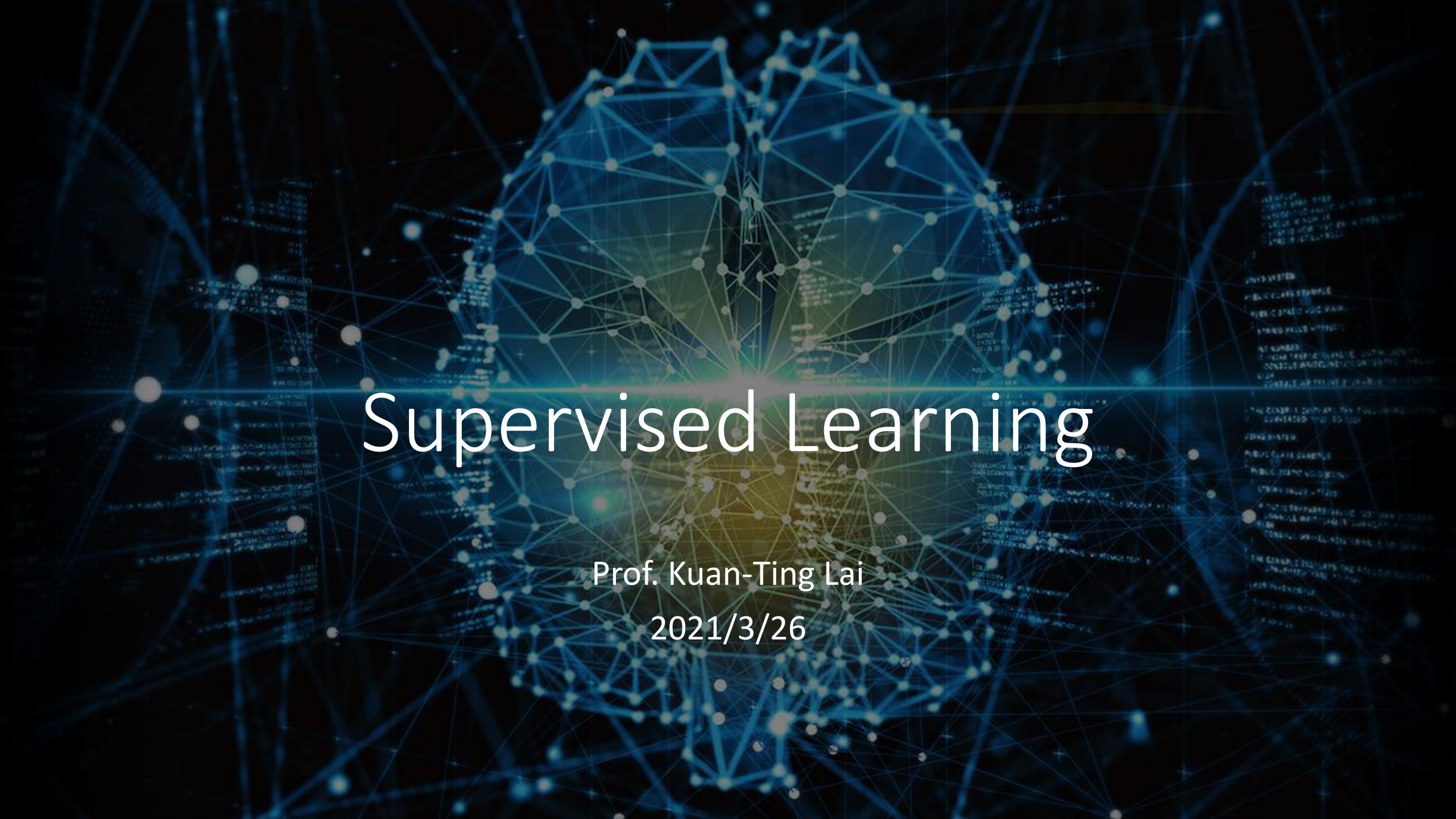
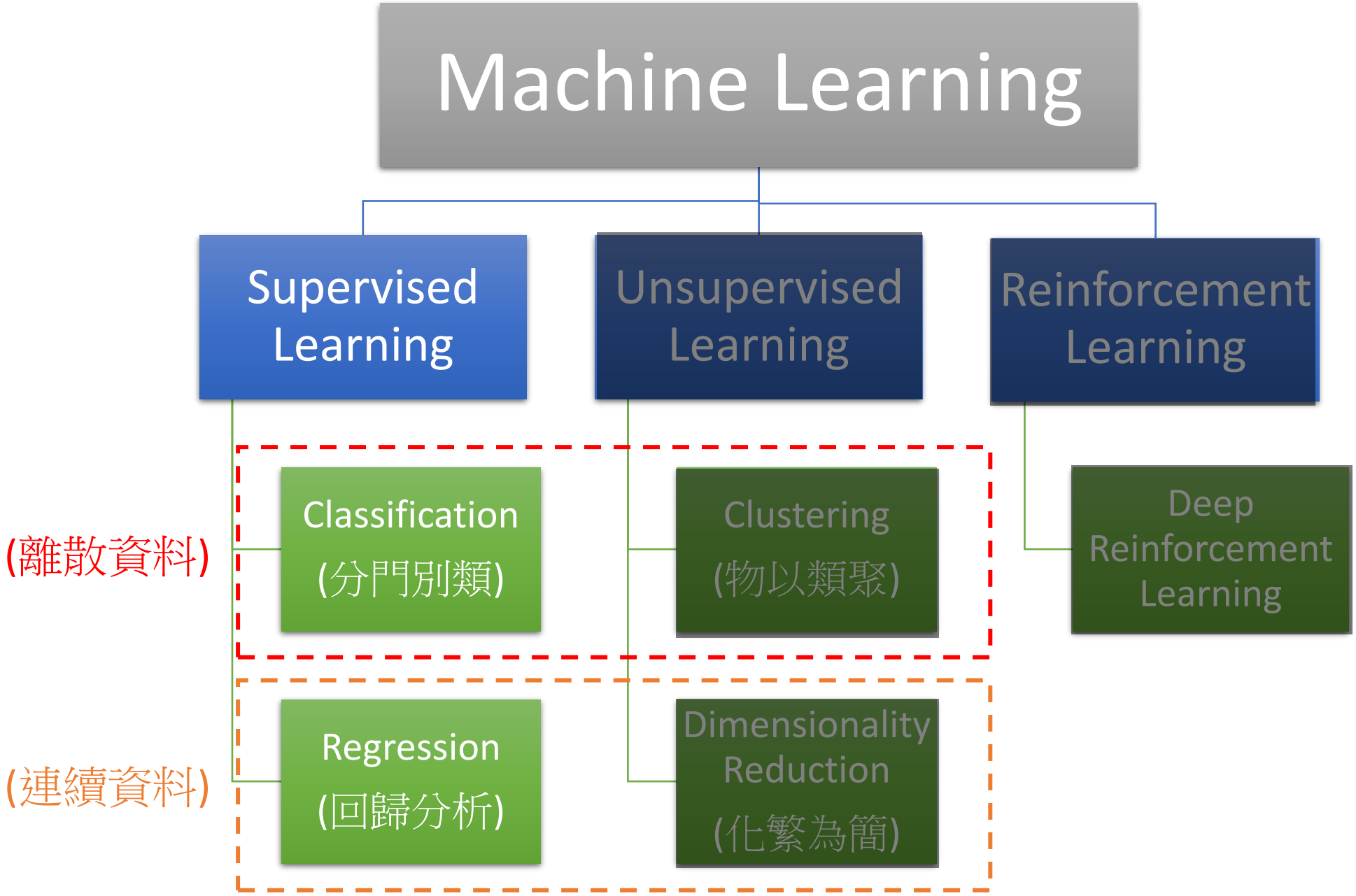


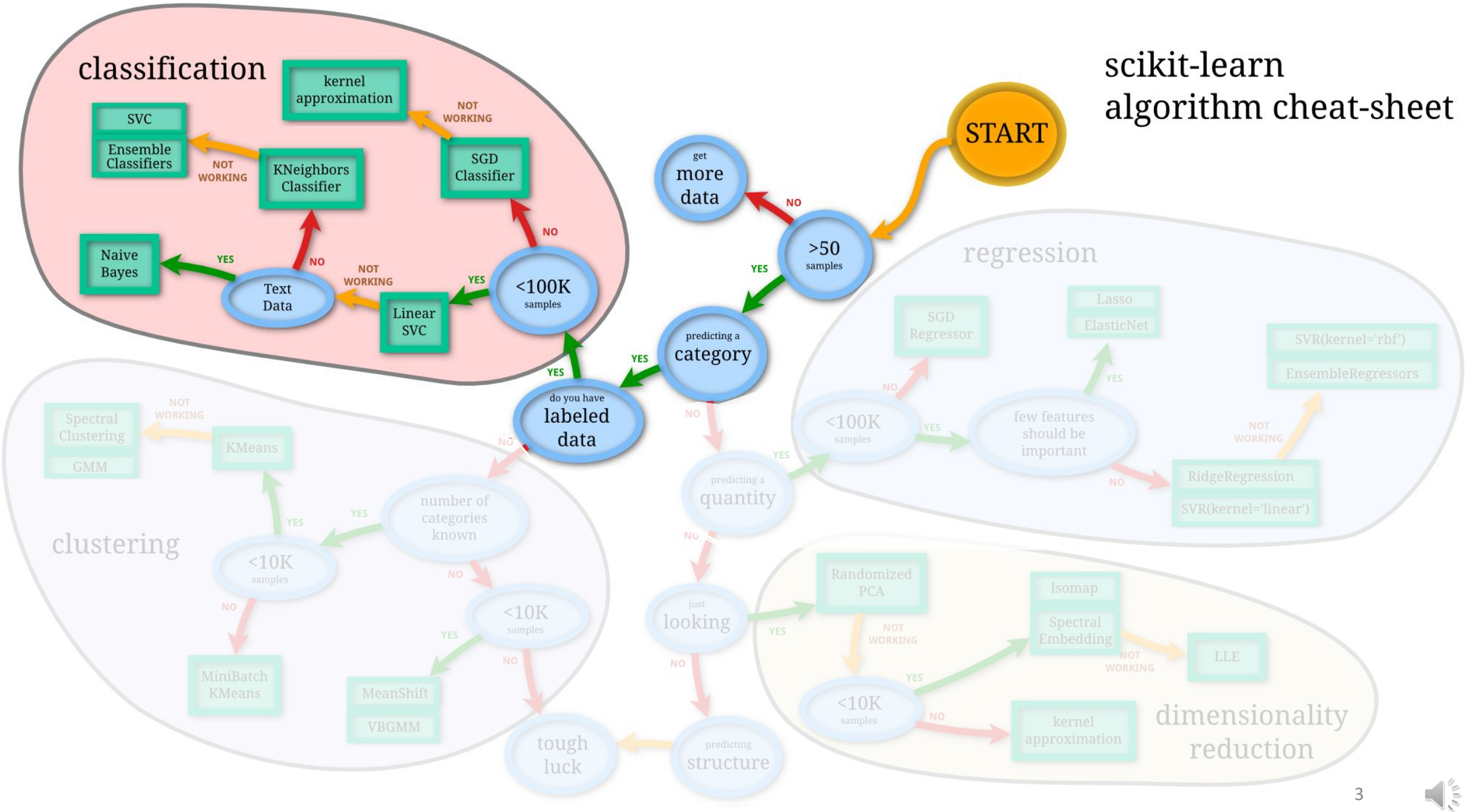
Supervised Learning

Prof. Kuan-Ting Lai

2021/3/26

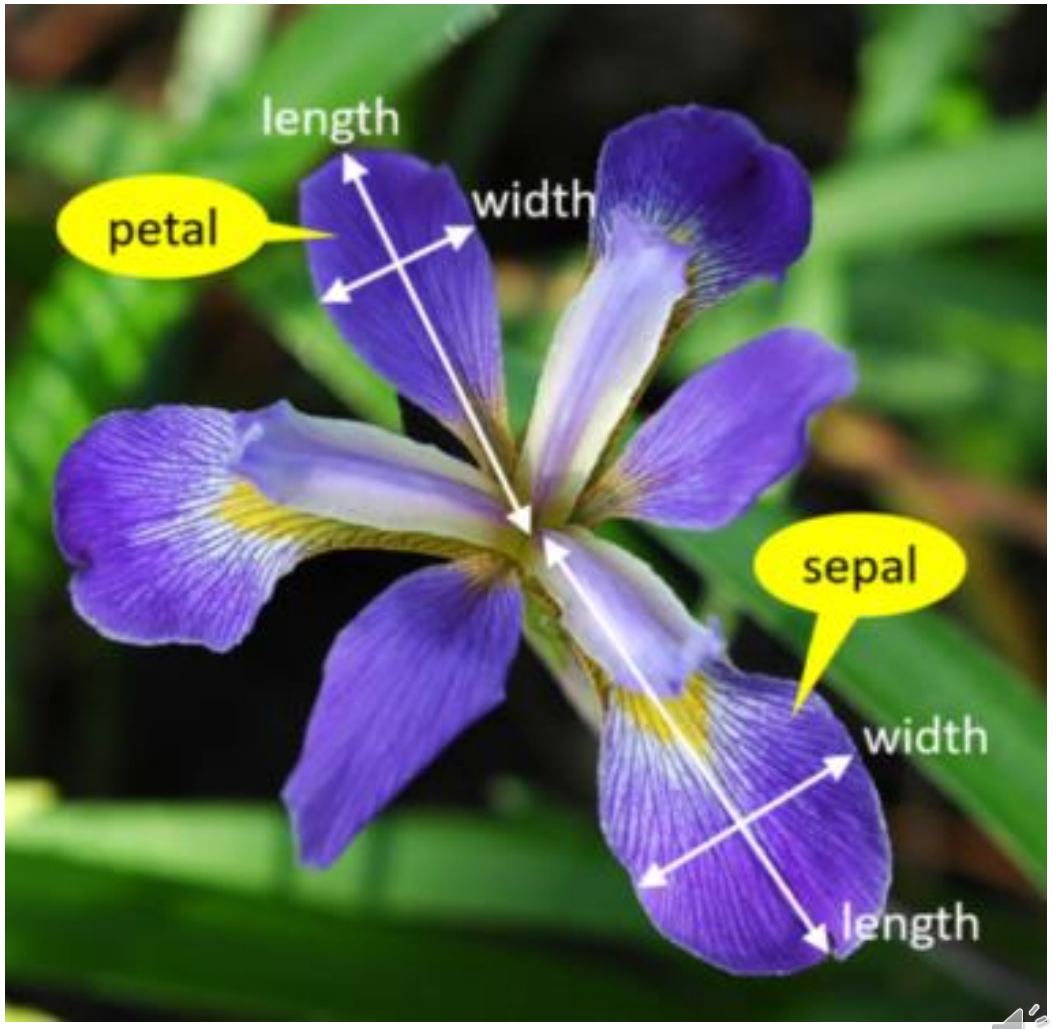

Machine Learning
Supervised Learning
Unsupervised Learning
Reinforcement Learning
(離散資料)
Classification
(分門別類)
Clustering
(物以類聚)
Deep Reinforcement Learning
(連續資料)
Regression
(回歸分析)
Dimensionality Reduction
(化繁為簡)

scikit-learn algorithm cheat-sheet



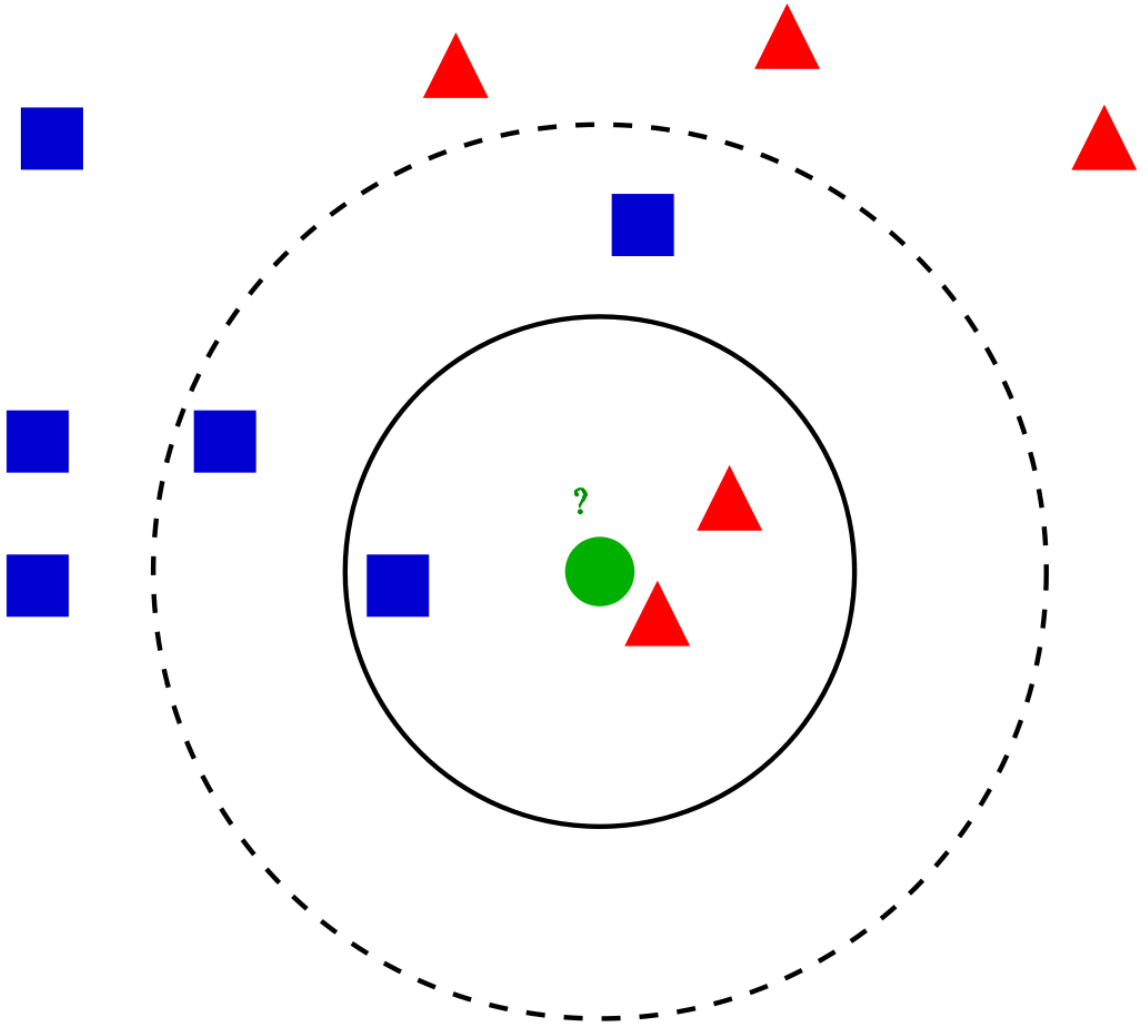
Iris Flower Classification

- 3 Classes
- 4 Features
- 50 samples for each class (Total: 150)
- Feature Dimension: 4
  - Sepal length (cm), sepal width, petal length, petal width



k-Nearest Neighbors (k-NN)

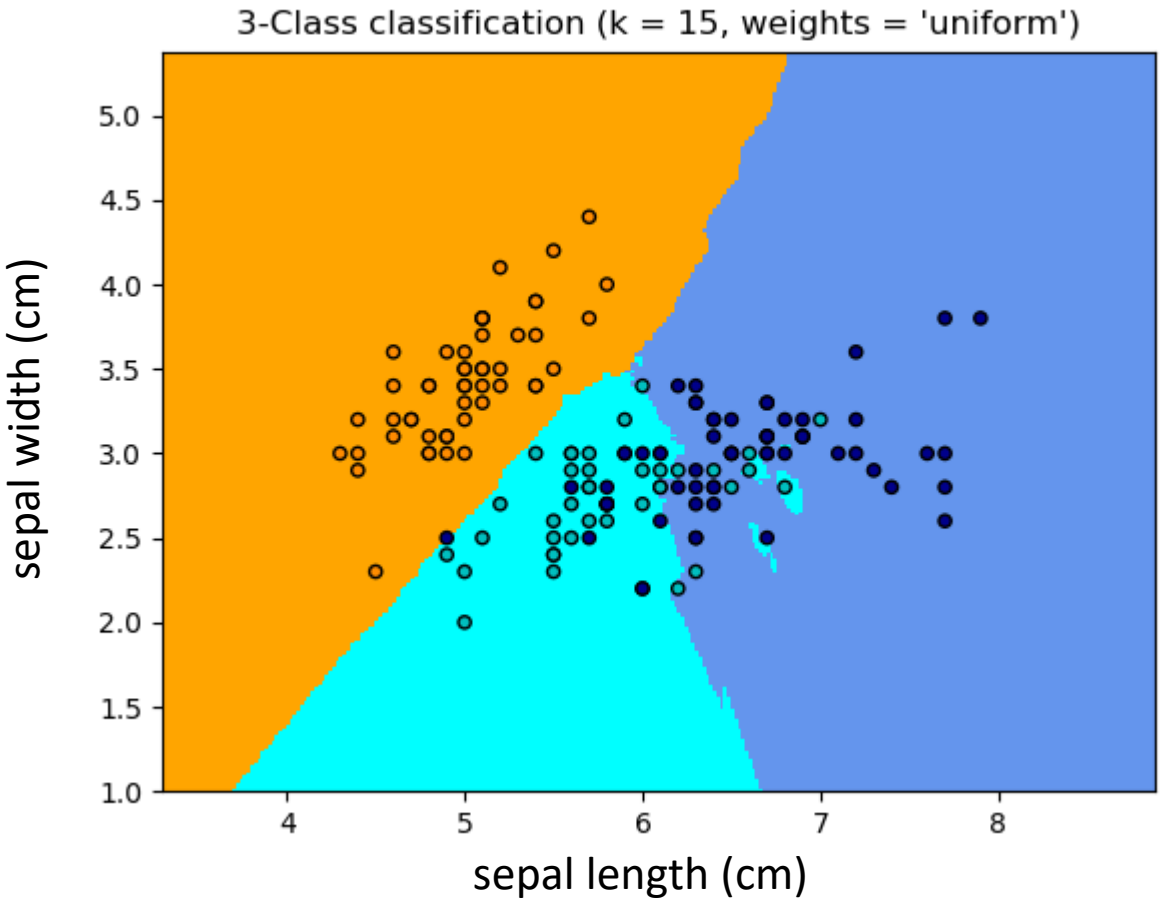
- Predict input using k nearest neighbors in training set
- No need for training
- Can be used for both classification and regression



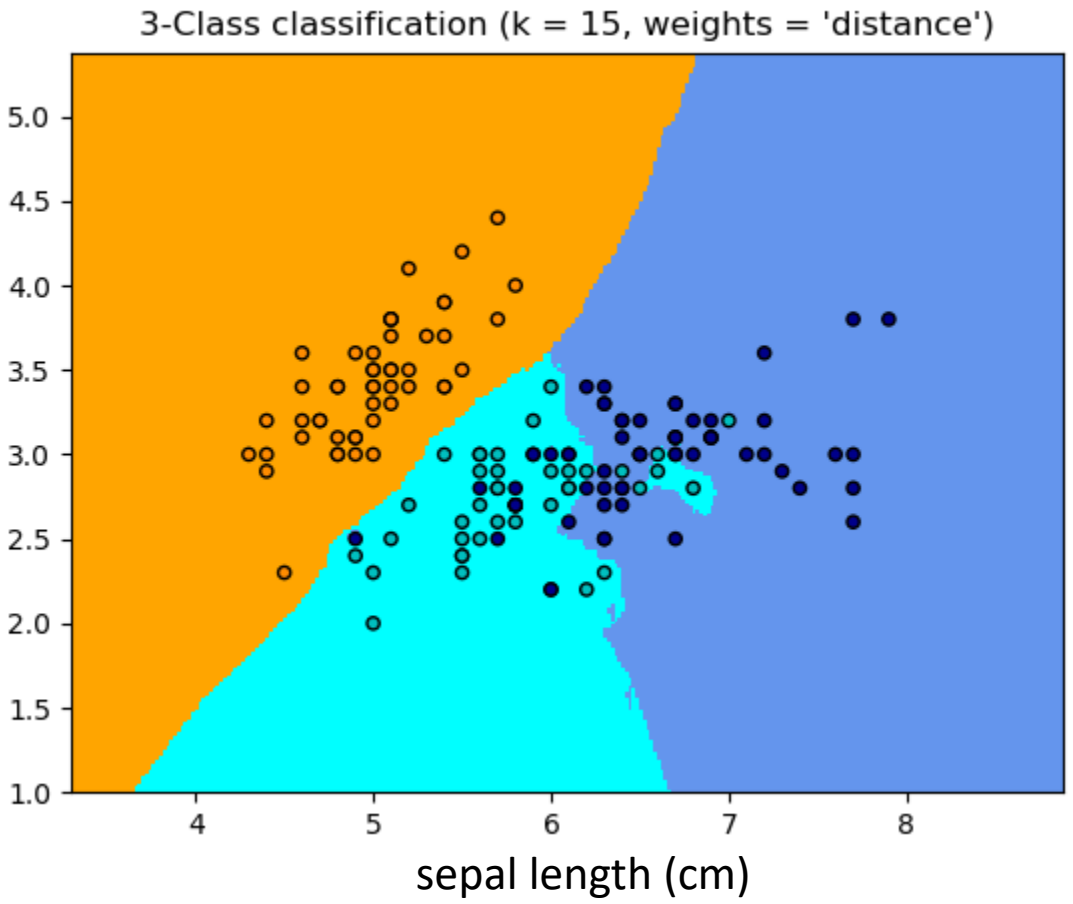
https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm

k-NN for Iris Classification

- Accuracy = 80.7%



- Accuracy = 92.7%



Linear Classifier

$$y' = w_1 x_1 + w_2 x_2 + b$$

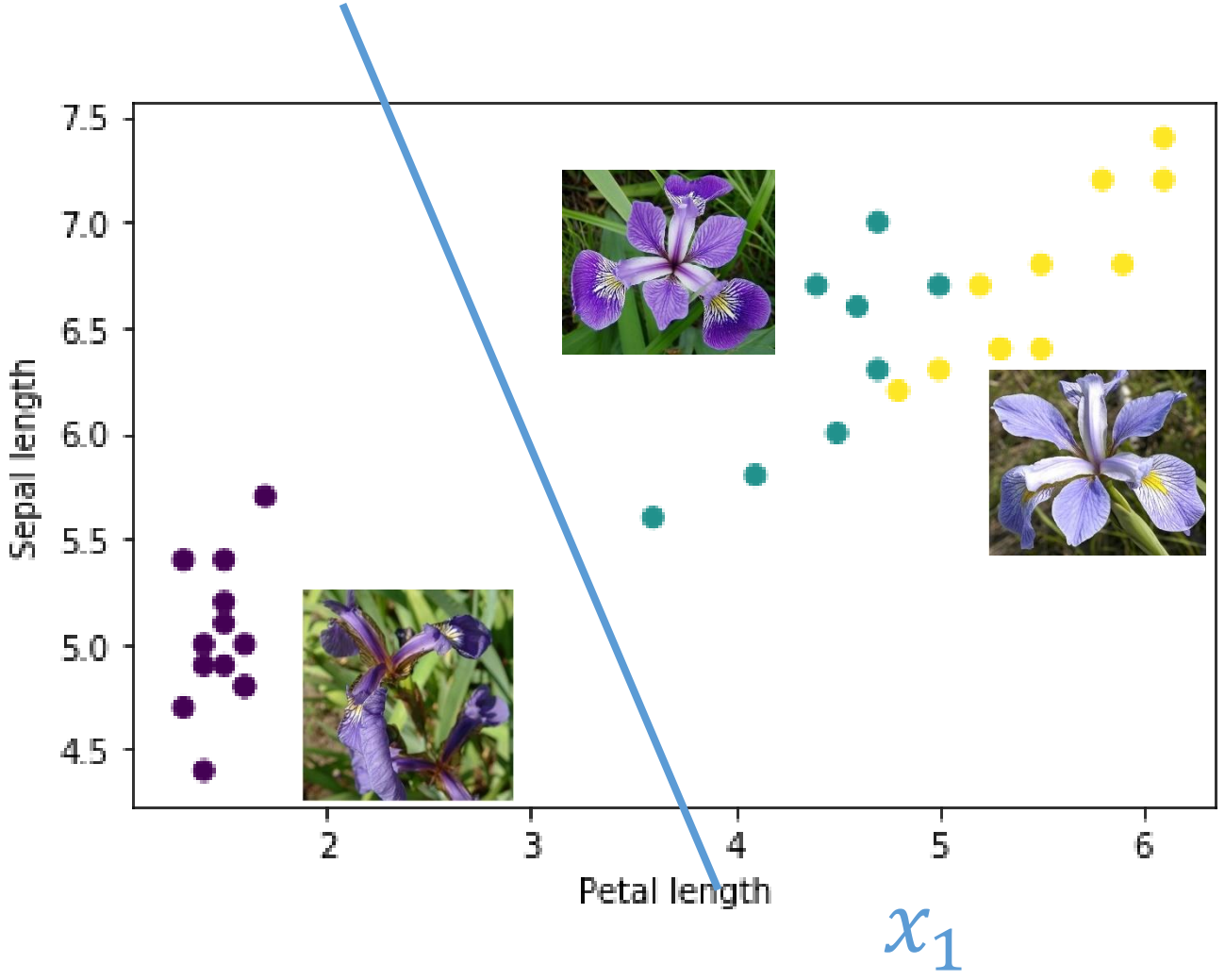
$$\Downarrow$$

$$\begin{bmatrix} w_1 & w_2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + b$$

$$y' = \mathbf{w}^T \mathbf{x} + b$$

$$w^T x + b \leq 0$$

$$w^T x + b > 0$$



Training Linear Classifier

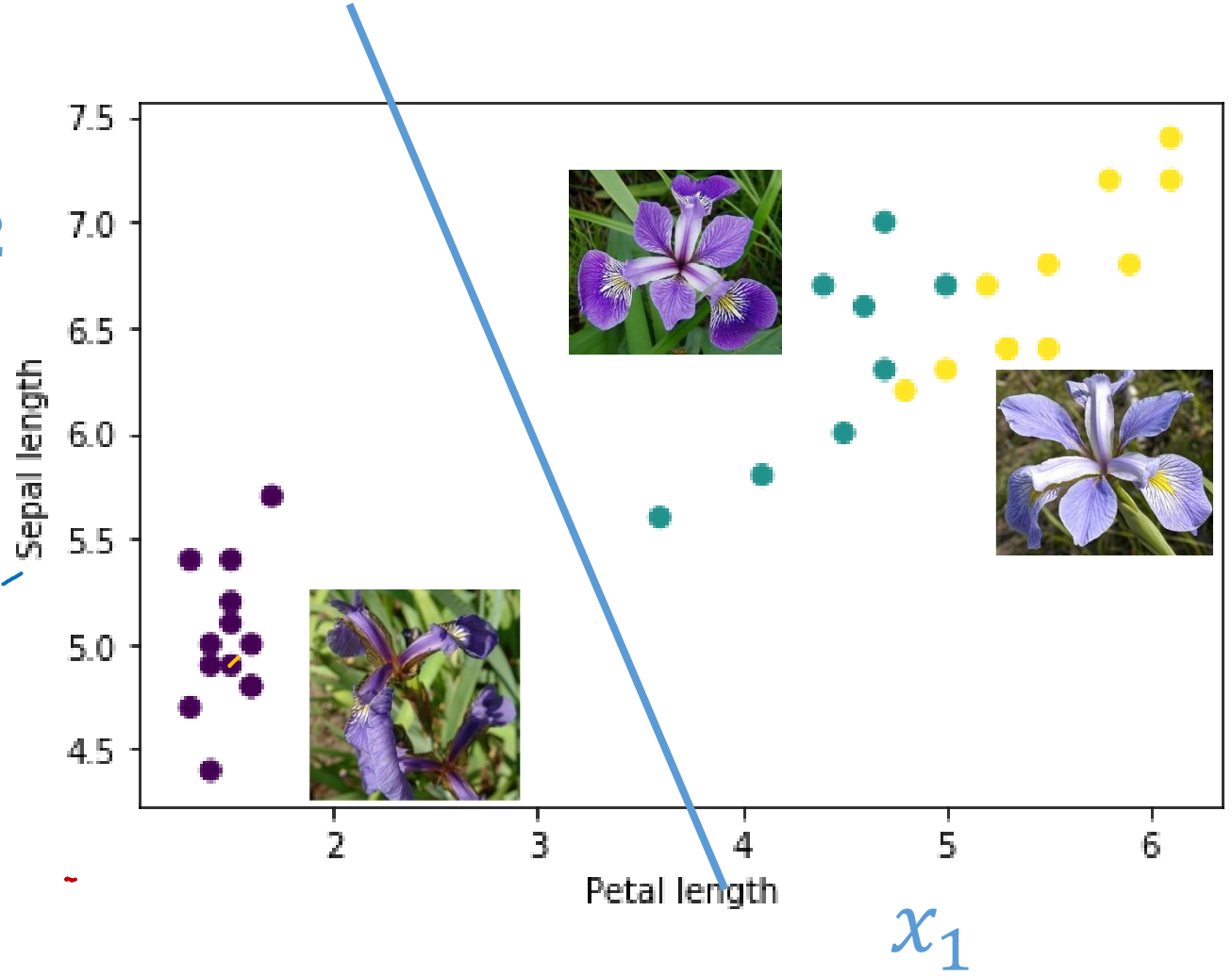
- Perceptron

$$\Delta = y - y'$$

Loop {

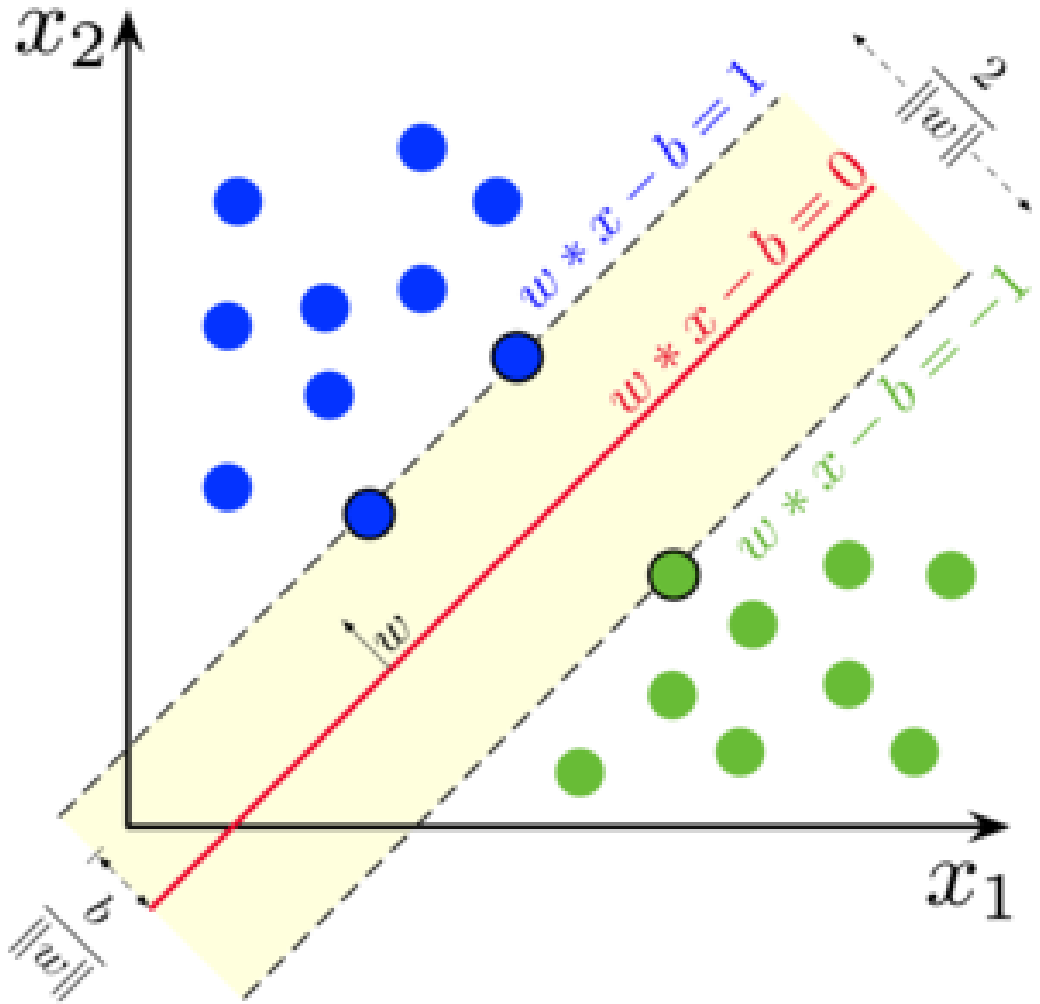
$$W_{t+1} \leftarrow W_t + \alpha(y - y')x$$

} Until $\Delta \approx 0$



Support Vector Machine (SVM)

- Choose the hyperplanes that have the largest separation (margin)



Loss Function of SVM

- Calculate prediction errors

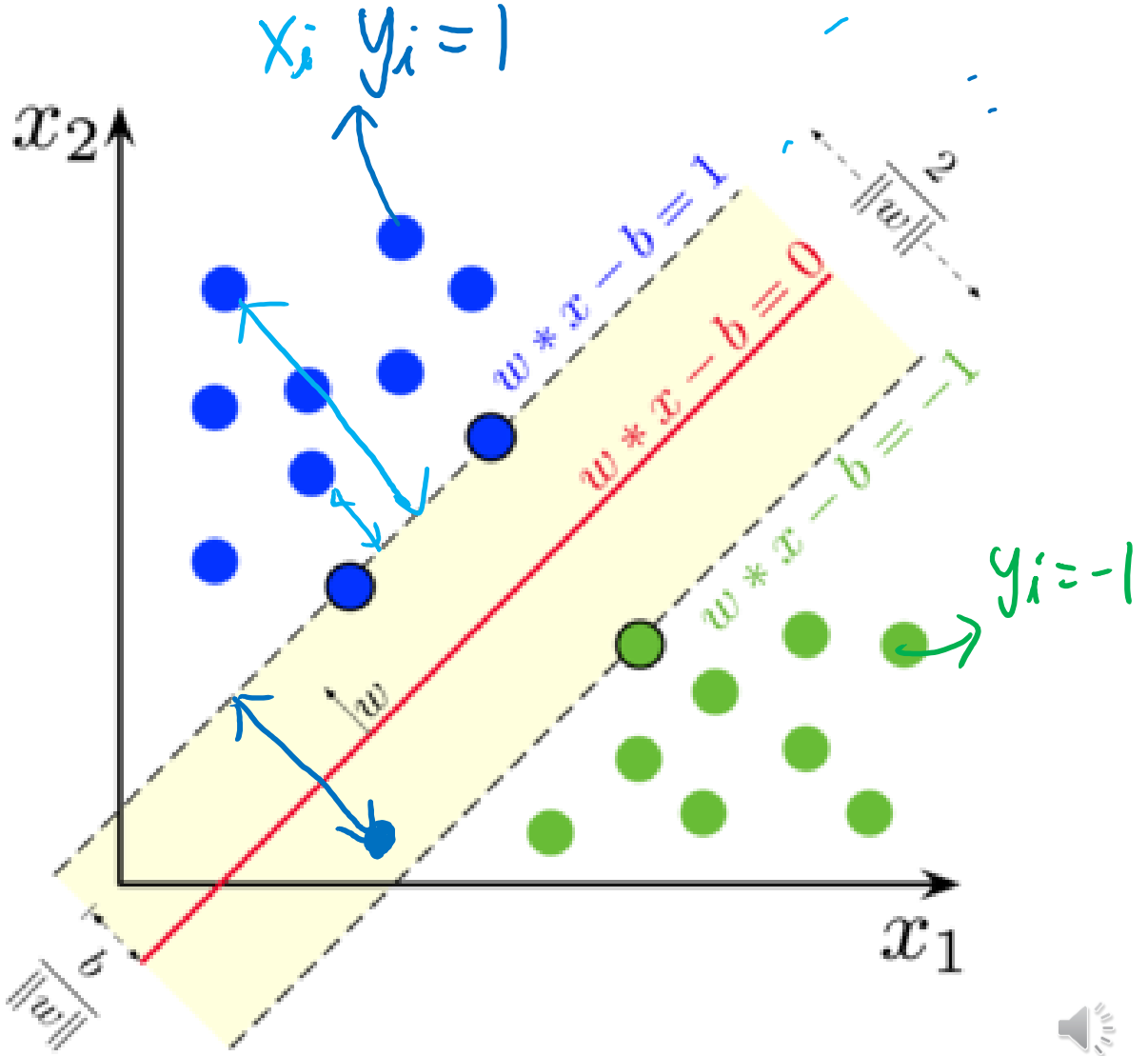
$$y_i' = W^T x_i - b \geq 1$$

$$y_i' = W^T x_i - b \leq -1$$

$$y_i(W^T x_i - b) \geq 1$$

$$\text{Loss} = \max(0, 1 - y_i(W^T x_i - b))$$

Hinge Loss

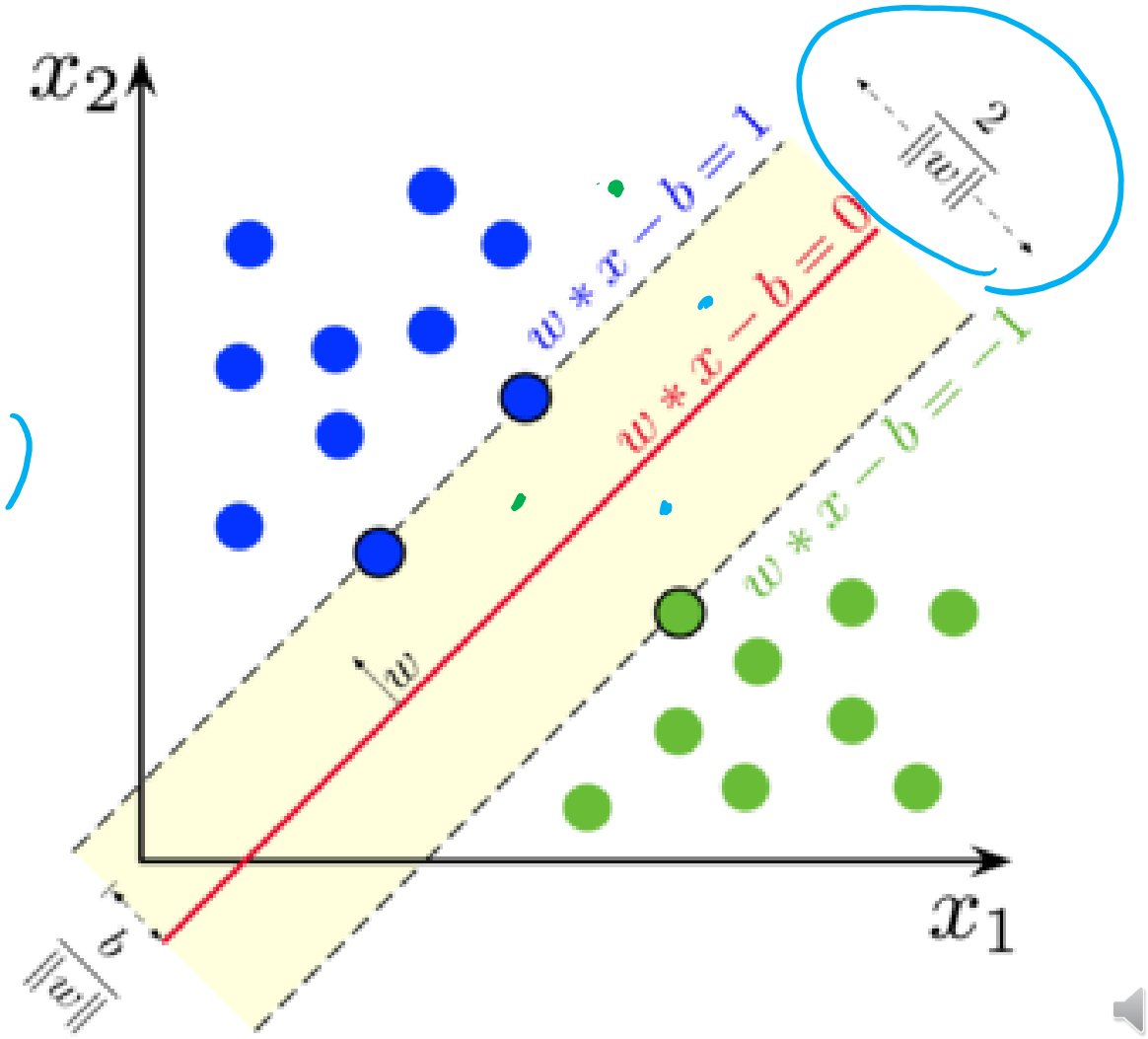


SVM Optimization

- Maximize the margin while reduce hinge loss
- Hinge loss: $\max(0, 1 - y_i(\vec{w} \cdot \vec{x}_i - b))$

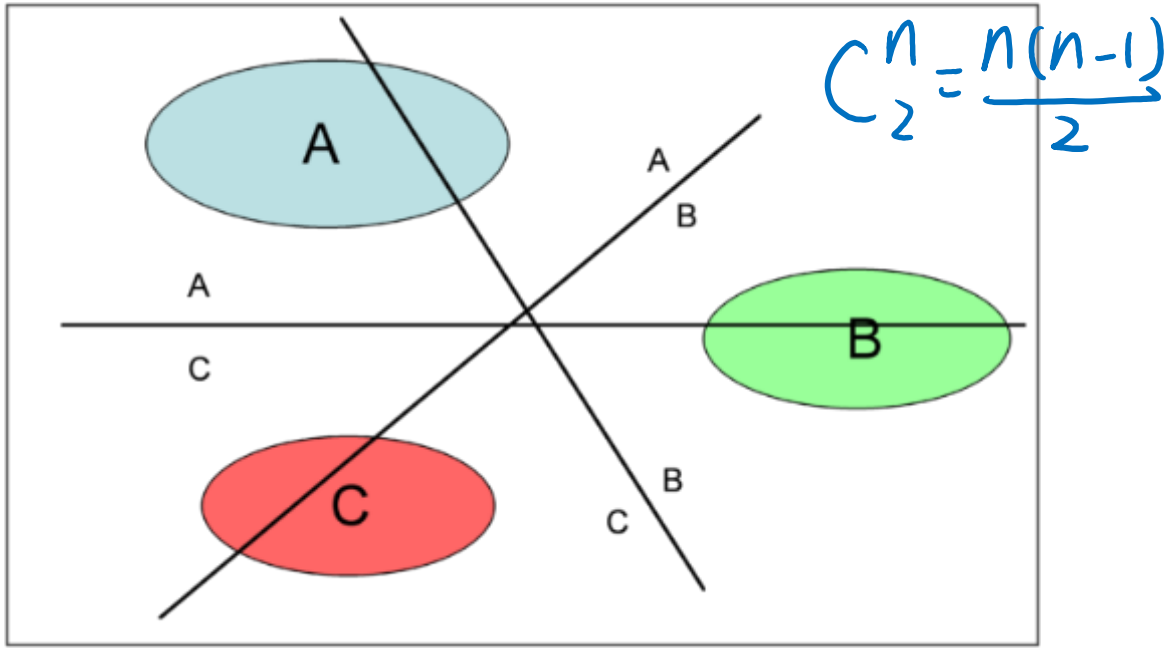
min. $\|w\|$

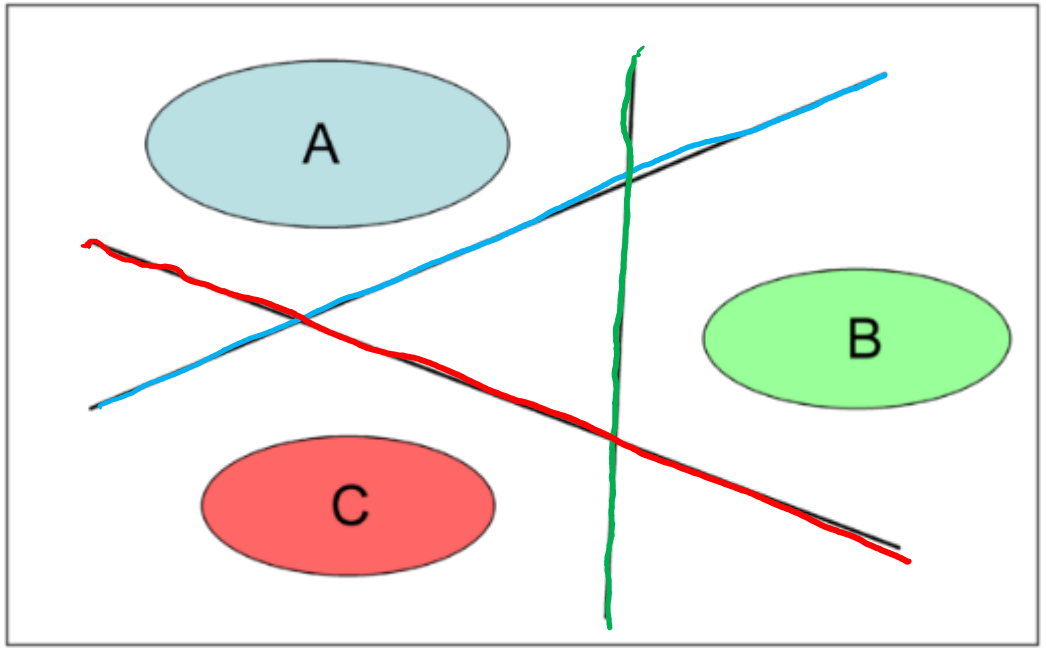
s.t. $\max(0, 1 - y_i(w^T x_i - b))$



Multi-class SVM

- One-against-One
- One-against-All

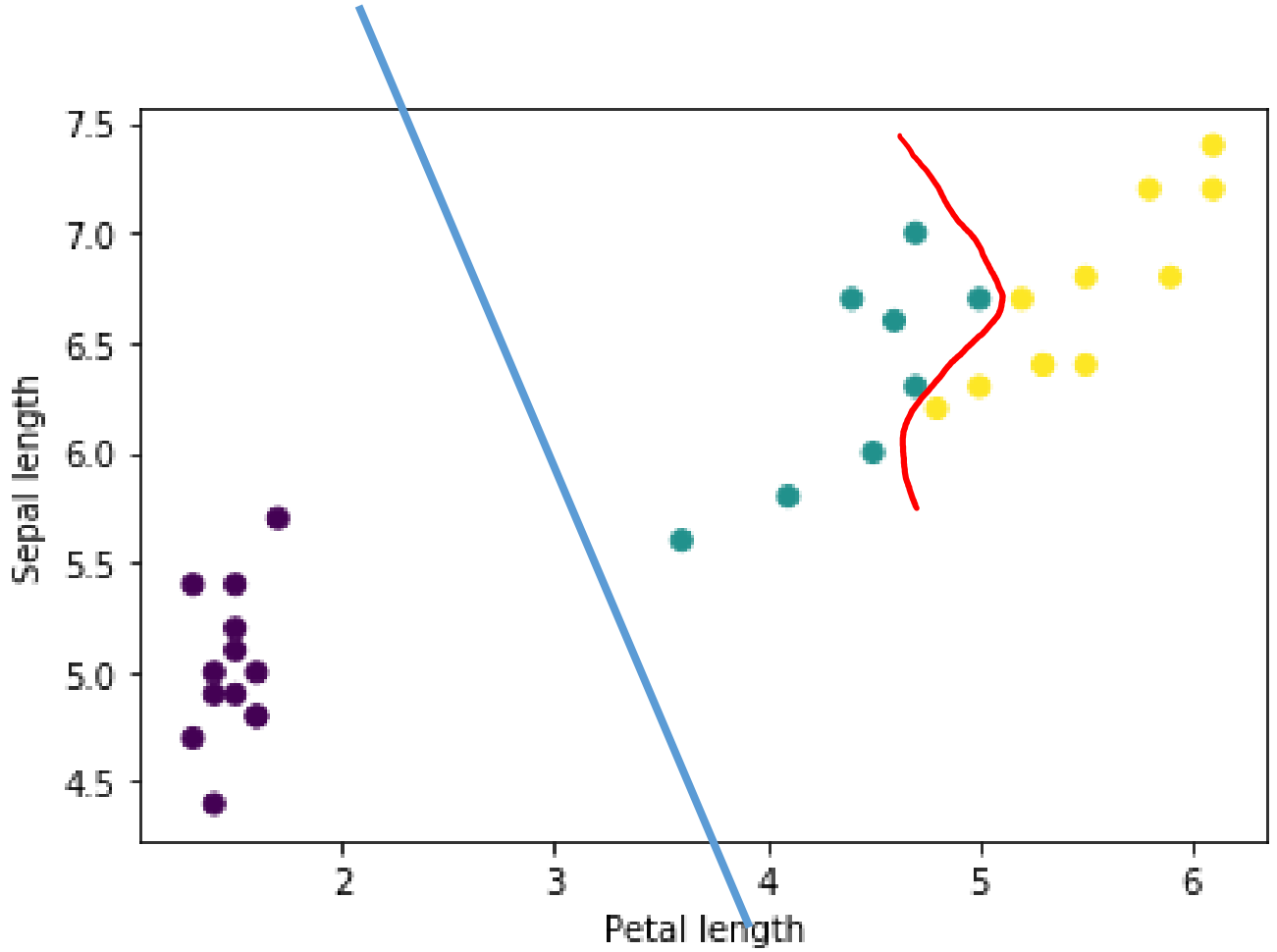




https://courses.media.mit.edu/2006fall/mas622j/Projects/aisen-project/

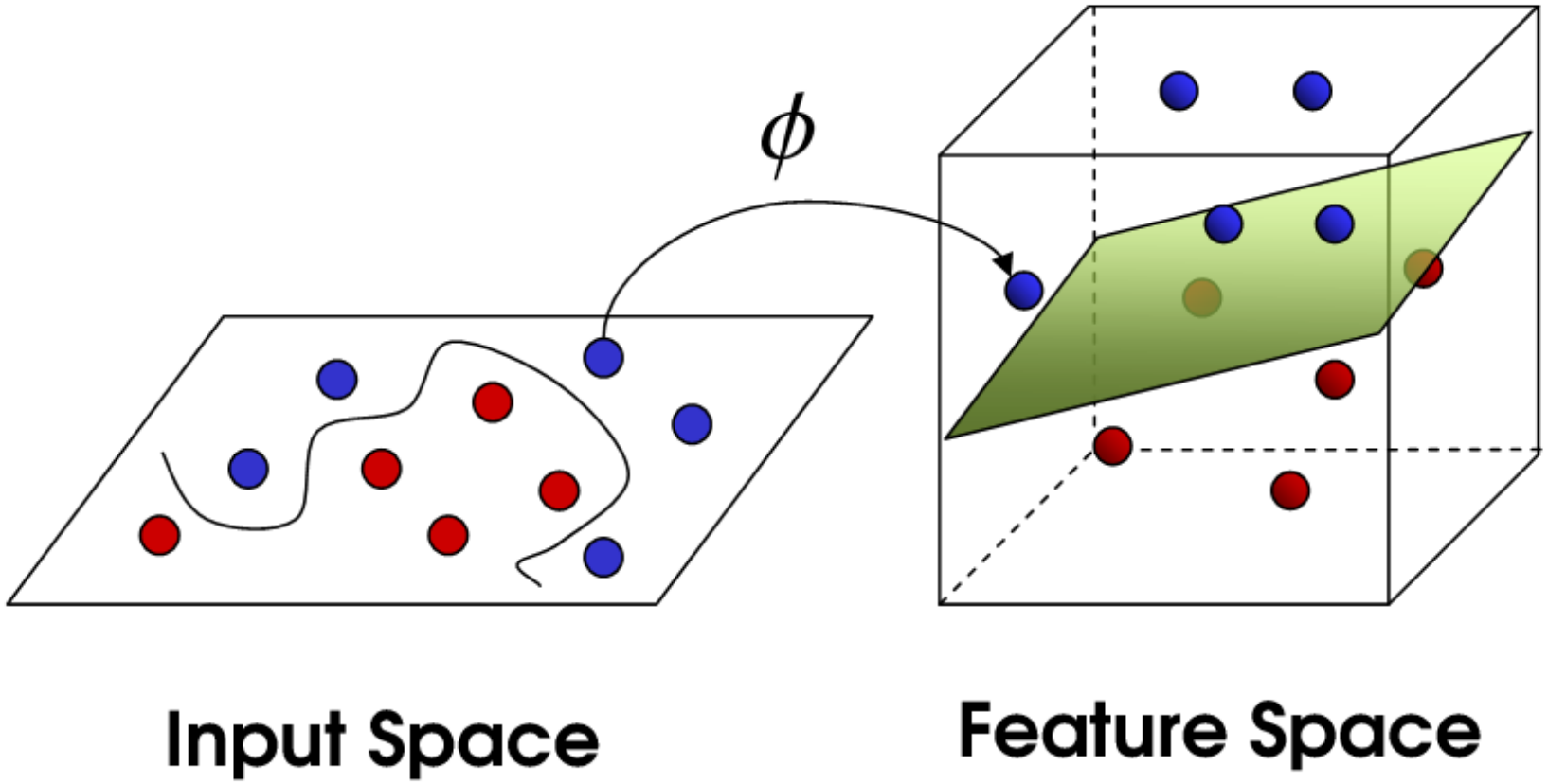
Nonlinear Problem?

- How to separate Versicolor and Virginica?



SVM Kernel Trick

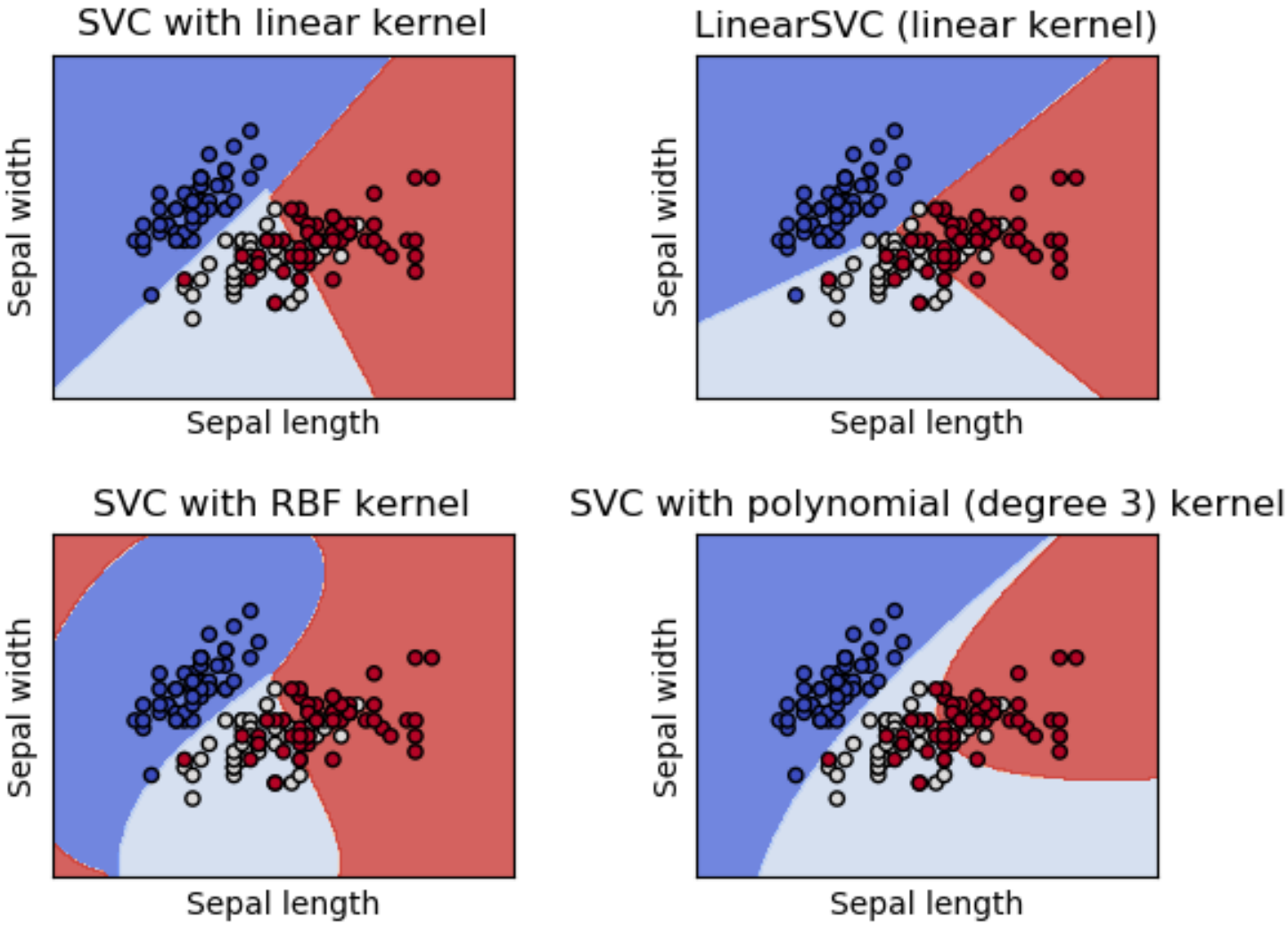
- Project data into higher dimension and calculate the inner products



https://datascience.stackexchange.com/questions/17536/kernel-trick-explanation

Nonlinear SVM for Iris Classification

Accuracy = 82.7%



Logistic Regression

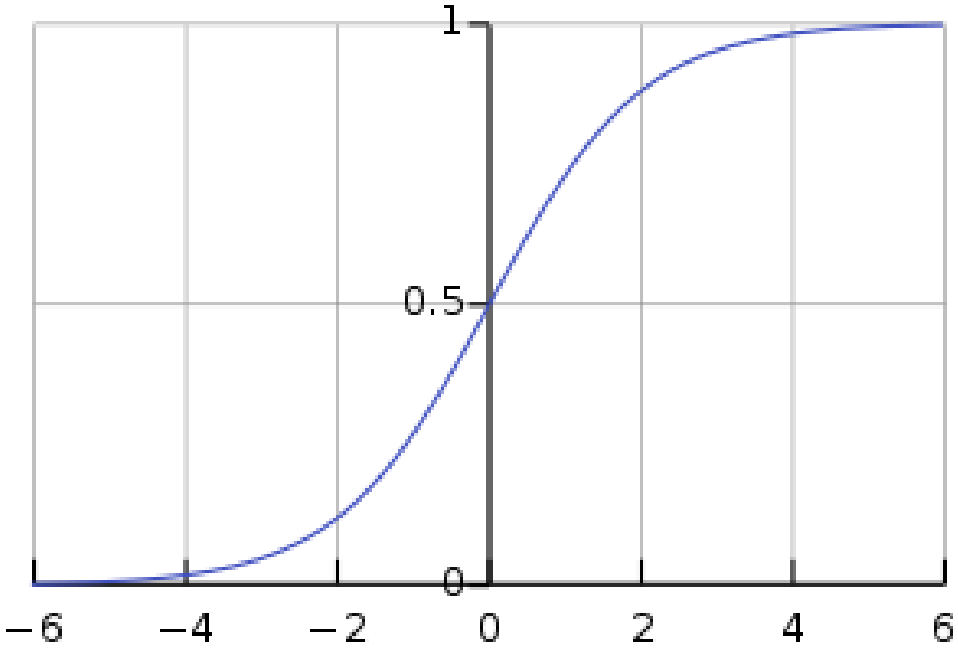
- Sigmoid function

$$S(x) = \frac{e^x}{e^x + 1} = \frac{1}{1 + e^{-x}}$$

- Derivative of Sigmoid

$$S(x) = S(x)(1 - S(x))$$

S-shaped curve



https://en.wikipedia.org/wiki/Sigmoid_function

Decision Boundary

- Binary classification with decision boundary t

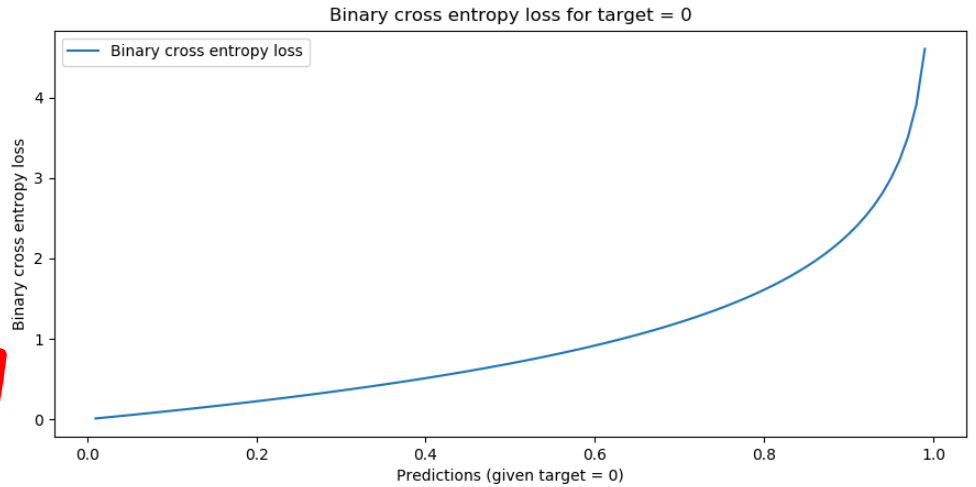
$$y' = P(x, w) = P_\theta(x) = \frac{1}{1 + e^{-(\boldsymbol{w}^T \boldsymbol{x} + b)}}$$

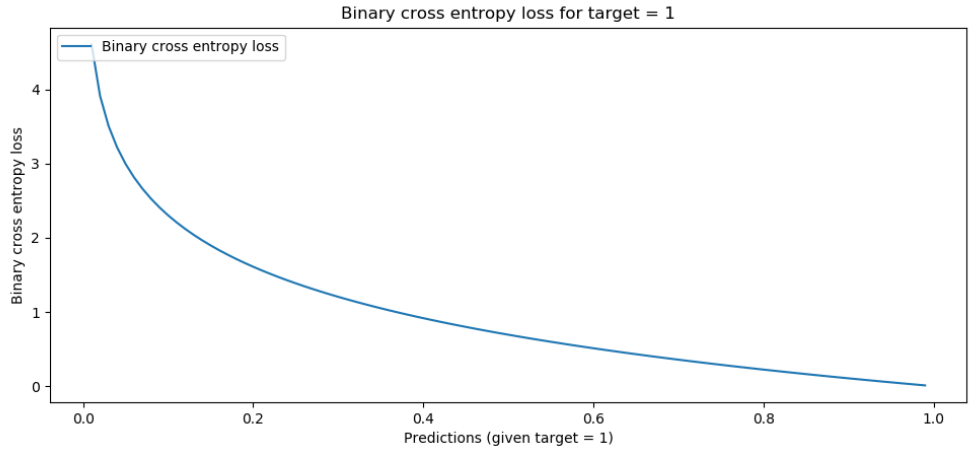
$$y' = \begin{cases} 0, & x < t \\ 1, & x \geq t \end{cases}$$

Cross Entropy Loss

- Loss function: cross entropy

$$\text{loss} = \begin{cases} -\log\left(1 - P_\theta(x)\right), & if\ y = 0 \\ -\log\left(P_\theta(x)\right), & if\ y = 1 \end{cases}$$





https://towardsdatascience.com/a-guide-to-neural-network-loss-functions-with-applications-in-keras-3a3baa9f71c5

Cross Entropy Loss

- Loss function: cross entropy

$$\text{loss}= \begin{cases} -\log\big(1 - P_\theta(x)\big), & if\ y = 0 \\ -\log\big(P_\theta(x)\big), & if\ y = 1 \end{cases}$$

$$\Rightarrow L_\theta(\mathrm{x}) = -y\log\big(P_\theta(x)\big) + -(1-\mathrm{y})\log\big(1 - P_\theta(x)\big)$$

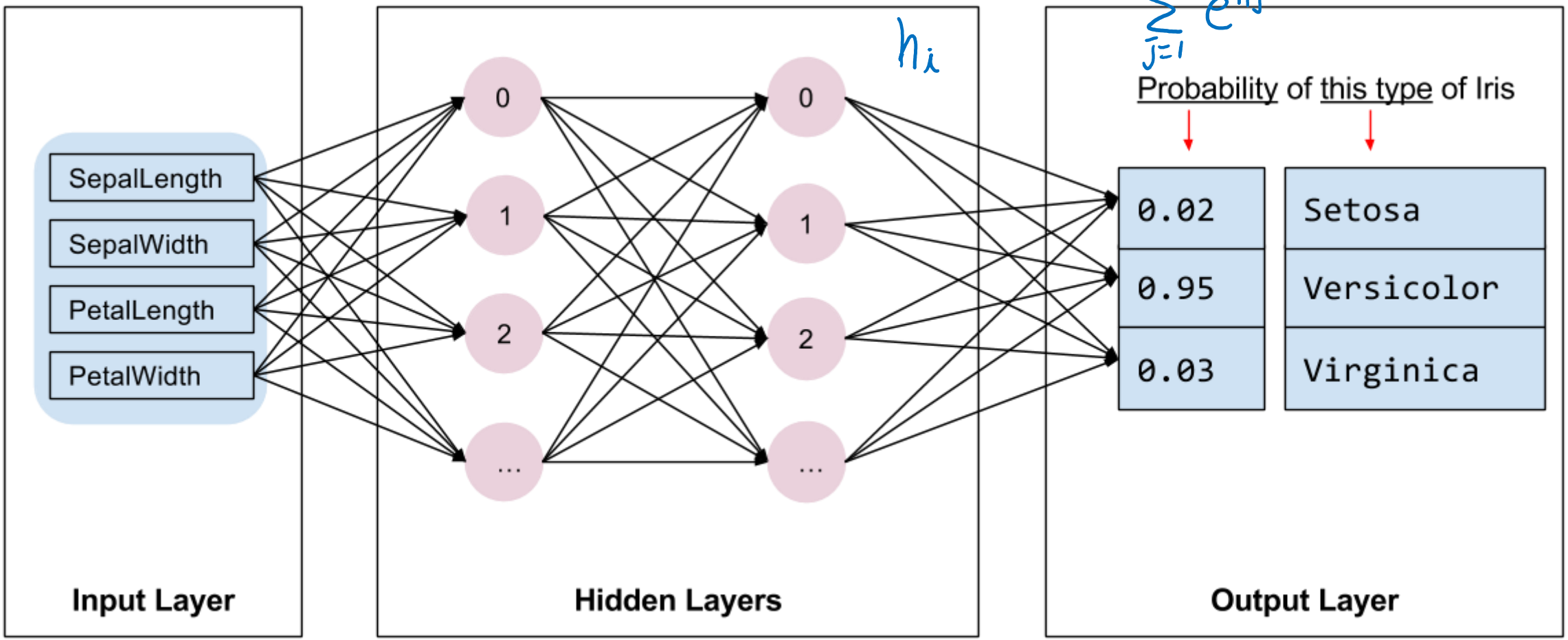
$$\nabla L_W(\mathrm{x}) = -\big(y - P_\theta(x)\big)x$$

y'

https://towardsdatascience.com/a-guide-to-neural-network-loss-functions-with-applications-in-keras-3a3baa9f71c5

Using Neural Network

$$\text{Softmax} \quad \frac{e^{h_i}}{\sum_{j=1}^{K} e^{h_j}}$$



https://www.tensorflow.org/tutorials/customization/custom_training_walkthrough

Evaluating All Classifiers on Iris Flower Dataset

Notebook Link

```
from sklearn.metrics import accuracy_score
from sklearn.linear_model import *
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn import datasets
from sklearn.naive_bayes import GaussianNB
```

```
[2] iris = datasets.load_iris()
    X = iris.data
    y = iris.target
    print('There are {} training samples in the Iris dataset'.format(len(X)))
```

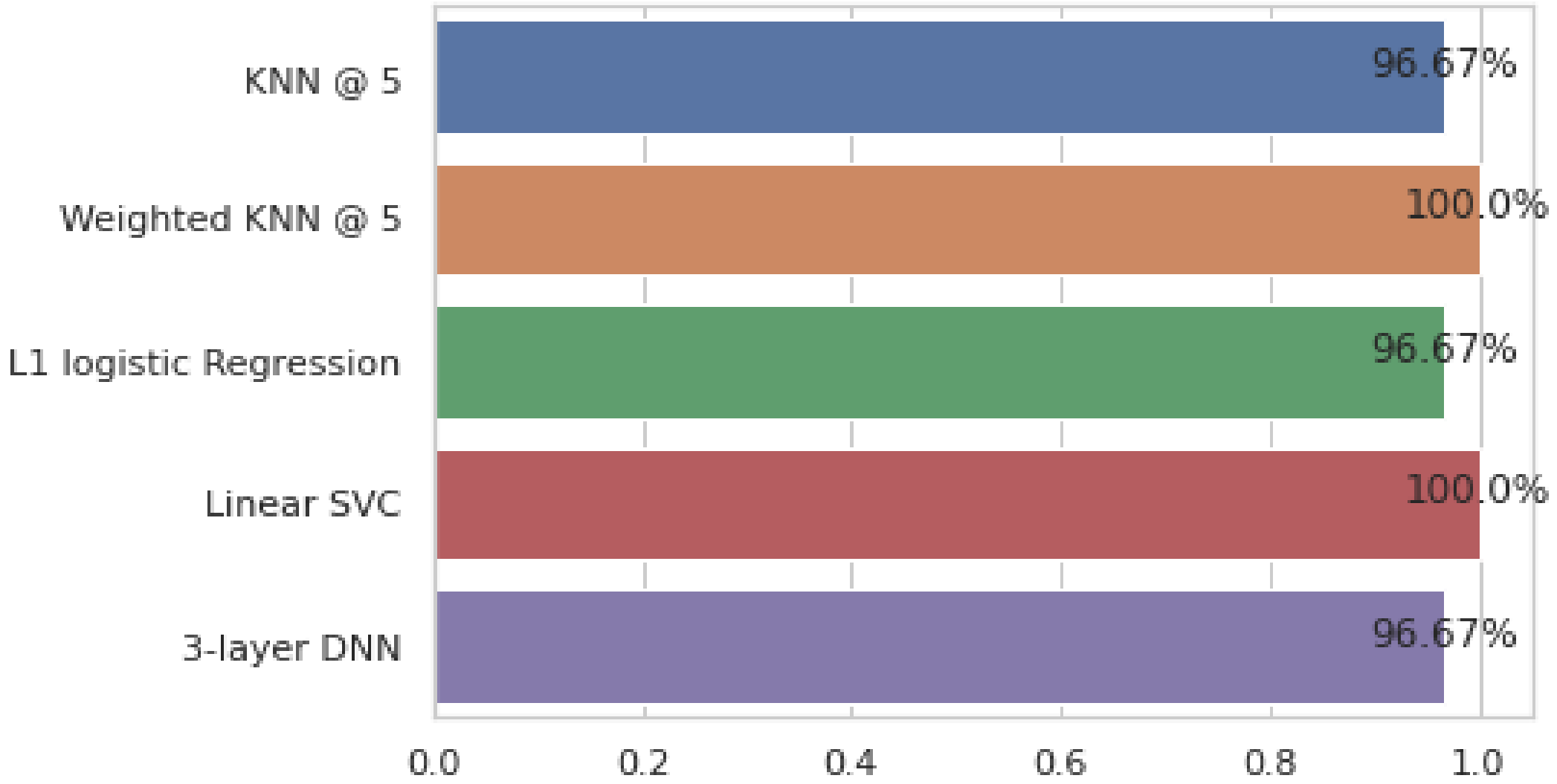
```
There are 150 training samples in the Iris dataset
```

```
[3] # Create different classifiers.
    knn = 5
    Cost = 10 # For regularization

    classifiers = {
            'KNN @ ' + str(knn): KNeighborsClassifier(knn, 'uniform'),
            'Weighted KNN @ ' + str(knn): KNeighborsClassifier(knn, 'distance'),
            'Perceptron': Perceptron(),
            'L1 logistic Regression': LogisticRegression(C=Cost, penalty='l1', solver='saga', multi_class='multinomial',max_iter=10000),
            'Linear SVC': SVC(kernel='linear', C=Cost, probability=False, random_state=0),
            'Naive Bayes': GaussianNB()
    }
```

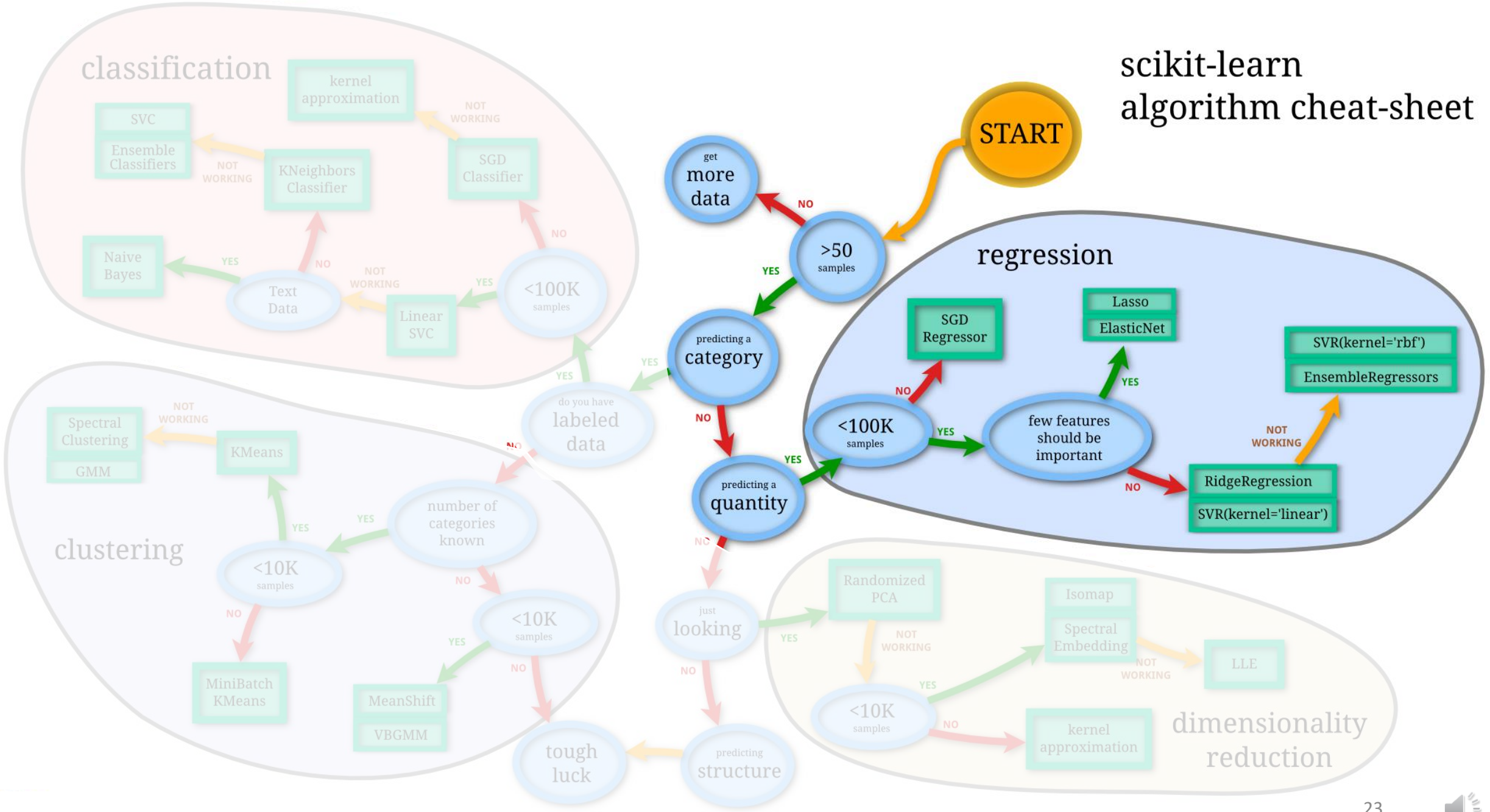
```
[4] # Training
    for index, (name, classifier) in enumerate(classifiers.items()):
            classifier.fit(X, y)

            y_pred = classifier.predict(X)
            accuracy = accuracy_score(y, y_pred)
            print("Accuracy (train) for %s: %0.1f%% " % (name, accuracy * 100))
```

Classifier Evaluation on Iris dataset

https://colab.research.google.com/drive/1CK7NFp6qX0XoGZWqryCDzdHKc3N4nD4J

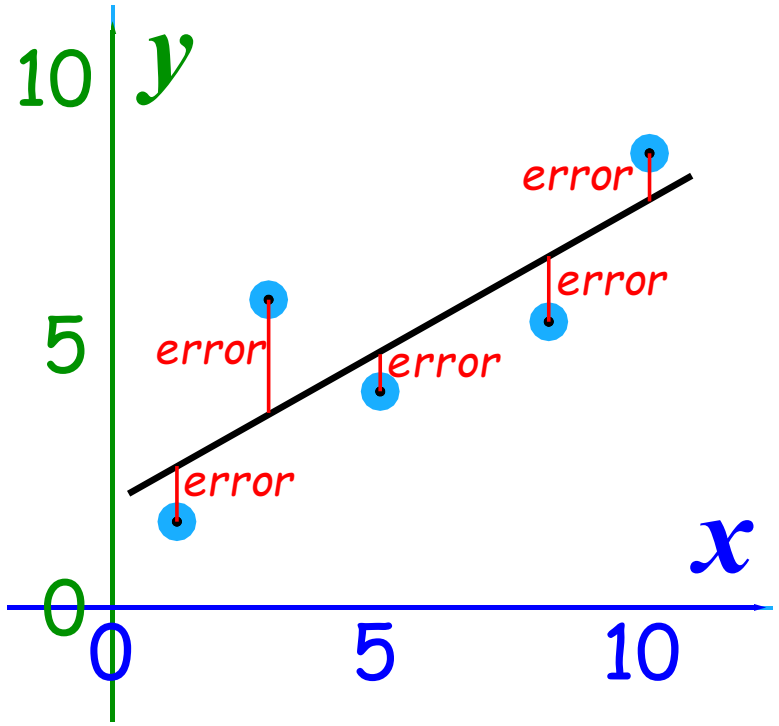


scikit-learn algorithm cheat-sheet



Linear Regression (Least squares)

- Find a "line of best fit“ that minimizes the total of the square of the errors



$$y' = W^T x + b$$

$$\sum (y_i - y_i')^2$$

$$y = X^T W + b$$

$$= [X \; 1] \begin{bmatrix} W \\ b \end{bmatrix}$$

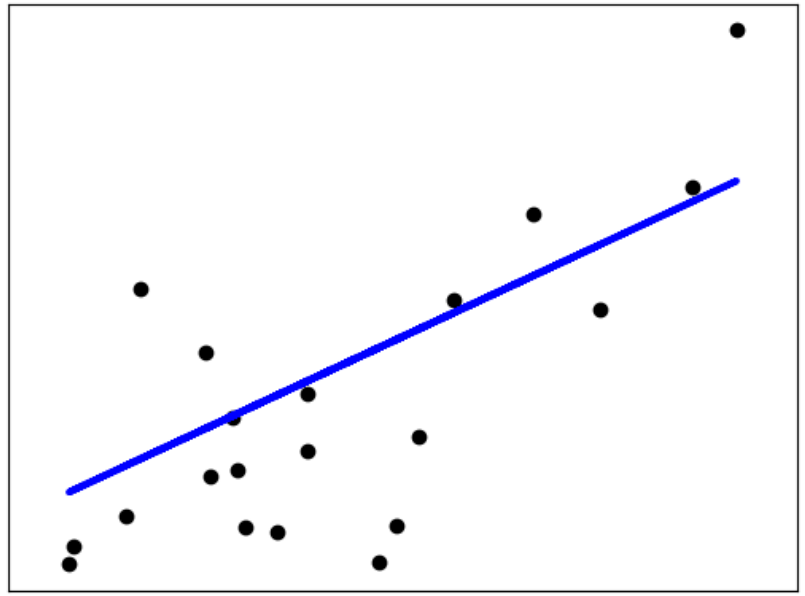
$$= X \cdot w$$

$$W = (X^T X)^{-1} X^T y$$

Scikit Learn Diabetes Dataset

- Ten baseline variables, age, sex, body mass index, average blood pressure, and six blood serum measurements were obtained for each of n = 442 diabetes patients

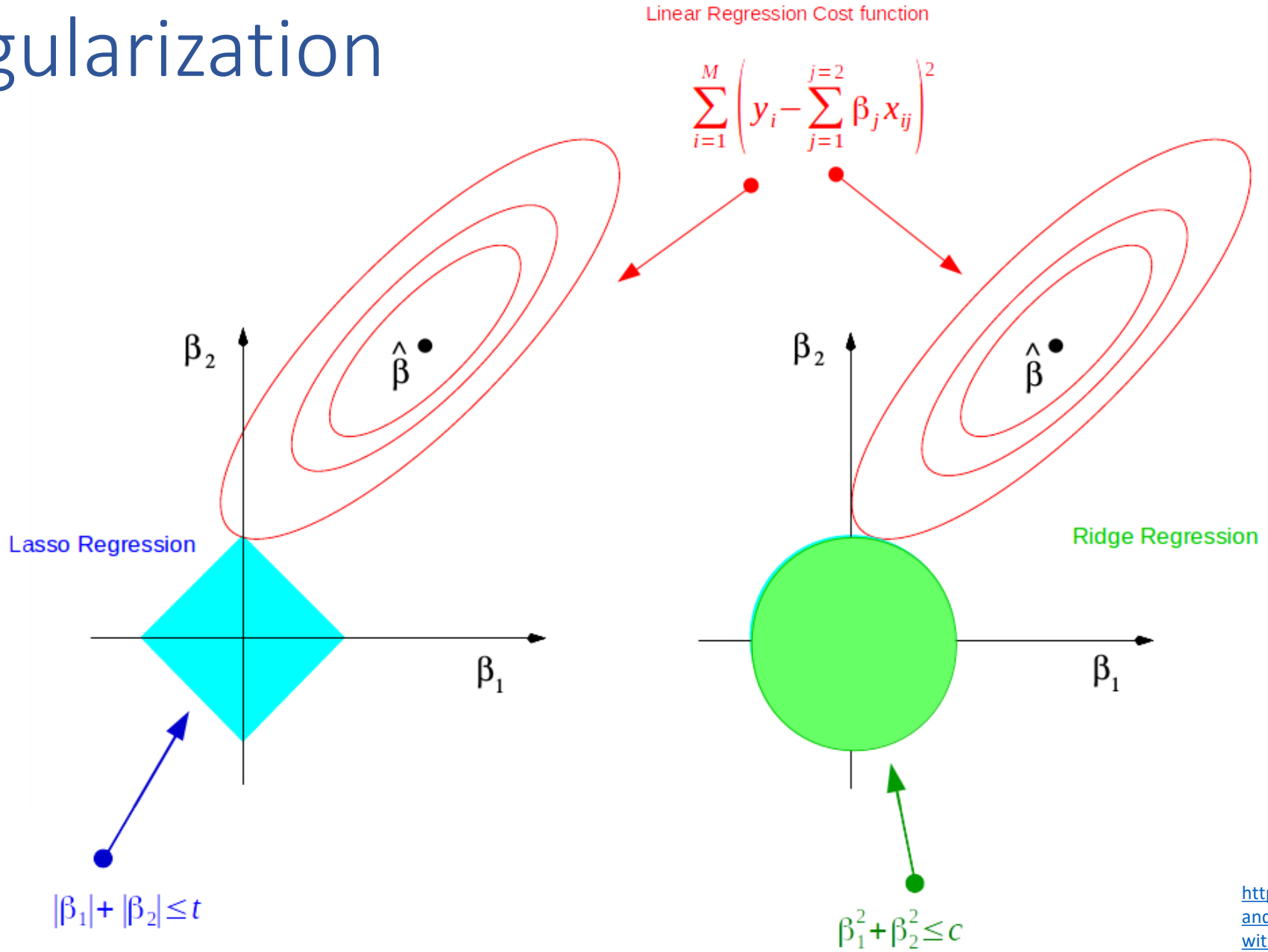
Samples total	442
Dimensionality	10
Features	real, -.2 < x < .2
Targets	integer 25 - 346



BMI

https://www4.stat.ncsu.edu/~boos/var.select/diabetes.html

Regularization



https://towardsdatascience.com/ridge-and-lasso-regression-a-complete-guide-with-python-scikit-learn-e20e34bcbf0b

Ridge, Lasso and ElasticNet

- Ridge regression: $\min_w ||Xw - y||_2^2 + \alpha||w||_2^2$
- Lasso regression: $\min_w \frac{1}{2n_{\text{samples}}}||Xw - y||_2^2 + \alpha||w||_1$
- Elastic Net: $\min_w \frac{1}{2n_{\text{samples}}}||Xw - y||_2^2 + \alpha\rho||w||_1 + \frac{\alpha(1-\rho)}{2}||w||_2^2$

Predicting Boston House Prices

Boston House Price Dataset

- Objective: predict the median price of homes
- Small dataset with 506 samples and 13 features
  - https://www.kaggle.com/c/boston-housing

1	crime	per capita crime rate by town.	8	dis	weighted mean of distances to five Boston employment centres.
2	zn	proportion of residential land zoned for lots over 25,000 sq.ft.	9	rad	index of accessibility to radial highways.
3	indus	proportion of non-retail business acres per town.	10	tax	full-value property-tax rate per $10,000.
4	chas	Charles River dummy variable (= 1 if tract bounds river; 0 otherwise).	11	ptratio	pupil-teacher ratio by town.
5	nox	nitrogen oxides concentration	12	black	1000(Bk - 0.63)^2 where Bk is the proportion of blacks by town.
6	rm	average number of rooms per dwelling.	13	lstat	lower status of the population (percent).
7	age	proportion of owner-occupied units built prior to 1940.			

Normalize the Data

- the feature is centered around 0 and has a unit standard deviation
- Note that the quantities (mean, std) used for normalizing the test data are computed using the training data!

```
# Nomalize the data
mean = train_data.mean(axis=0)
train_data -= mean
std = train_data.std(axis=0)
train_data /= std
test_data -= mean
test_data /= std
```


Regressor Comparison

+ 程式碼 + 文字 Notebook link...

```
[1] from sklearn.metrics import mean_squared_error, mean_absolute_error
    from sklearn.linear_model import *
    from sklearn import datasets
```

```
data = datasets.load_boston()
X = data.data
y = data.target
y_mean = y.mean()
print('There are {} training samples in the Boston House dataset. The average price is {}'.format(len(X), y_mean))
```

```
There are 506 training samples in the Boston House dataset. The average price is 22.532806324110677
```

```
[3] mean = X.mean(axis=0)
    X -= mean
    std = X.std(axis=0)
    X /= std
```

```
[4] regressors = {
        'Linear Regression': LinearRegression(),
        'Ridge Regression': Ridge(alpha=.5),
        'LASSO': Lasso(alpha=0.1),
        'ElasticNet': ElasticNet(alpha=0.1)
    }
```

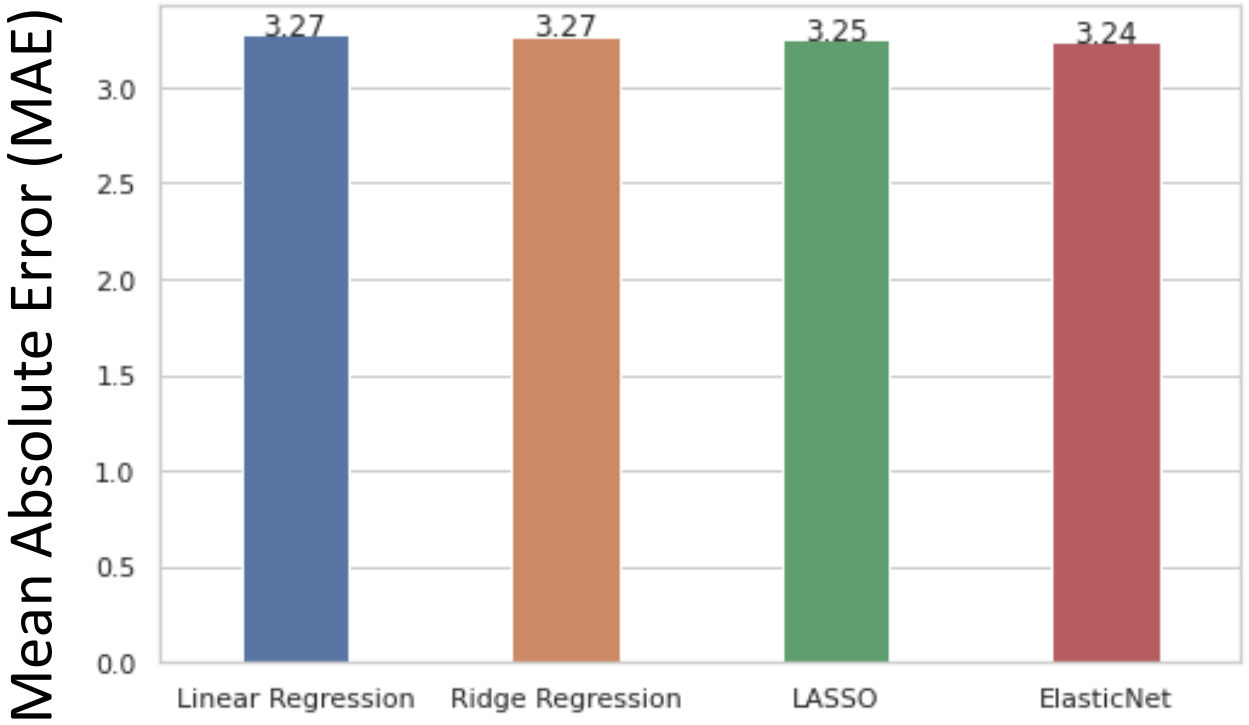
```
[5] # Training
    mae_list = []
    for index, (name, regressor) in enumerate(regressors.items()):
        regressor.fit(X, y)

        y_pred = regressor.predict(X)
        mse = mean_absolute_error(y, y_pred)
        print("The Mean Absolute Error for %s: %0.4f " % (name, mse))
        mae_list.append(mse)
```

```
The Mean Absolute Error for Linear Regression: 3.2709
The Mean Absolute Error for Ridge Regression: 3.2691
The Mean Absolute Error for LASSO: 3.2464
The Mean Absolute Error for ElasticNet: 3.2387
```

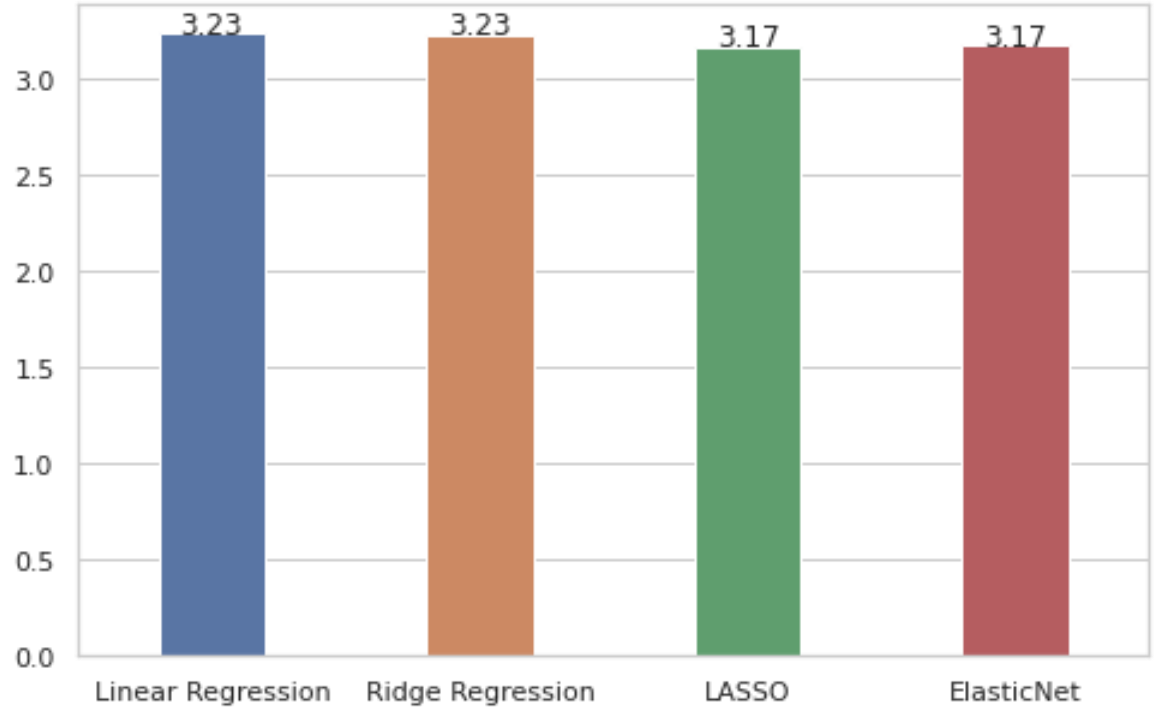
0 秒 完成時間：上午10:41

Comparison of Regularization Methods

Training Data (506 samples)



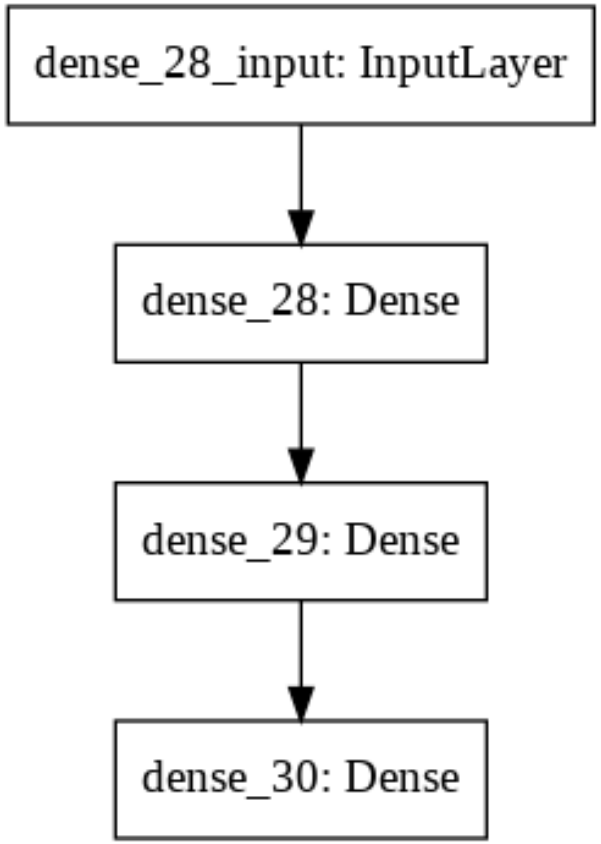
Test Data (102 samples)



https://colab.research.google.com/drive/1lgITg2vEmKfgqp7yDtrOCbWmtYuzRwIm

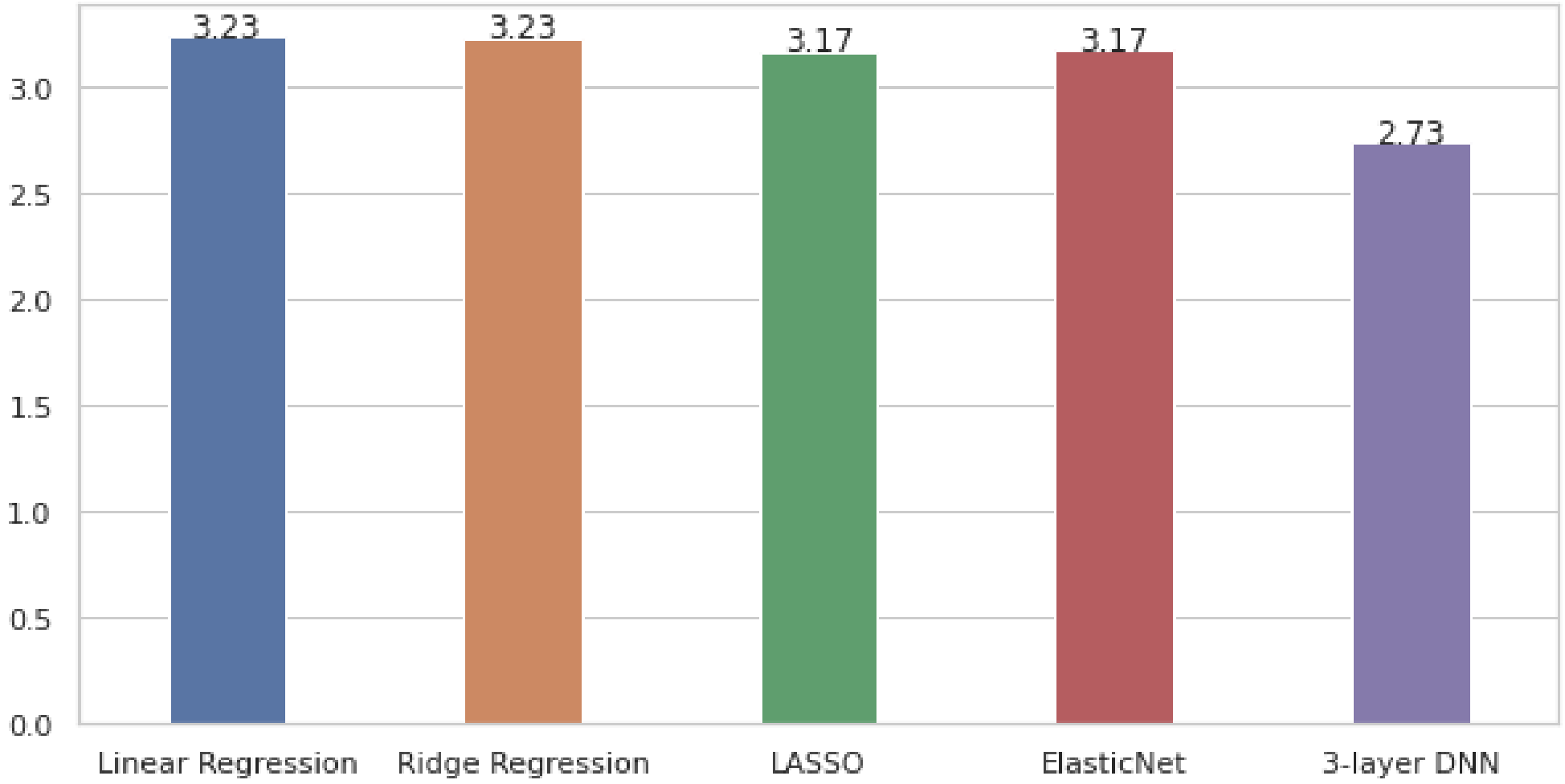
Predicting House Price using DNN

```
Model: "sequential_10"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_28 (Dense)             (None, 64)                896
_________________________________________________________________
dense_29 (Dense)             (None, 64)                4160
_________________________________________________________________
dense_30 (Dense)             (None, 1)                 65
=================================================================
Total params: 5,121
Trainable params: 5,121
Non-trainable params: 0
_________________________________________________________________
```



https://colab.research.google.com/drive/1tJztaaOIxbk_VuPKm8NpN7Cp_XABqyPQ

Final Results



References

- https://ml-cheatsheet.readthedocs.io/en/latest/index.html
- https://towardsdatascience.com/ridge-and-lasso-regression-a-complete-guide-with-python-scikit-learn-e20e34bcbf0b
- https://en.wikipedia.org/wiki/Naive_Bayes_classifier