

## Course Overview

We are now here:

- tools
- theory

Titanic Notebook

**Lectures: Quant Models and CS Code Examples**

**Brainstorm Challenge and Validate (4)**

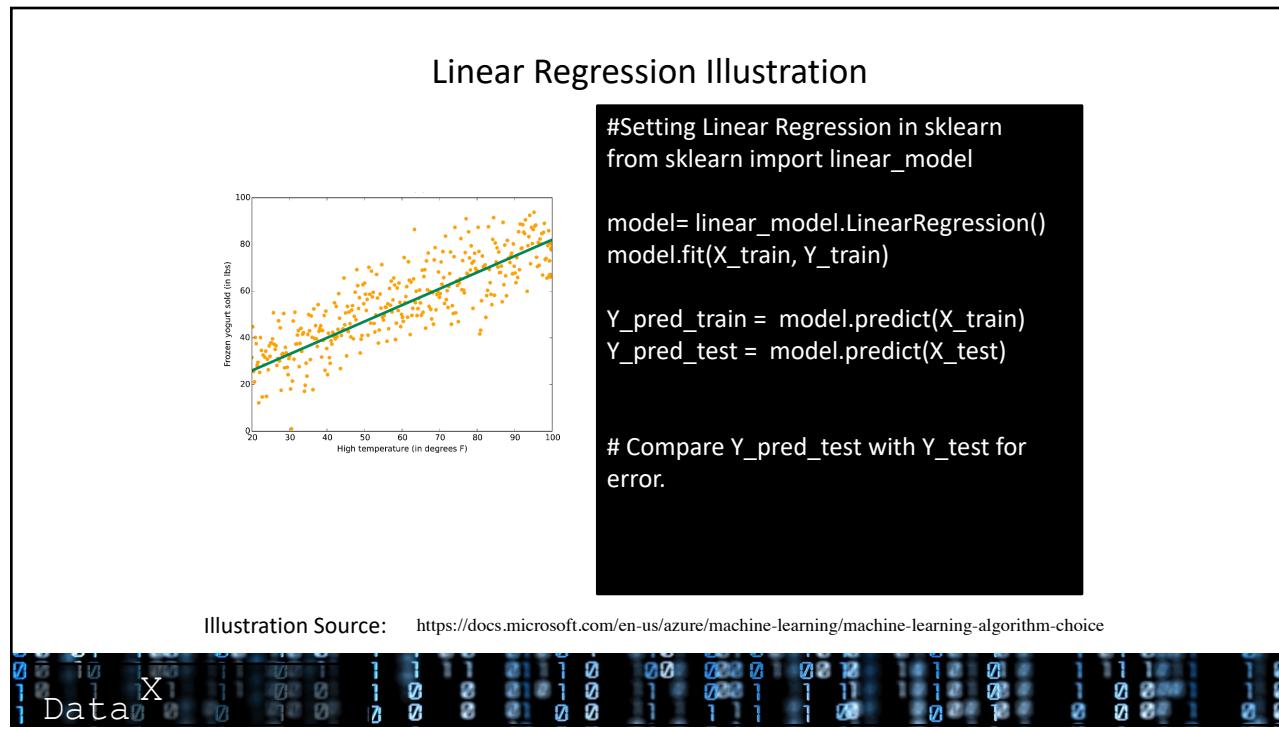
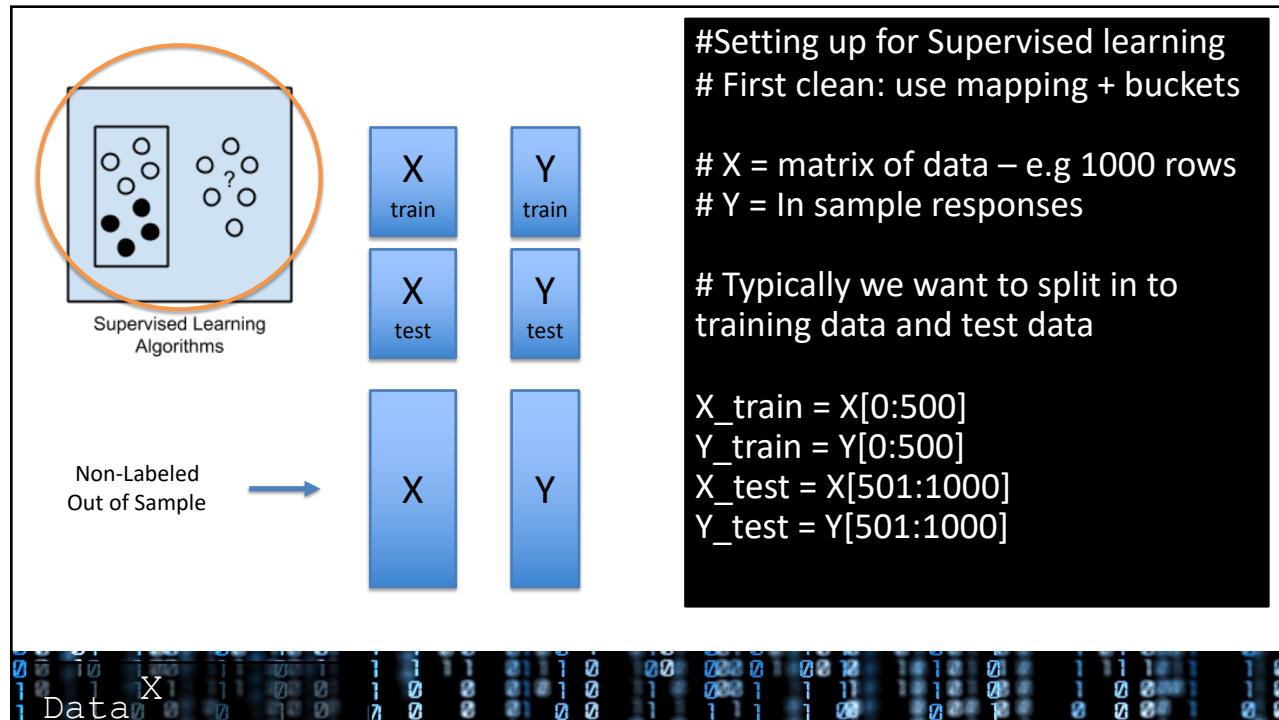
**Propose Low Tech Solution (1)**

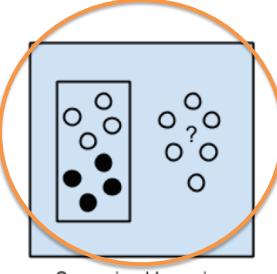
**Execute \* Iterate BMoE Reflections Agile Sprint (8)**

**Demo or Die (1)**

[Open-ended, real-world project:](#) Typically 5 students, with available advisor network







**Common Issue:**  
Do you have enough data to train and then test?

Small training set -> ?  
All training data -> ?

How to use the data efficiently?

|                     |       |    |       |
|---------------------|-------|----|-------|
| X                   | train | Y  | train |
| X                   | test  | Y  | test  |
| X<br>Non<br>Labeled |       | Y? |       |

```
#Setting up for Supervised learning
# First clean: use mapping + buckets

# X = matrix of data – e.g 1000 rows
# Y = In sample responses

# Typically we want to split in to
# training data and test data

X_train = X[0:500]
Y_train = Y[0:500]
X_test = X[501:1000]
Y_test = Y[501:1000]
```

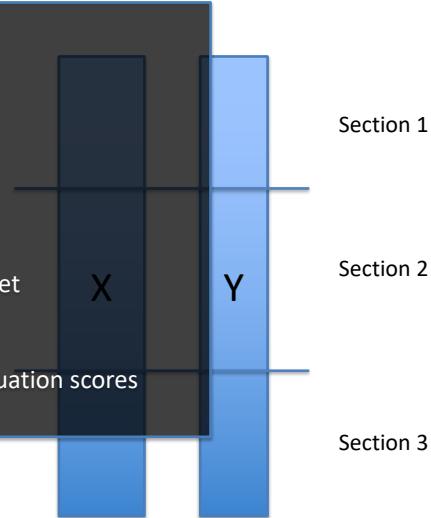


- Common Issue: Having enough data to train and test
- Cross Validation
- K-fold (ie 3-fold, 4-fold, ..)
- Example:
  - Train (1,2) -> Test with 3
  - Train (2,3) -> Test with 1
  - Train (1,3) -> Test with 2
  - Estimate model error as average of all 3

The general procedure is as follows:

Common issue: Having enough data to train and test

1. Shuffle the dataset randomly.
2. Split the dataset into k groups
3. For each unique group:
  - Take the group as a hold out or test data set
  - Take the remaining groups as a training data set
  - Fit a model on the training set and evaluate it on the test set
  - Retain the evaluation score and discard the model
4. Summarize the skill of the model using the sample of model evaluation scores
  - Train (1,2) -> Test with 3
  - Train (2,3) -> Test with 1
  - Train (1,3) -> Test with 2
  - Estimate model error as average of all 3



[A Gentle Introduction to k-fold Cross-Validation](#) by Jason Brownlee

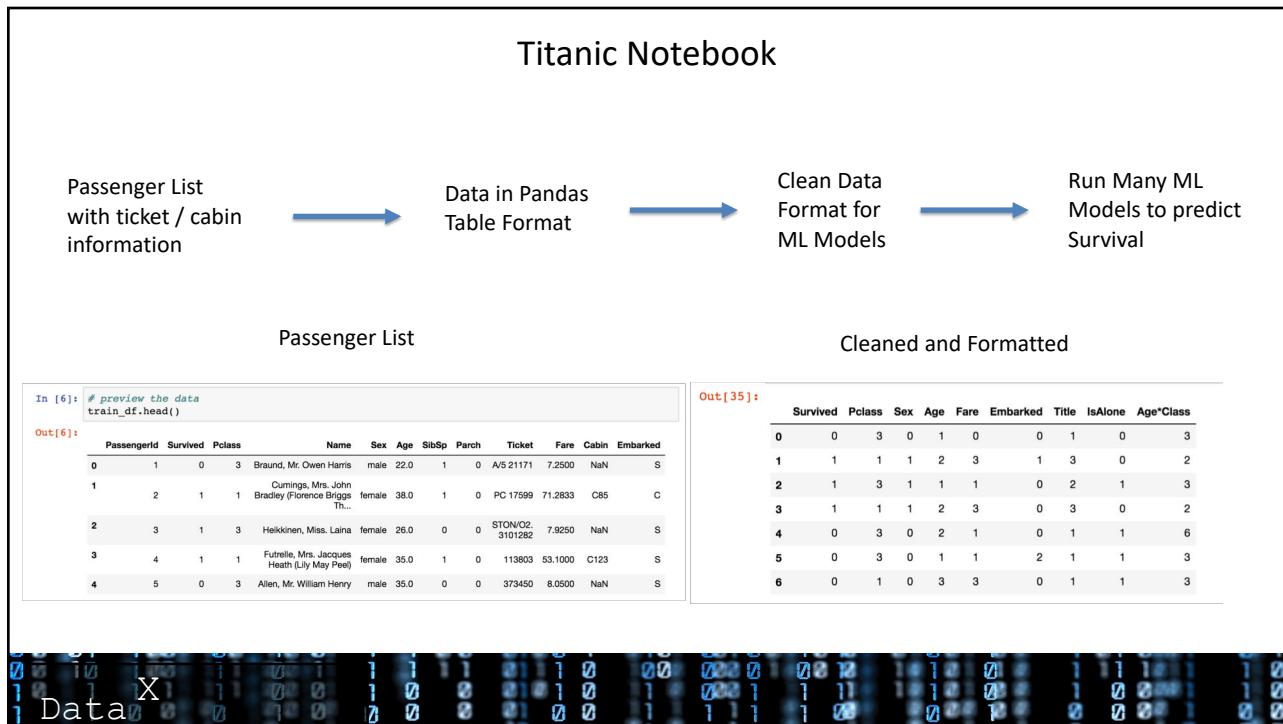
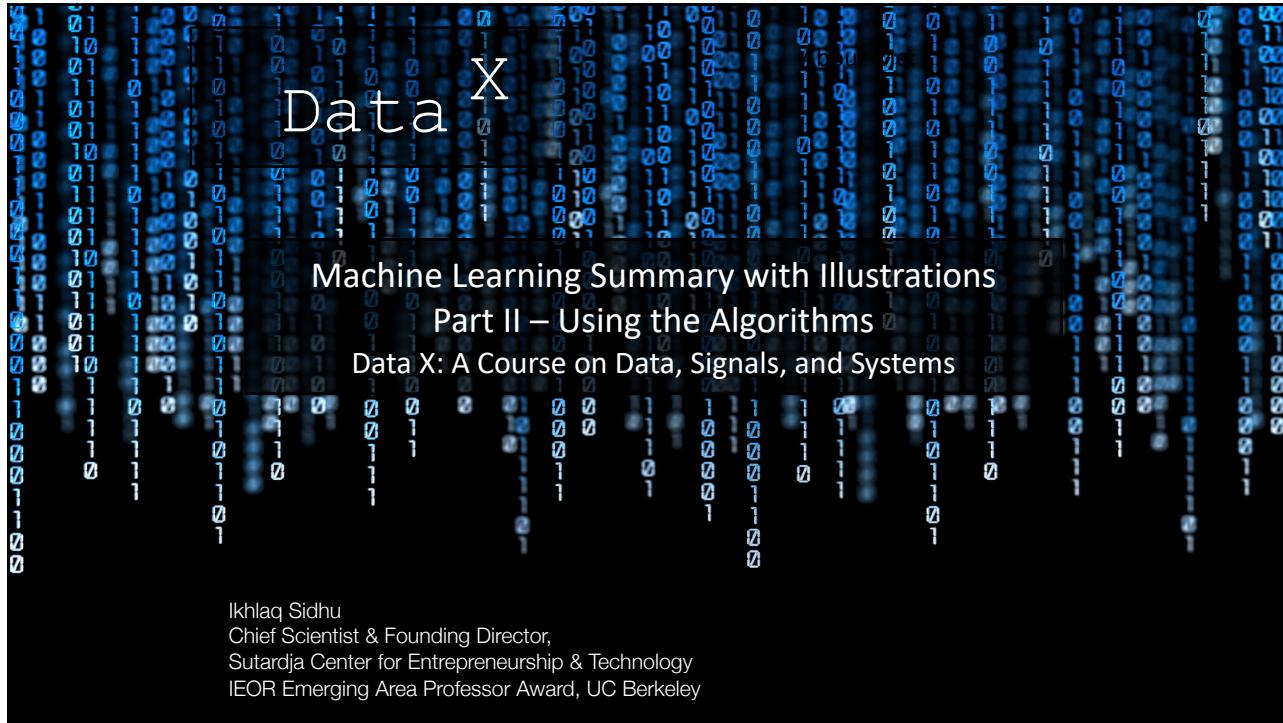


This Section:

- Context of the Titanic notebook
- Setting up data tables for training and testing ML Models
- Linear regression example in Scikit for prediction
- Cross validation (k-fold)

Next Section: ML Algorithms for Classification





## Our experiment with the Titanic Data Set

| Model                      | Score |
|----------------------------|-------|
| Random Forest              | 86.76 |
| Decision Tree              | 86.76 |
| KNN                        | 84.74 |
| Support Vector Machines    | 83.84 |
| Logistic Regression        | 80.36 |
| Linear SVC                 | 79.01 |
| Perceptron                 | 78.00 |
| Naive Bayes                | 72.28 |
| Stochastic Gradient Decent | 72.28 |

More Accuracy  
Generally more training time  
More risk of overfitting

Less Accuracy  
Generally less computation

Data X

## Logistic Regression Illustration

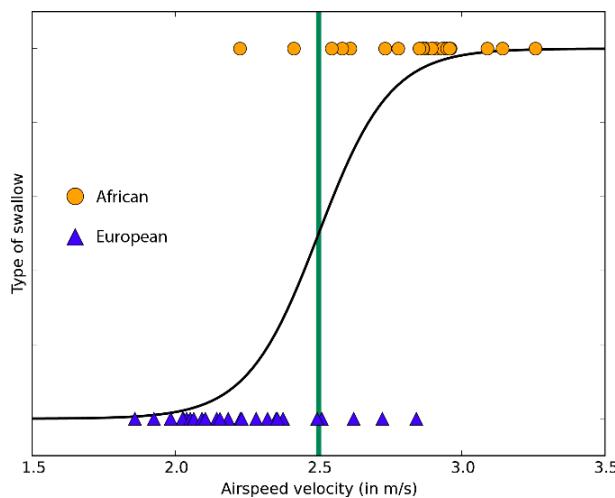
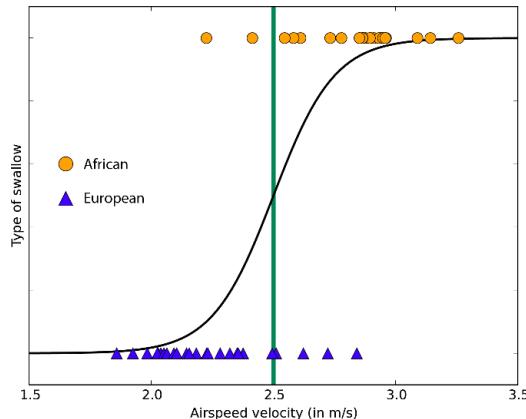


Illustration Source: <https://docs.microsoft.com/en-us/azure/machine-learning/machine-learning-algorithm-choice>

Data X

## Logistic Regression Illustration



```
from sklearn.linear_model import LogisticRegression

# Logistic Regression
logreg = LogisticRegression()
logreg.fit(X_train, Y_train)      #option for weights
Y_pred = logreg.predict(X_test)   #no options

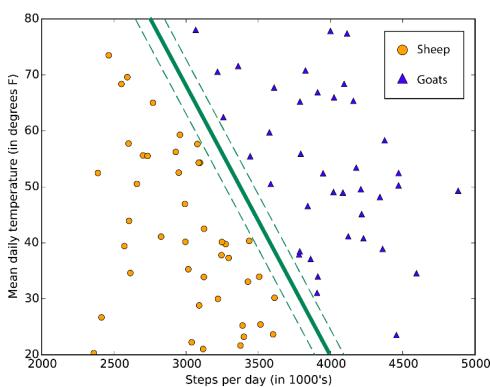
# Error
acc_log = round(logreg.score(X_train, Y_train) * 100, 2)
acc_log

# or compare Y_pred with Y_test
```

Illustration Source: <https://docs.microsoft.com/en-us/azure/machine-learning/machine-learning-algorithm-choice>



## Support Vector Machine (SVM) Illustration



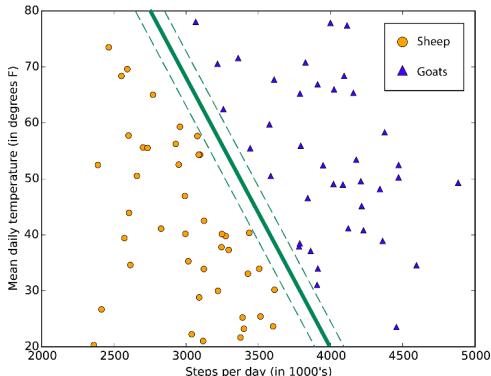
*A typical support vector machine class boundary maximizes the margin separating two classes*

Illustration Source:

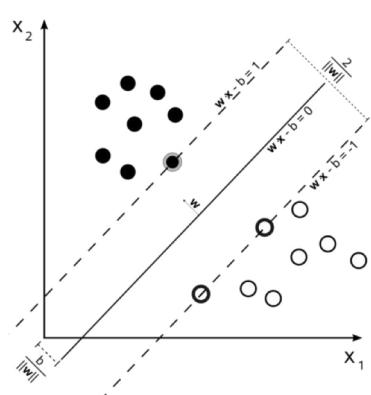
<https://docs.microsoft.com/en-us/azure/machine-learning/machine-learning-algorithm-choice>



## Support Vector Machine (SVM) Illustration



*A typical support vector machine class boundary maximizes the margin separating two classes*



$$\vec{w} \cdot \vec{x} - b = 0$$

$$\vec{w} \cdot \vec{x} - b = 1$$

and

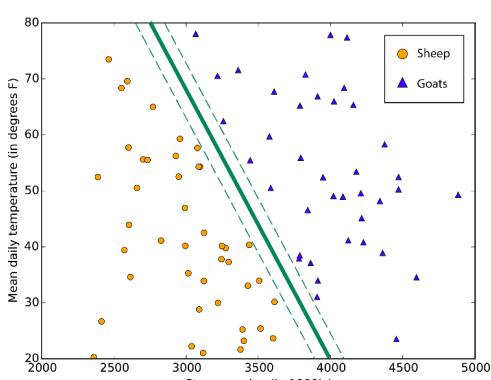
$$\vec{w} \cdot \vec{x} - b = -1.$$

$\frac{b}{\|w\|}$  determines the offset

Illustration Source:



## Support Vector Machine (SVM) Illustration



*A typical support vector machine class boundary maximizes the margin separating two classes*

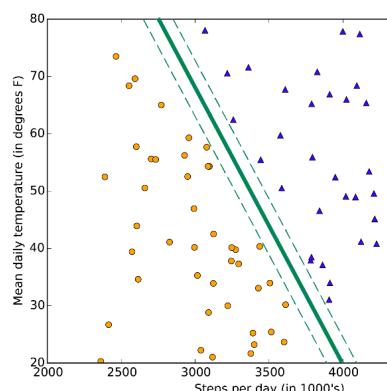
### SVM Considerations:

1. Robust
2. Effective in high dimension
3. Even when data rows < feature dimensions
4. Overfitting is possible, regularization is often needed
5. To predict for sparse data, must train with sparse data

Illustration Source:



## Support Vector Machine (SVM) Illustration



```
from sklearn.svm import SVC, LinearSVC
svc = SVC()
svc.fit(X_train, Y_train)
Y_pred = svc.predict(X_test)

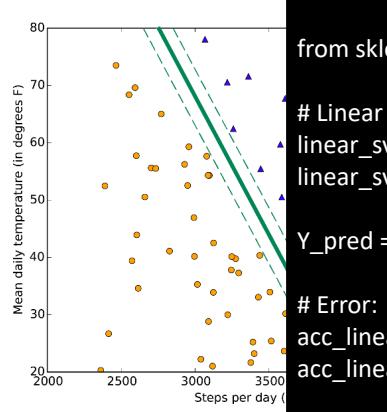
# Error
acc_svc = round(svc.score(X_train, Y_train) * 100, 2)
acc_svc

# or compare Y_pred with Y_test
```

Illustration Source: <https://docs.microsoft.com/en-us/azure/machine-learning/machine-learning-algorithm-choice>



## Support Vector Machine (SVM) Illustration



```
from sklearn.svm import SVC, LinearSVC
# Linear SVC
linear_svc = LinearSVC()
linear_svc.fit(X_train, Y_train)

Y_pred = linear_svc.predict(X_test)

# Error:
acc_linear_svc = round(linear_svc.score(X_train, Y_train) * 100, 2)
acc_linear_svc

# or compare Y_pred with Y_test
```

Illustration Source: <https://docs.microsoft.com/en-us/azure/machine-learning/machine-learning-algorithm-choice>



**KNN Method:** Find the k nearest images and have them vote on the label (i.e. take the mode)

| Colour | Water | Rock |
|--------|-------|------|
| Red    | 109   | 24   |
| Green  | 112   | 14   |
| Blue   | 105   | 13   |
| Red    | 137   | 15   |
| Green  | 164   | 11   |
| Blue   | 125   | 1    |
| Red    | 179   | 24   |
|        | 209   | 20   |
|        | 177   | 13   |
|        | 136   | 17   |
|        | 119   | 7    |
| ?      | 107   | 0    |

**KNN / K Means Illustration**

**the data**

**NN classifier**

**5-NN classifier**

**Data**

**KNN Method:** Find the k nearest images and have them vote on the label (i.e. take the mode)

| Colour | Water | Rock |
|--------|-------|------|
| Red    | 109   | 24   |
| Green  | 112   | 14   |
| Blue   | 105   | 13   |
| Red    | 137   | 15   |
| Green  | 164   | 11   |
| Blue   | 125   | 1    |
| Red    | 179   | 24   |
|        | 209   | 20   |
|        | 177   | 13   |
|        | 136   | 17   |
|        | 119   | 7    |
| ?      | 107   | 0    |

**KNN / K Means Illustration**

**the data**

**NN classifier**

**5-NN classifier**

Example of  $k$ -NN classification. The test sample (green circle) should be classified either to the first class of blue squares or to the second class of red triangles. If  $k = 3$  (solid line circle) it is assigned to the second class because there are 2 triangles and only 1 square inside the inner circle. If  $k = 5$  (dashed line circle) it is assigned to the first class (3 squares vs. 2 triangles inside the outer circle). - Wikipedia

<https://docs.microsoft.com/en-us/azure/machine-learning/machine-learning-algorithm-choice>

**Data**

**KNN / K Means Illustration**

**KNN Method:** Find the k nearest images and have them vote on the label (i.e. take the mode)

| Colour | Water | Rock |
|--------|-------|------|
| Red    | 109   | 24   |
| Green  | 112   | 14   |
| Blue   | 105   | 13   |
| Red    | 137   | 15   |
| Green  | 164   | 11   |
| Blue   | 125   | 1    |
| Red    | 179   | 24   |
|        | 209   | 20   |
|        | 177   | 13   |
|        | 136   | 17   |
|        | 119   | 7    |
| ?      | 107   | 0    |

**the data** **NN classifier** **5-NN classifier**

Example of k-NN classification. The test sample (green circle) should be classified either to the first class of blue squares or to the second class of red triangles. If  $k = 3$  (solid line circle) it is assigned to the second class because there are 2 triangles and only 1 square inside the inner circle. If  $k = 5$  (dashed line circle) it is assigned to the first class (3 squares vs. 2 triangles inside the outer circle). - Wikipedia

<https://docs.microsoft.com/en-us/azure/machine-learning/machine-learning-algorithm-choice>

**K-means (data is not labeled)**

**Data**

**K Means / KNN Illustration**

**the data** **NN classifier** **5-NN classifier**

**KNN Method:** Find the k nearest images and have them vote on the label (i.e. take the mode)

```

from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors = 3)
knn.fit(X_train, Y_train)
Y_pred = knn.predict(X_test)

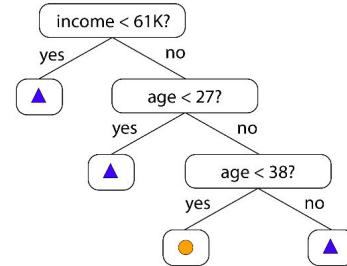
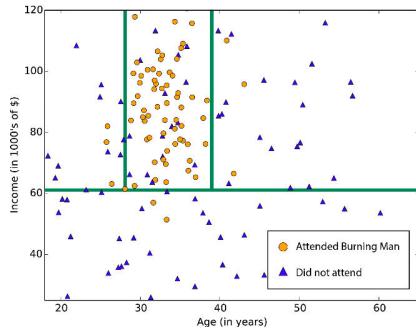
acc_knn = round(knn.score(X_train, Y_train) * 100, 2)
acc_knn
# or compare Y_pred with Y_test

```

Illustration Source: <https://docs.microsoft.com/en-us/azure/machine-learning/machine-learning-algorithm-choice>

**Data**

# Decision Tree Illustration



| Person | F1(>61K) | F2 (<27y) ... | Y  |
|--------|----------|---------------|----|
| A      | 1        | 0             | 0  |
| B      | 0        | 1             | 1  |
| C      | 0        | 0             | 0  |
| ..     | ..       | ..            | .. |

- Can be implemented in logic
  - Complexity is in training
  - Order of decisions matters for speed and accuracy

**Illustration Source:** <https://docs.microsoft.com/en-us/azure/machine-learning/machine-learning-algorithm-choice>



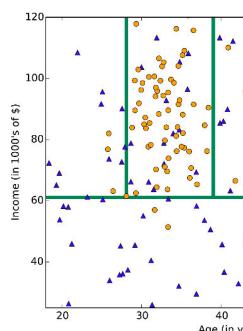
## Decision Tree Illustration

```
from sklearn import tree
```

```
decision_tree = DecisionTreeClassifier()
decision_tree.fit(X_train, Y_train)
Y_pred = decision_tree.predict(X_test)

# Error
acc_decision_tree = round(decision_tree.score(X_train, Y_train) * 100, 2)
acc_decision_tree

# or compare Y_pred with Y_test
```



## Illustration Source:



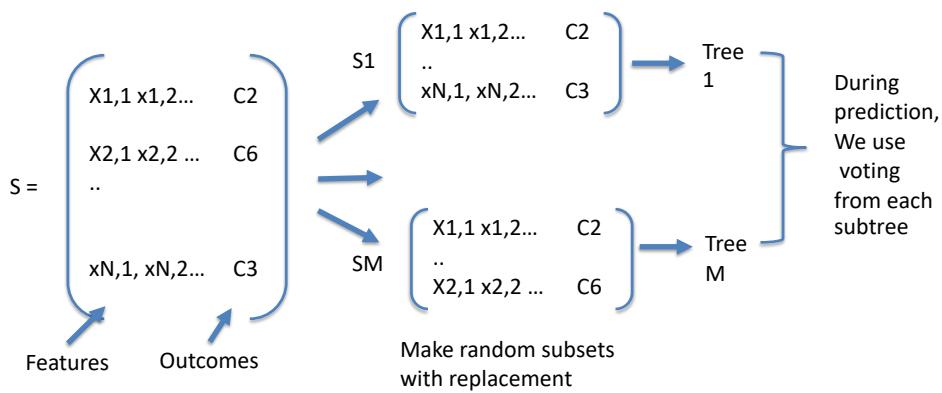
# Our experiment with the Titanic Data Set

| Model                      | Score |
|----------------------------|-------|
| Random Forest              | 86.76 |
| Decision Tree              | 86.76 |
| KNN                        | 84.74 |
| Support Vector Machines    | 83.84 |
| Logistic Regression        | 80.36 |
| Linear SVC                 | 79.01 |
| Perceptron                 | 78.00 |
| Naive Bayes                | 72.28 |
| Stochastic Gradient Decent | 72.28 |

- More Accuracy
- Generally more training time
- More risk of overfitting

Less Accuracy  
Generally less computation

## Random Forest – A type of bagging/ensemble approach



Advantages: One of most accurate  
Efficient prediction over large data

Disadvantages: Overfit and Training time

Data X

# Trees Can be Extended with Bagging

Explain  
bagging a  
Random  
Forrest

```
from sklearn.ensemble import RandomForestClassifier

random_forest =
RandomForestClassifier(n_estimators=1000)
random_forest.fit(X_train, Y_train)
Y_pred = random_forest.predict(X_test)
random_forest.score(X_train, Y_train)

# Error
acc_random_forest = round(random_forest.score(X_train,
Y_train) * 100, 2)
acc_random_forest

# or compare Y_pred with Y_test
```

## Our experiment with the Titanic Data Set

| Model                      | Score |
|----------------------------|-------|
| Random Forest              | 86.76 |
| Decision Tree              | 86.76 |
| KNN                        | 84.74 |
| Support Vector Machines    | 83.84 |
| Logistic Regression        | 80.36 |
| Linear SVC                 | 79.01 |
| Perceptron                 | 78.00 |
| Naive Bayes                | 72.28 |
| Stochastic Gradient Decent | 72.28 |

More Accuracy  
Generally more training time  
More risk of overfitting

Less Accuracy  
Generally less computation

Data X

## Neural Network Illustration

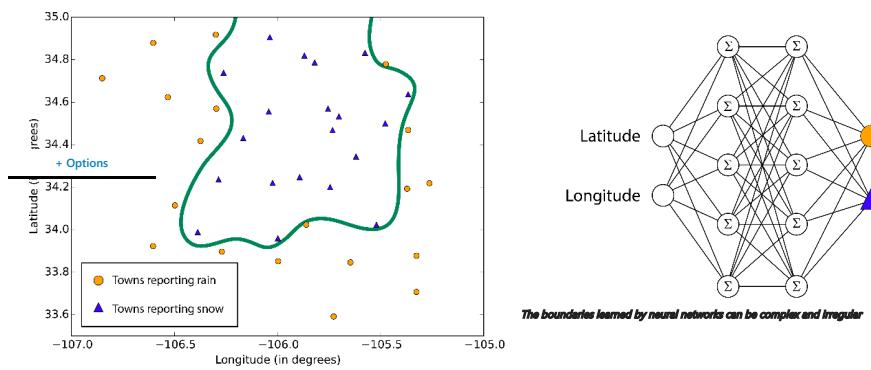
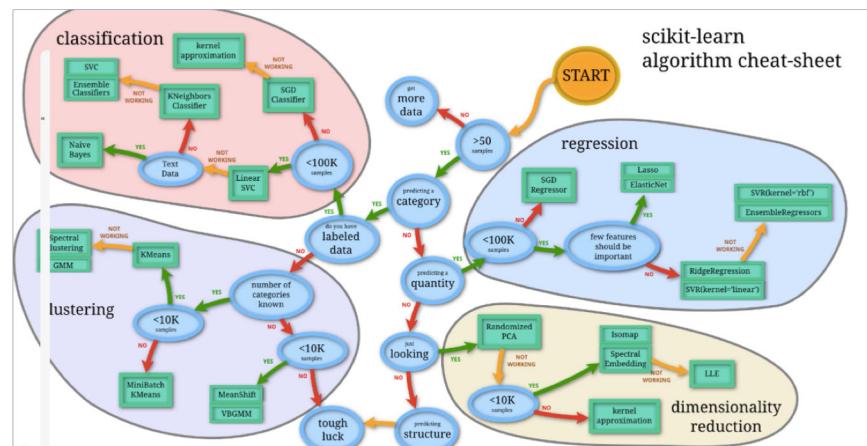
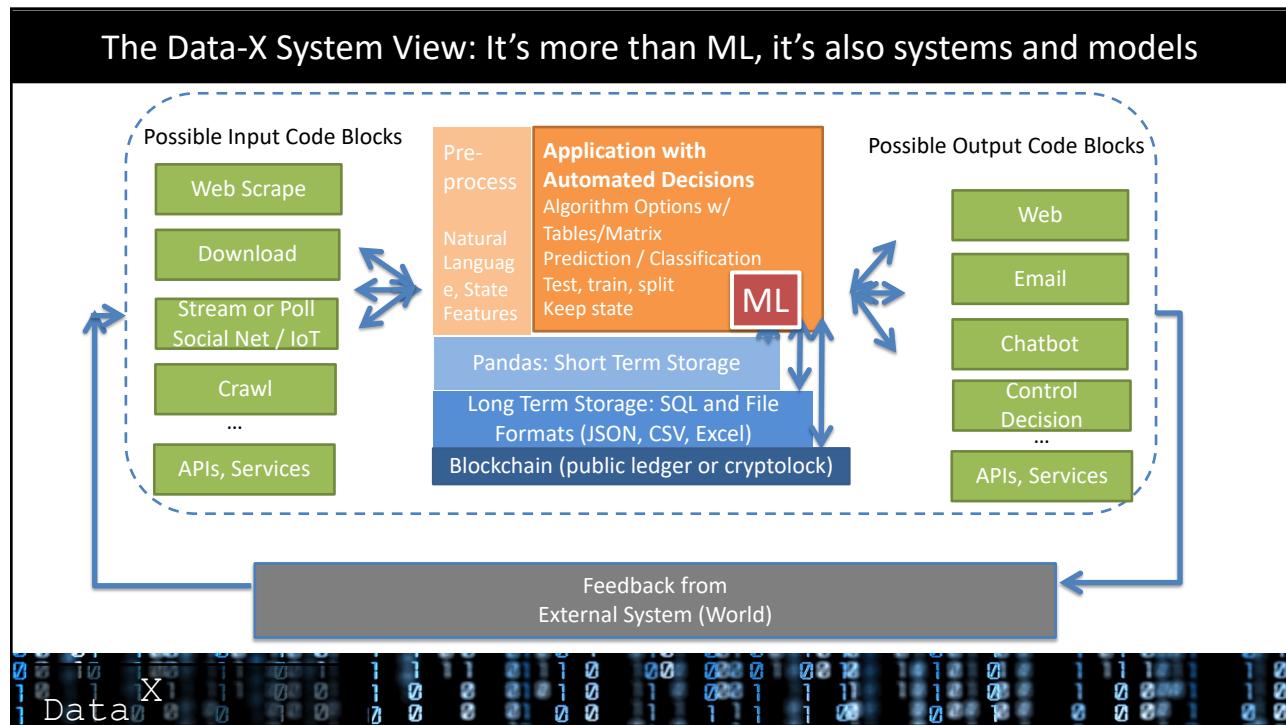


Illustration Source: <https://docs.microsoft.com/en-us/azure/machine-learning/machine-learning-algorithm-choice>



## Scikit-Learn Algorithm





End of Section