Machine Learning Summary with Illustrations
Part I – Setting UP for ML
Data X: A Course on Data, Signals, and Systems

Ikhlaq Sidhu
Chief Scientist & Founding Director,
Sutardja Center for Entrepreneurship & Technology
IEOR Emerging Area Professor Award, UC Berkeley
Open-ended, real-world project: Typically 5 students, with available advisor network
# 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

\[
\begin{align*}
X_{\text{train}} &= X[0:500] \\
Y_{\text{train}} &= Y[0:500] \\
X_{\text{test}} &= X[501:1000] \\
Y_{\text{test}} &= Y[501:1000]
\end{align*}
\]
Logistic regression

Data with a linear trend

Although it confusingly includes 'regression' in the name, logistic regression is actually a powerful tool for two-class and multiclass classification. It's fast and simple. The fact that it uses an 'S'-shaped curve instead of a straight line makes it a natural fit for dividing data into groups. Logistic regression gives linear class boundaries, so when you use it, make sure a linear approximation is something you can live with.

Illustration Source: https://docs.microsoft.com/en-us/azure/machine-learning/machine-learning-algorithm-choice
# 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:**
Do you have enough data to train and then test?

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

How to use the data efficiently?
• 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:

1. Shuffle the dataset randomly.
2. Split the dataset into k groups
3. For each unique group:
   1. Take the group as a hold out or test data set
   2. Take the remaining groups as a training data set
   3. Fit a model on the training set and evaluate it on the test set
4. Retain the evaluation score and discard the model

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
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
Titanic Notebook

Passenger List with ticket / cabin information → Data in Pandas Table Format → Clean Data Format for ML Models → Run Many ML Models to predict Survival

Passenger List

<table>
<thead>
<tr>
<th>PassengerId</th>
<th>Survived</th>
<th>Pclass</th>
<th>Sex</th>
<th>Age</th>
<th>SibSp</th>
<th>Parch</th>
<th>Ticket</th>
<th>Fare</th>
<th>Cabin</th>
<th>Embarked</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>3</td>
<td>male</td>
<td>22.0</td>
<td>1</td>
<td>0</td>
<td>A/5 21171</td>
<td>7.2500</td>
<td>NaN</td>
<td>S</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>female</td>
<td>36.0</td>
<td>1</td>
<td>0</td>
<td>PC 17599</td>
<td>71.2833</td>
<td>C85</td>
<td>C</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>3</td>
<td>female</td>
<td>26.0</td>
<td>0</td>
<td>0</td>
<td>STON/O2. 3101282</td>
<td>7.9250</td>
<td>NaN</td>
<td>S</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>1</td>
<td>female</td>
<td>35.0</td>
<td>1</td>
<td>0</td>
<td>113803</td>
<td>53.1900</td>
<td>C123</td>
<td>S</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>3</td>
<td>male</td>
<td>35.0</td>
<td>0</td>
<td>0</td>
<td>373450</td>
<td>8.0500</td>
<td>NaN</td>
<td>S</td>
</tr>
</tbody>
</table>

Cleaned and Formatted

<table>
<thead>
<tr>
<th>Survived</th>
<th>Pclass</th>
<th>Sex</th>
<th>Age</th>
<th>Fare</th>
<th>Embarked</th>
<th>Title</th>
<th>IsAlone</th>
<th>Age*Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>0</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Our experiment with the Titanic Data Set

<table>
<thead>
<tr>
<th>Model</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random Forest</td>
<td>86.76</td>
</tr>
<tr>
<td>Decision Tree</td>
<td>86.76</td>
</tr>
<tr>
<td>KNN</td>
<td>84.74</td>
</tr>
<tr>
<td>Support Vector Machines</td>
<td>83.84</td>
</tr>
<tr>
<td>Logistic Regression</td>
<td>80.36</td>
</tr>
<tr>
<td>Linear SVC</td>
<td>79.01</td>
</tr>
<tr>
<td>Perceptron</td>
<td>78.00</td>
</tr>
<tr>
<td>Naive Bayes</td>
<td>72.28</td>
</tr>
<tr>
<td>Stochastic Gradient Decent</td>
<td>72.28</td>
</tr>
</tbody>
</table>

More Accuracy
Generally more training time
More risk of overfitting

Less Accuracy
Generally less computation
Decision forests (regression, two-class, and multiclass), decision jungles (two-class and multiclass), and boosted decision trees (regression and two-class) are all based on decision trees, a foundational machine learning concept. There are many variants of decision trees, but they all do the same thing—subdivide the feature space into regions with mostly the same label. These can be regions of consistent category or of constant value, depending on whether you are doing classification or regression.

### Flavors of machine learning

**Supervised**

This cheat sheet has a very specific audience in mind: a beginning data scientist with undergraduate-level machine learning, trying to choose an algorithm to start with in Azure Machine Learning Studio. That means that it makes some generalizations and oversimplifications, but it will point you in a safe direction. It also means that there are lots of algorithms not listed here. As Azure Machine Learning grows to encompass a more complete set of available methods, we'll add them.

These recommendations are compiled feedback and tips from a lot of data scientists and machine learning experts. We didn't agree on everything, but I've tried to harmonize our opinions into a rough consensus. Most of the statements of disagreement begin with “It depends…”

Read the path and algorithm labels on the chart as “For <path label> use <algorithm>.” For example, “For speed use two-class logistic regression.” Sometimes more than one branch will apply. Sometimes none of them will be a perfect fit. They're intended to be rule-of-thumb recommendations, so don't worry about it being exact. Several data scientists I talked with said that the only sure way to find the very best algorithm is to try all of them.

Here's an example from the Cortana Intelligence Gallery of an experiment that tries several algorithms against the same data and compares the results:

**Compare Multi-class Classifiers:** Letter recognition.

To download and print a diagram that gives an overview of the capabilities of Machine Learning Studio, see Overview diagram of Azure Machine Learning Studio capabilities.

### Supervised learning algorithms

Supervised learning algorithms make predictions based on a set of examples. For instance, historical stock prices can be used to hazard guesses at future prices. Each example used for training is labeled with the value of interest—in this case the stock price. A supervised learning algorithm looks for patterns in those value labels. It can use any information that might be relevant—the day of the week, the season, the company's financial data, the type of industry, the presence of disruptive geopolitical events—and each algorithm looks for

Trees, forests, and jungles

A logistic regression to two-class data with just one feature
- the class boundary is the point at which the logistic curve is just as close to both classes

Decision forests (regression, two-class, and multiclass), decision jungles (two-class and multiclass), and boosted decision trees (regression and two-class) are all based on decision trees, a foundational machine learning concept. There are many variants of decision trees, but they all do the same thing—subdivide the feature space into regions with mostly the same label. These can be regions of consistent category or of constant value, depending on whether you are doing classification or regression.

```
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:
Support Vector Machine (SVM) Illustration

The two-class averaged perceptron is neural networks’ answer to skyrocketing training times. It uses a network structure that gives linear class boundaries. It is almost primitive by today’s standards, but it has a long history of working robustly and is small enough to learn quickly.

Support vector machines (SVMs) find the boundary that separates classes by as wide a margin as possible. When the two classes can’t be clearly separated, the algorithms find the best boundary they can. As written in Azure Machine Learning, the two-class SVM does this with a straight line only. (In SVM-speak, it uses a linear kernel.) Because it makes this linear approximation, it is able to run fairly quickly. Where it really shines is with feature-intense data, like text or genomic. In these cases SVMs are able to separate classes more quickly and with less overfitting than most other algorithms, in addition to requiring only a modest amount of memory.

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

\[
\mathbf{w} \cdot \mathbf{x} - b = 0.
\]

and

\[
\mathbf{w} \cdot \mathbf{x} - b = 1
\]

\[
\mathbf{w} \cdot \mathbf{x} - b = -1.
\]

The amount of separation is determined by \( \frac{b}{||w||} \) which determines the offset.

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

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)

# 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)

<table>
<thead>
<tr>
<th>Colour</th>
<th>Water</th>
<th>Rock</th>
</tr>
</thead>
<tbody>
<tr>
<td>Red</td>
<td>109</td>
<td>24</td>
</tr>
<tr>
<td>Green</td>
<td>112</td>
<td>14</td>
</tr>
<tr>
<td>Blue</td>
<td>105</td>
<td>13</td>
</tr>
<tr>
<td>Red</td>
<td>137</td>
<td>15</td>
</tr>
<tr>
<td>Green</td>
<td>164</td>
<td>11</td>
</tr>
<tr>
<td>Blue</td>
<td>125</td>
<td>1</td>
</tr>
<tr>
<td>Red</td>
<td>179</td>
<td>24</td>
</tr>
<tr>
<td></td>
<td>209</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>177</td>
<td>13</td>
</tr>
<tr>
<td></td>
<td>136</td>
<td>17</td>
</tr>
<tr>
<td></td>
<td>119</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>107</td>
<td>0</td>
</tr>
</tbody>
</table>
**KNN Method**: Find the $k$ nearest images and have them vote on the label (i.e., take the mode).

**Supervised learning algorithms** make predictions based on a set of examples. For instance, historical stock prices can be used to hazard guesses at future prices. Each example used for training is labeled with the value of interest—in this case the stock price. A supervised learning algorithm looks for patterns in those value labels. It can use any information that might be relevant—the day of the week, the season, the company's financial data, the type of industry, the presence of disruptive geopolitical events—and each algorithm looks for patterns.

**KNN / K Means Illustration**

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

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

**KNN / K Means Illustration**

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

K Means / KNN Illustration

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

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

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
Decision Tree Illustration

### Flavors of machine learning

**Supervised**

This cheat sheet has a very specific audience in mind: a beginning data scientist with undergraduate-level machine learning, trying to choose an algorithm to start with in Azure Machine Learning Studio. That means that it makes some generalizations and oversimplifications, but it will point you in a safe direction. It also means that there are lots of algorithms not listed here. As Azure Machine Learning grows to encompass a more complete set of available methods, we’ll add them.

These recommendations are compiled feedback and tips from a lot of data scientists and machine learning experts. We didn’t agree on everything, but I’ve tried to harmonize our opinions into a rough consensus. Most of the statements of disagreement begin with “It depends…”

Read the path and algorithm labels on the chart as “For <path label> use <algorithm>.” For example, “For speed use two class logistic regression.” Sometimes more than one branch will apply. Sometimes none of them will be a perfect fit. They’re intended to be rule-of-thumb recommendations, so don’t worry about it being exact. Several data scientists I talked with said that the only sure way to find the very best algorithm is to try all of them.

Here’s an example from the Cortana Intelligence Gallery of an experiment that tries several algorithms against the same data and compares the results:

**Compare Multi-class Classifiers:** Letter recognition.

To download and print a diagram that gives an overview of the capabilities of Machine Learning Studio, see [Overview diagram of Azure Machine Learning Studio capabilities](https://docs.microsoft.com/en-us/azure/machine-learning/machine-learning-algorithm-choice).

Supervised learning algorithms make predictions based on a set of examples. For instance, historical stock prices can be used to hazard guesses at future prices. Each example used for training is labeled with the value of interest—in this case the stock price. A supervised learning algorithm looks for patterns in those value labels. It can use any information that might be relevant—the day of the week, the season, the company’s financial data, the type of industry, the presence of disruptive geopolitical events—and each algorithm looks for

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

Decision Tree Illustration

```python
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: https://pypi.org/project/scikit-learn/
Our experiment with the Titanic Data Set

<table>
<thead>
<tr>
<th>Model</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random Forest</td>
<td>86.76</td>
</tr>
<tr>
<td>Decision Tree</td>
<td>86.76</td>
</tr>
<tr>
<td>KNN</td>
<td>84.74</td>
</tr>
<tr>
<td>Support Vector Machines</td>
<td>83.84</td>
</tr>
<tr>
<td>Logistic Regression</td>
<td>80.36</td>
</tr>
<tr>
<td>Linear SVC</td>
<td>79.01</td>
</tr>
<tr>
<td>Perceptron</td>
<td>78.00</td>
</tr>
<tr>
<td>Naive Bayes</td>
<td>72.28</td>
</tr>
<tr>
<td>Stochastic Gradient Decent</td>
<td>72.28</td>
</tr>
</tbody>
</table>

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

Less Accuracy
- Generally less computation
Random Forest – A type of bagging/ensemble approach

\[ S = \begin{cases} X_{1,1}, X_{1,2}, \ldots, X_{1,N}, X_{2,1}, X_{2,2}, \ldots, X_{2,N}, \ldots, X_{C,1}, X_{C,2}, \ldots, X_{C,N}, C_1, C_2, \ldots, C_C \end{cases} \]

- **Features**
- **Outcomes**

Make random subsets with replacement

\[ \text{During prediction, We use voting from each subtree} \]

**Advantages:**
- One of most accurate
- Efficient prediction over large data

**Disadvantages:**
- Overfit
- Training time
Trees Can be Extended with Bagging

```python
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

<table>
<thead>
<tr>
<th>Model</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random Forest</td>
<td>86.76</td>
</tr>
<tr>
<td>Decision Tree</td>
<td>86.76</td>
</tr>
<tr>
<td>KNN</td>
<td>84.74</td>
</tr>
<tr>
<td>Support Vector Machines</td>
<td>83.84</td>
</tr>
<tr>
<td>Logistic Regression</td>
<td>80.36</td>
</tr>
<tr>
<td>Linear SVC</td>
<td>79.01</td>
</tr>
<tr>
<td>Perceptron</td>
<td>78.00</td>
</tr>
<tr>
<td>Naive Bayes</td>
<td>72.28</td>
</tr>
<tr>
<td>Stochastic Gradient Decent</td>
<td>72.28</td>
</tr>
</tbody>
</table>

More Accuracy
Generally more training time
More risk of overfitting

Less Accuracy
Generally less computation
Neural Network Illustration

The boundaries learned by neural networks can be complex and irregular.

Illustration Source: https://docs.microsoft.com/en-us/azure/machine-learning/machine-learning-algorithm-choice
Scikit-Learn Algorithm
The Data-X System View: It’s more than ML, it’s also systems and models

Possible Input Code Blocks:
- Web Scrape
- Download
- Stream or Poll Social Net / IoT
- Crawl...
- APIs, Services

Application with Automated Decisions:
- Algorithm Options w/ Tables/Matrix
- Prediction / Classification
- Test, train, split
- Keep state

Possible Output Code Blocks:
- Web
- Email
- Chatbot
- Control Decision...
- APIs, Services

Pre-process
- Natural Language, State Features

Pandas: Short Term Storage

Long Term Storage: SQL and File Formats (JSON, CSV, Excel)

Blockchain (public ledger or cryptolock)

Feedback from External System (World)
End of Section