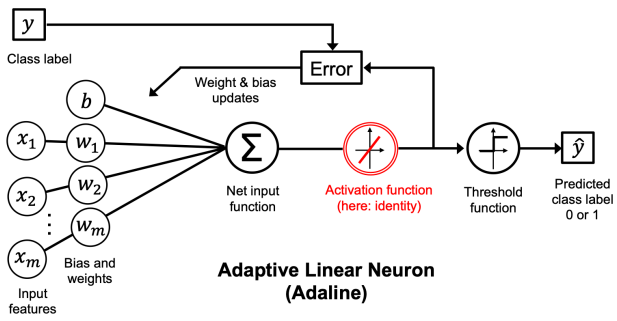
$$\text{logit}(p) = w_1 x_1 + \cdots + w_m x_m + b = \Sigma_{i=j} w_j x_j + b = \boldsymbol{w^T} + b$$

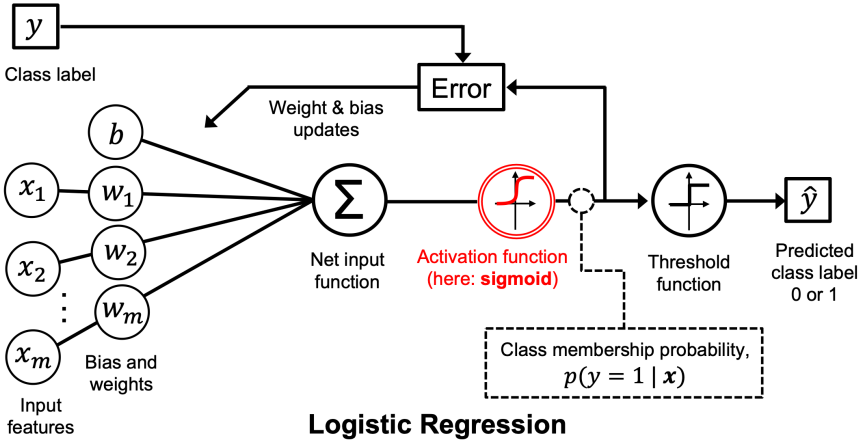
- Logit function map the probability p (class membership probability of an example given its feature) to a real-number range, we can use the inverse function to revert the real range to [0,1] range for the probability p
- The inverse of the logit function called the **logistic sigmoid function** or **sigmoid function**

$$\sigma(z) = \frac{1}{1 + e^{-z}} \qquad z = \boldsymbol{w^T} + b$$



Logistic regression compared to Adaline





- Top diagram : Adaline Linear Neuron
  - Illustrate an Adaline model with input features, weights, and a bias term, as well as the linear activation function.
  - Label the input features, weights, bias, and output (e.g., "x1", "x2", "w1", "w2", "b", "y").
- Bottom diagram : Logistic Regression
  - Illustrate a logistic regression model with input features, weights, and a bias term, as well as the sigmoid activation function.
  - Label the input features, weights, bias, and output (e.g., "x1", "x2", "w1", "w2", "b", "y").

The Logistic Function and the Cost Function

UNIVERSITY of WEST FLORIDA

- The logistic function, also known as the sigmoid function, is a function that maps a weighted input (net input) to a probability between 0 and 1.
- The logistic function is defined as $\sigma(z) = \frac{1}{1+e^{-z}}$, where z is the net input $z = \boldsymbol{w}^{\boldsymbol{T}} + b$
- The cost function is a function that measures how well the model predicts the class labels for the training examples.
- In logistic regression, a common cost function is the negative log-likelihood function, also called cross-entropy loss function, which is defined as

$$L(\mathbf{w}, \mathbf{b}|\mathbf{x}) = \Sigma_i^n \left(-y^{(i)} log\left(\sigma\left(z(i)\right)\right) - \left(1 - y^{(i)}\right) log\left(1 - \sigma\left(z^{(i)}\right)\right)\right)$$

- The goal of training a logistic regression model is to find the parameter values that minimize the cost function.

Deriving likelihood function

UNIVERSITY of WEST FLORIDA

- Given a binary classification with class 0 and 1, we can model the label as Bernoulli variable, with the probability p being 1: $Y \sim Bern(p)$
- For single data point, we can express the probability of the label being 1 or 0 given the input as:

$$P\left(Y = 1 \middle| X = x^{(i)}\right) = \sigma\left(z^{(i)}\right)$$
$$P\left(Y = 0 \middle| X = x^{(i)}\right) = 1 - \sigma\left(z^{(i)}\right)$$

- Given that all training examples are independent, the probability of all events occurring is the likelihood of the training labels. We can express this as:

$$\mathcal{L}(\mathbf{w}, \mathbf{b}|\mathbf{x}) = \Pi_{i=1}^{i=n} p\left(y^{(i)} \middle| x^{(i)}; \mathbf{w}, \mathbf{b}\right)$$

- By substituting the probability mass function of the Bernoulli variable, we arrive at the expression of the likelihood, which we try to maximize by changing the model parameters:

$$\mathcal{L}(\mathbf{w}, \mathbf{b}|\mathbf{x}) = \Pi_{i=1}^{i=n} \sigma\left(z^{(i)}\right)^{y^{(i)}} \left(1 - \sigma\left(z^{(i)}\right)\right)^{\left(1-y^{(i)}\right)}$$

Converting loglikelihood to loss function

- Since logarithm is a monotonic function, we can use it to transform the likelihood function into a more convenient form for optimization.
- Specifically, we can use the logarithm to convert the product of probabilities in the likelihood function into a sum of logarithms.
- Using the logarithm allows us to convert the problem of maximizing the likelihood function into the equivalent problem of minimizing the negative log likelihood function (also known as the loss function).
  - Maximizing $\mathcal{L}(\mathbf{w}, \mathbf{b}|\mathbf{x})$ = Minimizing $-\mathcal{L}(\mathbf{w}, \mathbf{b}|\mathbf{x})$
- The loss function can then be used as the objective function in an optimization algorithm like gradient descent to find the maximum likelihood estimates of the model parameters.
- We can now re-write the log likelihood as a "loss" function to minimize

$$\mathcal{L}(\mathbf{w}, \mathbf{b}|\mathbf{x}) = \Pi_{i=1}^{i=n} \sigma\left(z^{(i)}\right)^{y^{(i)}} \left(1 - \sigma\left(z^{(i)}\right)\right)^{\left(1-y^{(i)}\right)}$$

Loss function: $L(\mathbf{w}, \mathbf{b}|\mathbf{x})$

$$max\left(\mathcal{L}(\mathbf{w}, \mathbf{b}|\mathbf{x})\right) = max\left(log\left(\mathcal{L}(\mathbf{w}, \mathbf{b}|\mathbf{x})\right)\right) = min\left(-log\left(\mathcal{L}(\mathbf{w}, \mathbf{b}|\mathbf{x})\right)\right)$$

$$L(\mathbf{w}, \mathbf{b}|\mathbf{x}) = \Sigma_i^n \left(-y^{(i)} log\left(\sigma(z(i))\right) - \left(1 - y^{(i)}\right) log\left(1 - \sigma\left(z^{(i)}\right)\right)\right)$$

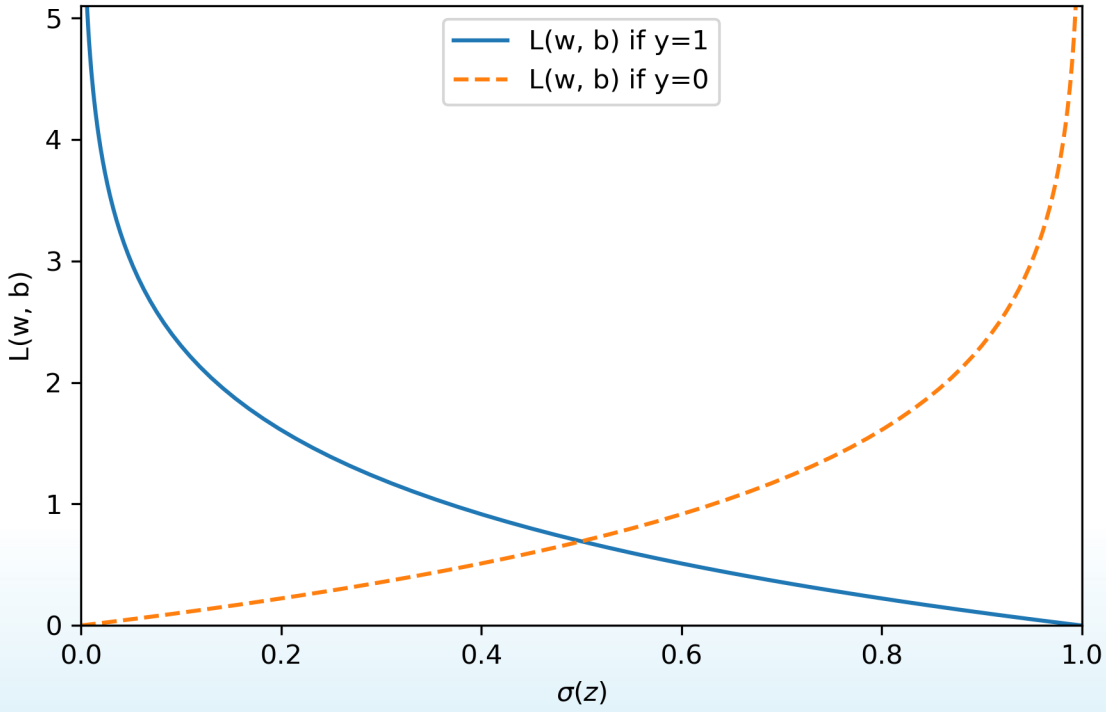
Interpreting Loss function

$$L(\mathbf{w}, \mathbf{b}|\mathbf{x}) = \Sigma_i^n \left(-y^{(i)} \, log \left(\sigma\left(z(i)\right)\right) - \left(1 - y^{(i)}\right) log \left(1 - \sigma\left(z^{(i)}\right)\right)\right)$$

Let's calculate the loss for one single example:

$$L(\sigma(z), y; \mathbf{w}, b) = \begin{cases} -log\left(\sigma\left(z(i)\right)\right) & if\ y = 1 \\ -log\left(1 - \sigma\left(z^{(i)}\right)\right) & if\ y = 0 \end{cases}$$

Let's plot the loss for different $\sigma(z)$:



Converting Adaline to Logistic Regression

UNIVERSITY of WEST FLORIDA

- Replace Adaline's loss function with logistic loss
- Replace linear activation with sigmoid activation
- Code implementation will follow

Logistic Regression Parameter Update

Using calculus we can show the parameter update is the same as Adaline

$$L(\mathbf{w}, \mathbf{b}|\mathbf{x}) = \Sigma_i^n \left(-y^{(i)} \, log\left(\sigma\left(z(i)\right)\right) - \left(1 - y^{(i)}\right) log\left(1 - \sigma\left(z^{(i)}\right)\right)\right)$$

$$\frac{\partial L}{\partial w_j} = \frac{\partial L}{\partial a}\frac{\partial a}{\partial z}\frac{\partial z}{\partial w_j} \text{ where } a = \sigma(z) = \frac{1}{1+e^{-z}}$$

Omitting averaging over the training example for brevity:

$$\frac{\partial L}{\partial a} = \frac{\mathrm{a} - \mathrm{y}}{\mathrm{a} \cdot (1 - \mathrm{a})} \qquad \frac{da}{dz} = \frac{e^{-z}}{(1 + e - z)^2} = a \cdot (1 - a) \qquad \frac{\partial z}{\partial w_j} = \mathrm{x_j}$$

$$\frac{\partial L}{\partial w_j} = (a - y)x_j = -(y - a)x_j$$

We take steps towards opposite direction of gradient, hence:

$$w_j \coloneqq w_j + \eta(y - a)x_j$$

Similarly, you can derive:

$$b \coloneqq b + \eta(y - a)$$

Training a logistic regression with scikit learn

UNIVERSITY of WEST FLORIDA

- Let's jump in the code to use scikit-learn more optimized implementation of logistic regression.
- Supports multi-class settings off-the-shelve

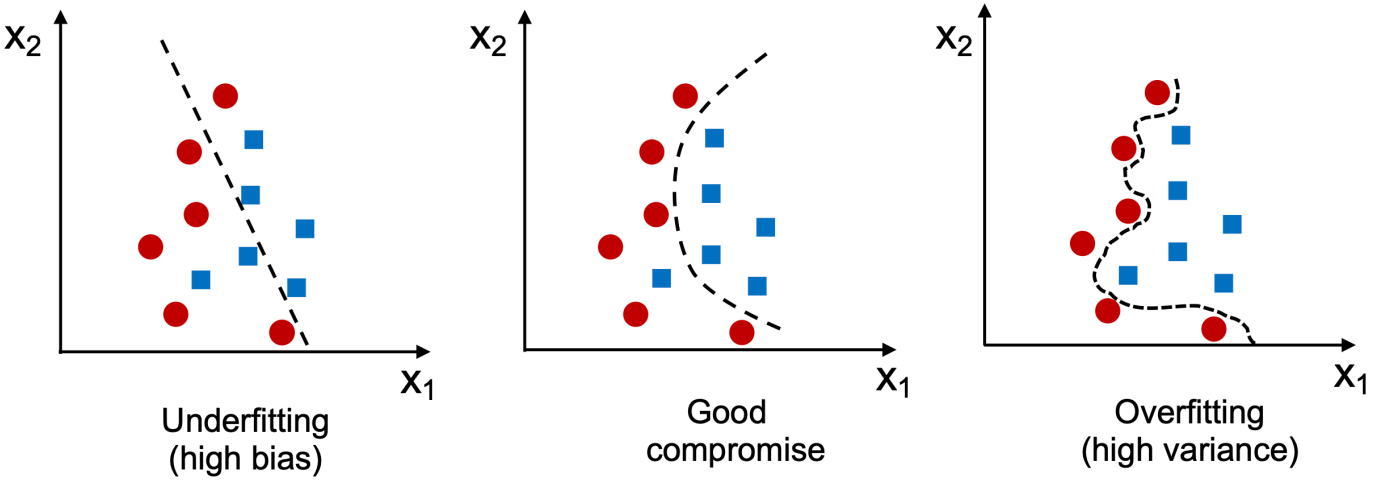
Optimizing Convex Loss Functions in Logistic Regression

UNIVERSITY of WEST FLORIDA

- Several algorithms exist for minimizing convex loss functions like logistic regression.
- SGD is not always the best optimization algorithm to use.
- scikit-learn implements a range of optimization algorithms, including:
  - Newton-CG
  - L-BFGS
  - Liblinear
  - SAG
  - SAGA
- There are advantages to using different algorithms depending on the specific use case.

Overfitting vs Underfitting

- Overfitting and underfitting are common problems in machine learning.
- Overfitting occurs when a model is too complex and performs well on the training data, but poorly on unseen data (test data). This is caused by having too many parameters.
- Underfitting occurs when a model is not complex enough to capture the patterns in the training data, resulting in poor performance on both the training and test data. This is caused by a high bias.



Bias-Variance Tradeoff

UNIVERSITY of WEST FLORIDA

- Bias-variance tradeoff is a fundamental concept in ML
- Variance measures the consistency of model predictions for a particular example when re-trained multiple times on different subsets of the training dataset
- Variance is a measure of sensitivity to randomness in the training data
- Bias measures the difference between the model predictions and the correct values when re-trained multiple times on different training datasets
- Bias is a measure of systematic error not due to randomness

Regularization

- Regularization is a technique for reducing the complexity of a model and avoiding overfitting.
- Regularization adds a penalty term to the loss function that the model is trying to minimize.
- The goal is to reduce the magnitude of the model weights, which reduces the complexity of the model.
- The most commonly used form of regularization is L2 regularization, also known as L2 shrinkage or weight decay.
- L2 regularization adds a penalty term equal to the sum of the squares of the model weights, multiplied by a regularization parameter λ.
- The regularization parameter determines the strength of the penalty term, and can be adjusted to find the optimal trade-off between model complexity and fit to the training data.

$$\frac{\lambda}{2n}\left|\left|\mathbf{w}\right|\right|^2 = \frac{\lambda}{2n}\Sigma_{j=1}^{m} w_j^2$$

Regularized Logistic Regression

$$L(\mathbf{w}, \mathbf{b}|\mathbf{x}) = \frac{1}{n}\Sigma_i^n \left(-y^{(i)}\, log\left(\sigma\left(z(i)\right)\right) - \left(1 - y^{(i)}\right) log\left(1 - \sigma\left(z^{(i)}\right)\right)\right) + \frac{\lambda}{2n}\left||\mathbf{w}|\right|^2$$

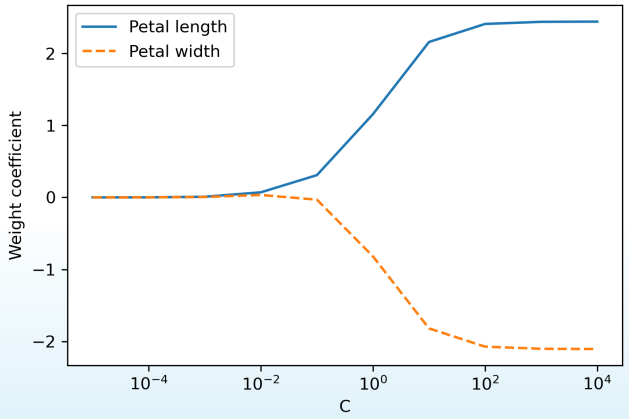
- Regularization parameter λ controls the strength of the fit to the training data
- Increasing the value of λ increases the regularization strength
- C is the inverse of λ and is used in scikit-learn's LogisticRegression class
- Decreasing the value of C increases the regularization strength
- Regularization is achieved by adding a penalty term to the loss function
- The partial derivative of the regularized loss function has an added penalty term compared to the unregularized loss function
- The regularized loss function balances between fitting the training data well and preventing overfitting

Partial derivative of unregularized loss:

$$\frac{\partial L(\mathbf{w}, \mathbf{b}|\mathbf{x})}{\partial w_j} = \left(\frac{1}{n}\Sigma_1^n\left(\sigma\left(\mathbf{w}^T\boldsymbol{x}^{(i)}\right) - y^{(i)}\right)x_j^{(i)}\right)$$

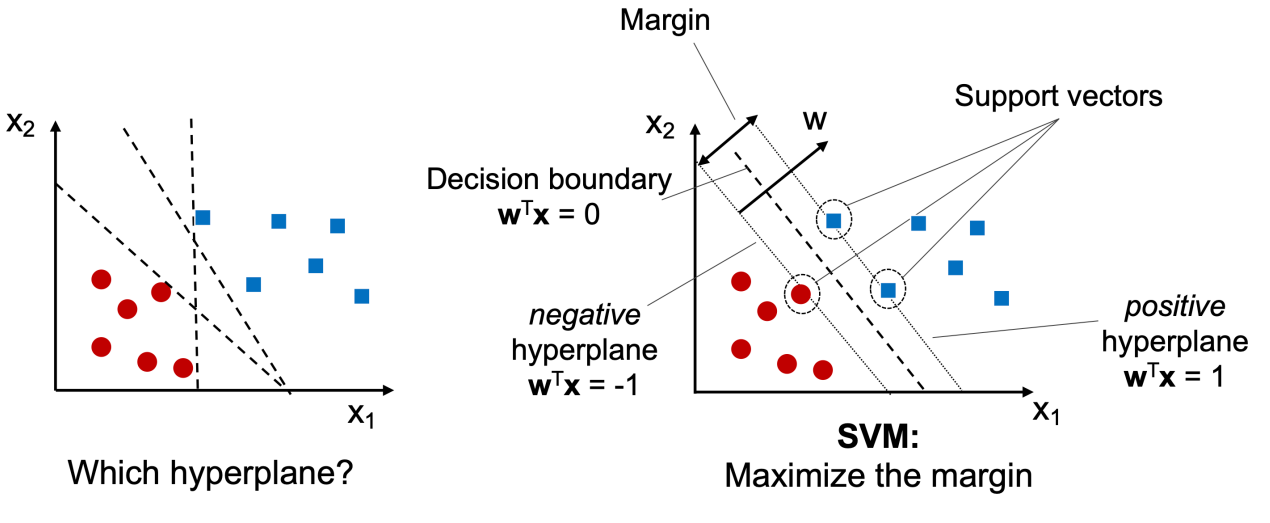
Partial derivative of unregularized loss:

$$\frac{\partial L(\mathbf{w}, \mathbf{b}|\mathbf{x})}{\partial w_j} = \left(\frac{1}{n}\Sigma_1^n\left(\sigma\left(\mathbf{w}^T\boldsymbol{x}^{(i)}\right) - y^{(i)}\right)x_j^{(i)}\right) + \frac{\lambda}{n}w_j$$



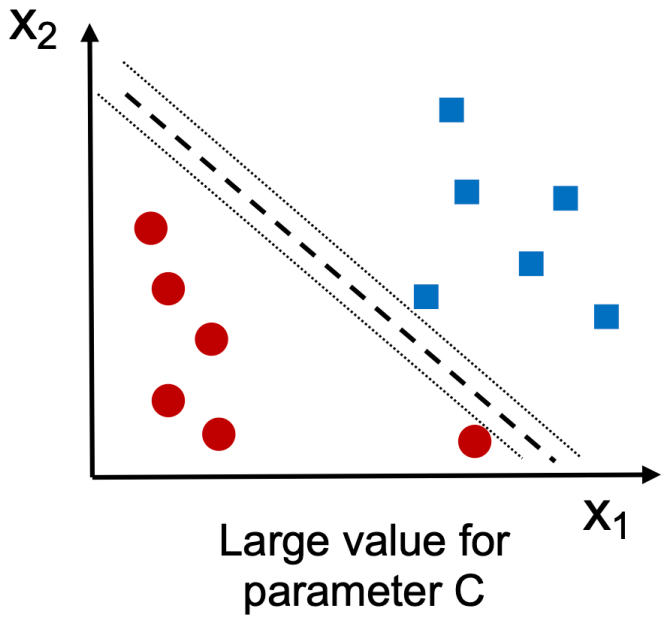
SVM

- Introduction to SVM
  - SVM is a supervised learning algorithm used for classification and regression.
  - SVM aims to find a hyperplane (decision boundary) that maximizes the margin between the classes.
- What is the margin?
  - The margin is the distance between the separating hyperplane and the closest training examples, called support vectors.
- Why maximize the margin?
  - Decision boundaries with large margins tend to have lower generalization error, which means they generalize better to unseen data.

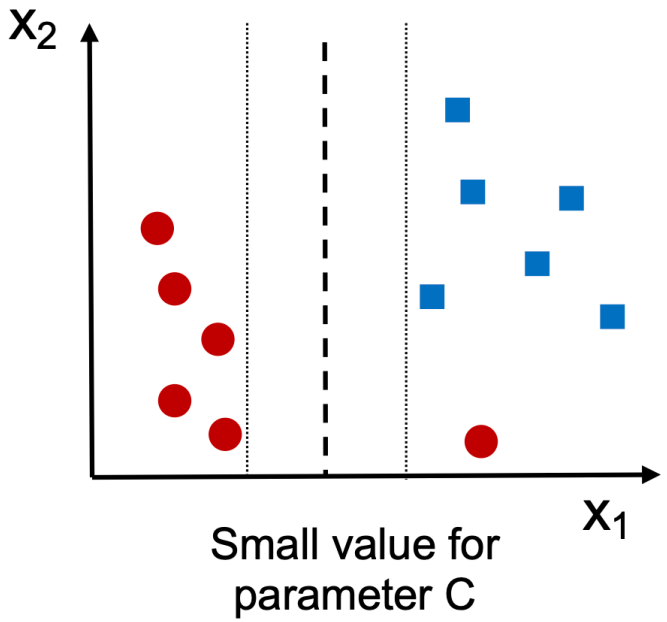


Introducing slack variable

- Introduction of slack variables in SVM
- Slack variables allow for relaxation of linear constraints to address misclassification
- Control the trade-off between bias and variance with parameter C
- Increasing the value of C leads to larger error penalties and stricter restrictions against misclassification
- Decreasing values of C increase bias (underfitting) and reduce variance (overfitting)



Large value for parameter C



Small value for parameter C

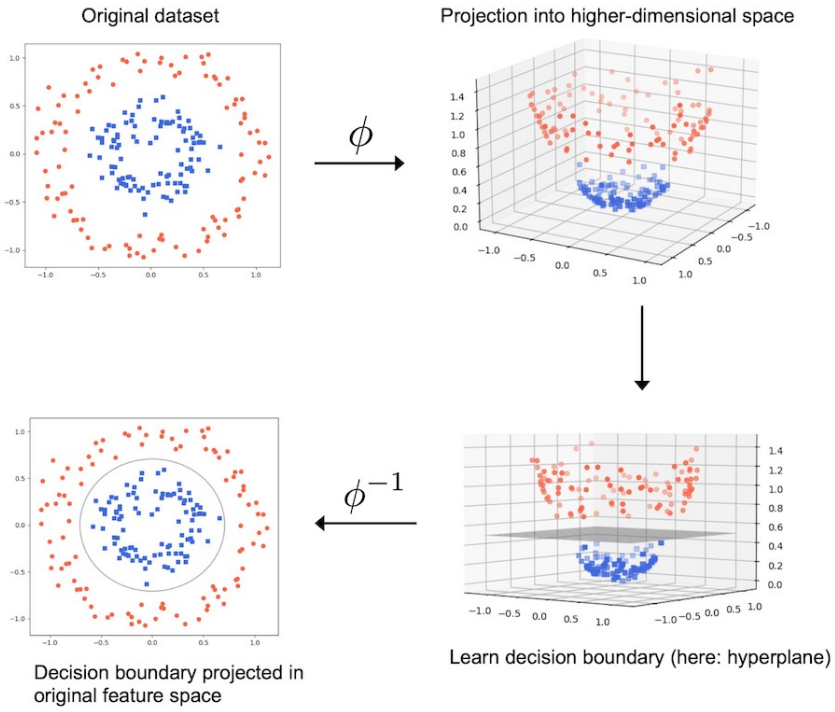
Training an SVM

UNIVERSITY of WEST FLORIDA

- Let's implement an SVM to classify the flowers in the Iris dataset using scikit-learn.

Kernel Method

- Non-linear data can be separated using a non-linear decision boundary
- Kernel method transforms non-linearly separable data into a higher dimensional space where the data becomes separable
- The transformed space is called feature space
- In feature space, a linear boundary can be used to separate the data
- The original, non-linear boundary can be found by transforming the linear boundary in feature space back to the original space
- The transformation from original space to feature space is done using a kernel function
- Common kernel functions include radial basis function (RBF), polynomial, and sigmoid
- Talking script:



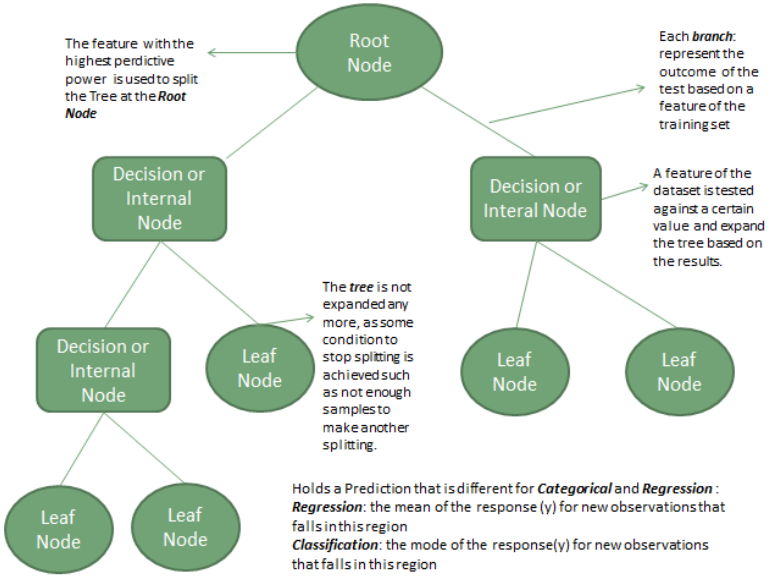
Training a Kernel SVM with scikit-learn

UNIVERSITY of WEST FLORIDA

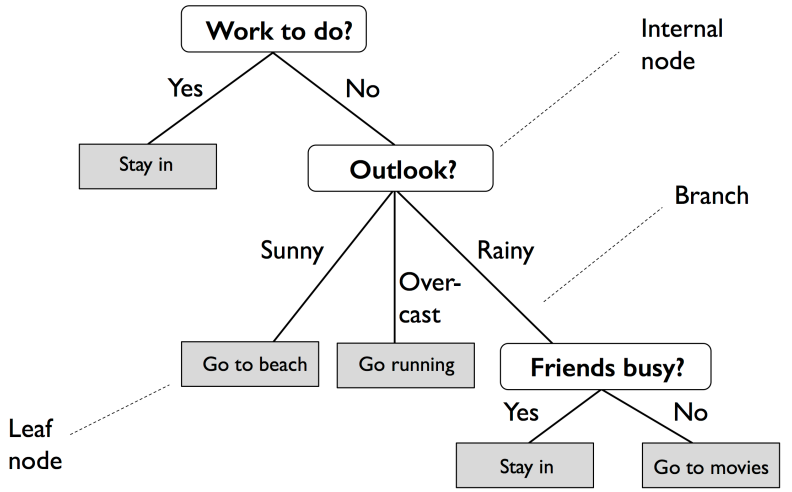
- In this section, we will use scikit-learn to train a kernel SVM on a non-linear dataset.

Understating Decision Trees

- Definition of decision trees
- How decision trees work
- Advantages and disadvantages of decision trees
- Maximizing information gain (IG) at each split
- Three impurity measures: Gini impurity, entropy, and classification error



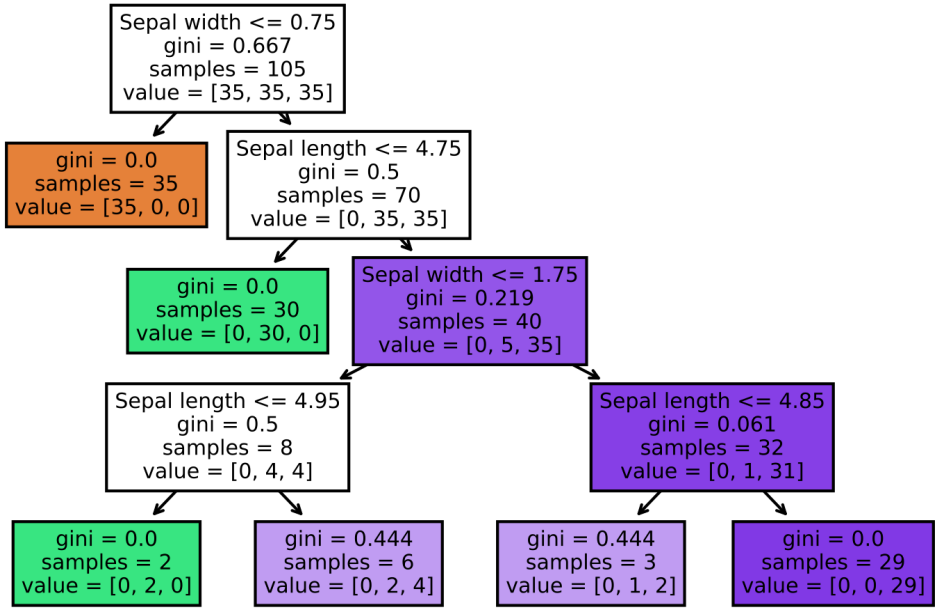
Picture is from financetrain.com



Constructing a Decision Tree

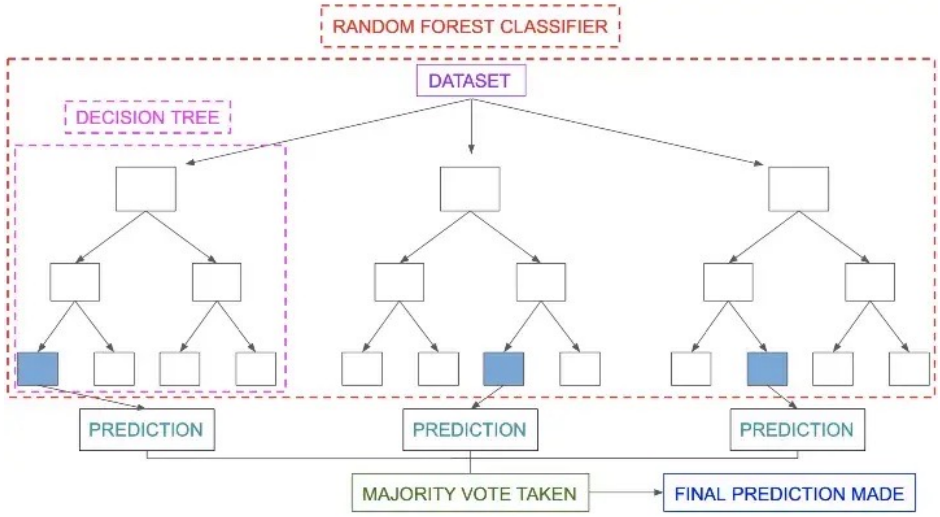
UNIVERSITY of WEST FLORIDA

- Splitting criteria
- Pruning
- Visual representation of a decision tree
- Handling missing values and categorical data



What are Random Forests?

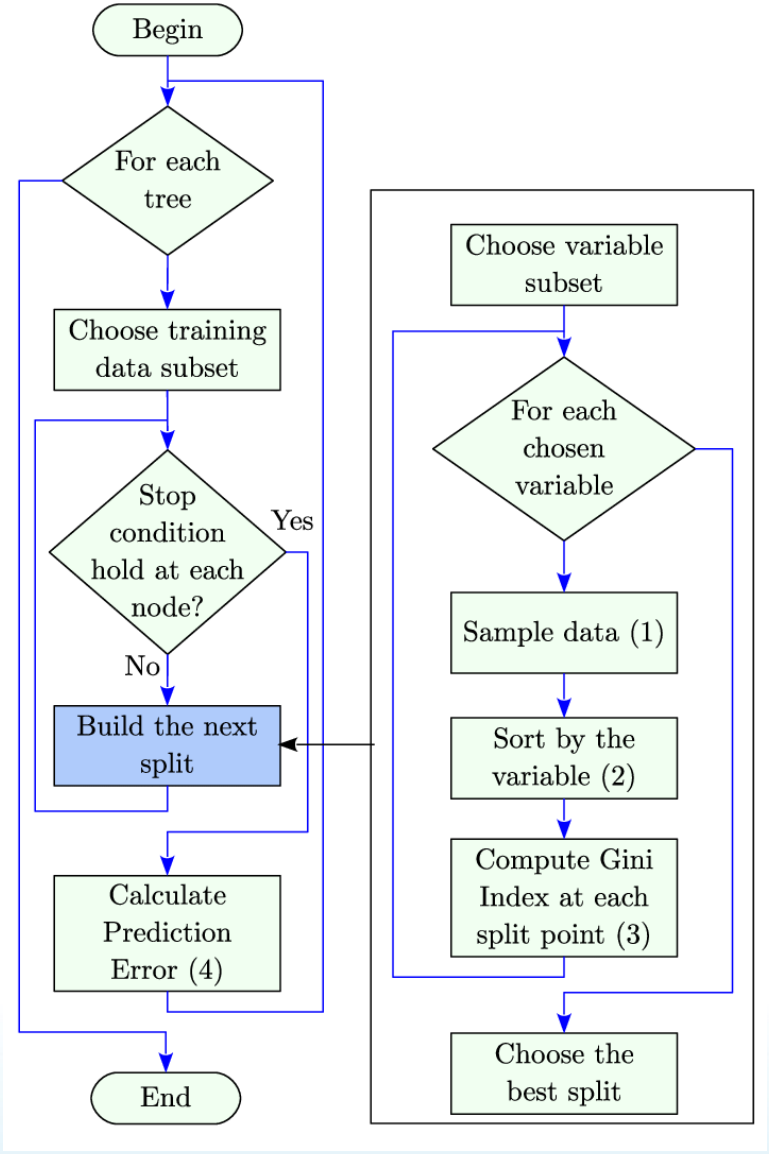
- Definition of random forests
- How random forests work
- Advantages and disadvantages of random forests
- Comparison with decision trees



A Simplified View of a Random Forest Classifier, by Karan Kashyap, Medium

Creating a Random Forest

1. Bootstrapping
2. Feature Bagging
3. Combining multiple decision trees
4. Hyperparameter tuning



Pros and Cons

UNIVERSITY of WEST FLORIDA

Random Forest

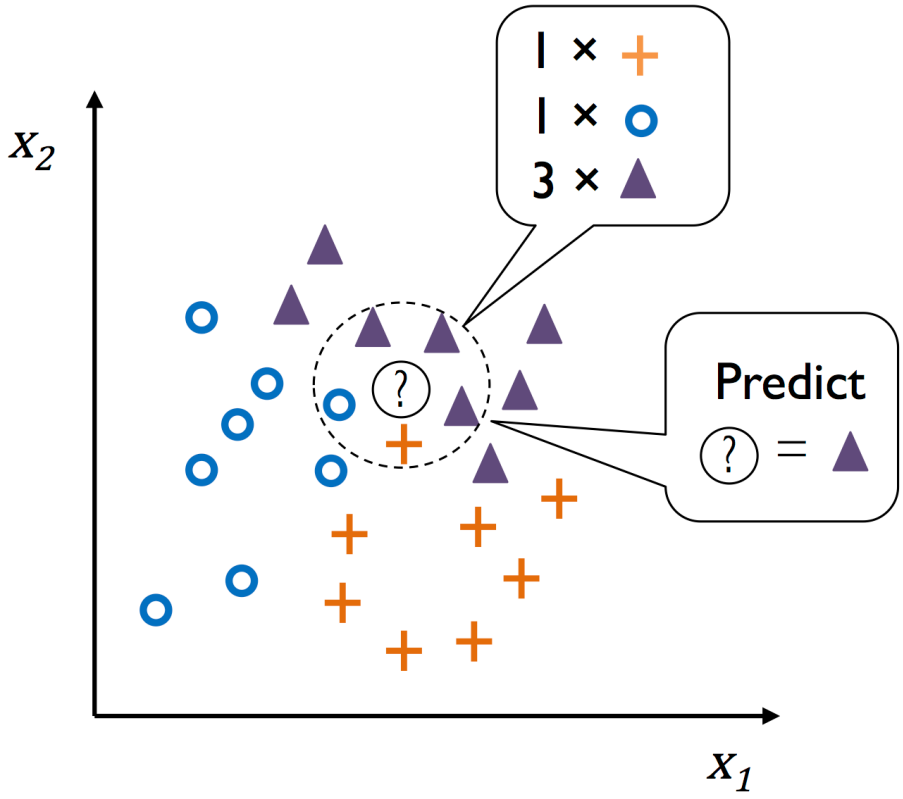
- Advantages of Random Forests:
  - Improved accuracy compared to individual decision trees
  - Reduced overfitting
  - Can handle complex relationships between features
- Disadvantages of Random Forests:
  - Slower to train and predict compared to individual decision trees
  - Higher memory usage
  - Can be more difficult to interpret

Decision Tree

- Advantages of Decision Trees:
  - Easy to interpret
  - Fast to train and predict
  - Can handle both numerical and categorical data
- Disadvantages of Decision Trees:
  - Prone to overfitting
  - Unstable, as small variations in the data can result in a completely different tree

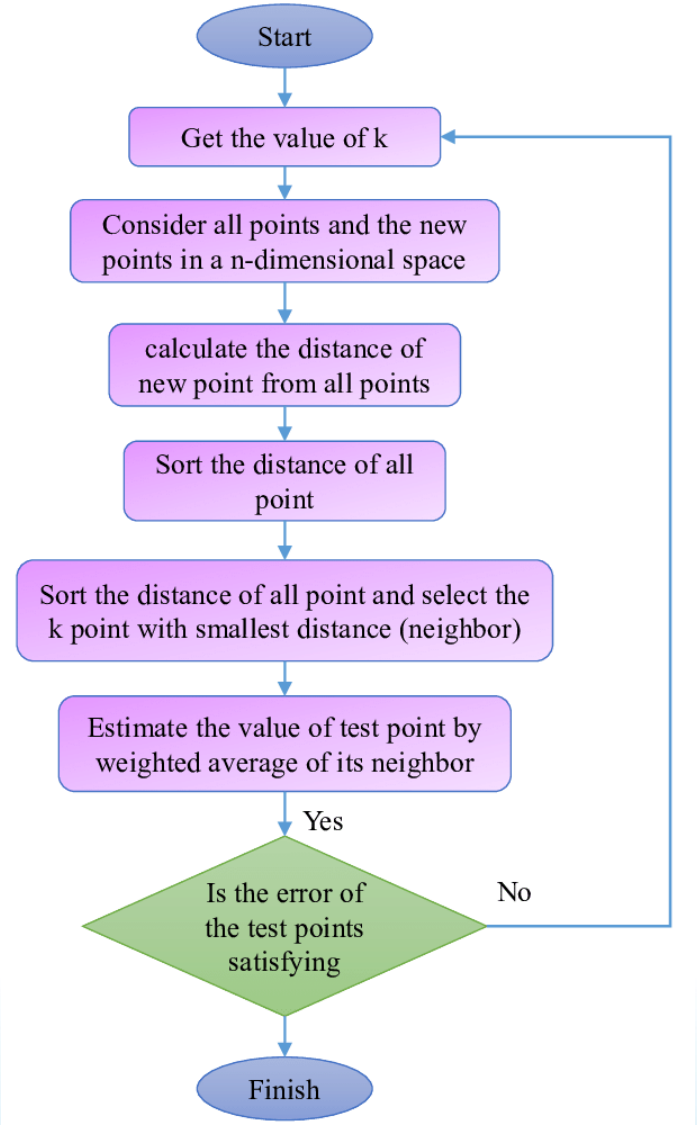
Introduction to KNN (K-Nearest Neighbors)

- Definition of KNN
- How KNN works
- Advantages and disadvantages of KNN
- Applications of KNN



How KNN Works

- Steps involved in the KNN algorithm
- How the number of neighbors (K) affects the performance of KNN
- How distance is measured in KNN
- Handling missing values and categorical data in KNN



Advantages and Disadvantages of KNN

UNIVERSITY of WEST FLORIDA

Advantages of KNN:

- Simplicity
- Non-linearity
- No assumptions about data distribution

Disadvantages of KNN:

- Computational expense
- Performance is sensitive to K
- Storage of large dataset

Module Summary

UNIVERSITY of WEST FLORIDA

- 5 steps for training supervised machine learning
- Training Perceptron via Scikit-learn
- Logistic Regression
  - Logistic Regression and Conditional Probability
  - Logistic Function and Cost Function
  - Deriving Likelihood Function
  - Converting Loglikelihood to Loss Function
  - Interpreting Loss Function
  - Converting Adaline to Logistic Regression
  - Logistic Regression Parameter Update
  - Training a Logistic Regression with Scikit-learn
  - Optimizing Convex Loss Functions in Logistic Regression
- Overfitting vs Underfitting
- Regularization
- Regularized Logistic Regression
- SVM
  - Introducing Slack Variables
  - Kernel Methods
- Decision Trees
- Random Forests
  - Creating a Random Forest
- KNN