CSI5180. Machine Learning for Bioinformatics Applications

Fundamentals of Machine Learning — tasks and performance metrics

by

Marcel Turcotte

Version **November 6, 2019**

Preamble

Preamble

Fundamentals of Machine Learning — tasks and performance metrics

In this lecture, we introduce concepts that will be essential throughout the semester: the types of machine learning tasks, the representation of the data, and the performance metrics.

General objective :

- **Describe** the fundamental concepts of machine learning

Learning objectives

- **Discuss** the type of tasks in machine learning
- **Present** the data representation
- **Describe** the main metrics used in machine learning

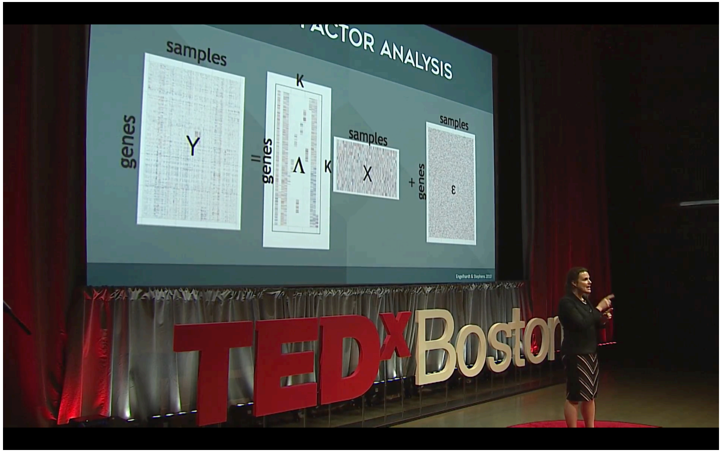
Reading:

- Larranaga, P. et al. Machine learning in bioinformatics. *Brief Bioinform* **7**:86112 (2006).
- Olson, R. S., Cava, W. L., Mustahsan, Z., Varik, A. & Moore, J. H. Data-driven advice for applying machine learning to bioinformatics problems. *Pac Symp Biocomput* **23**:192203 (2018).

Plan

Barbara Engelhardt, TEDx Boston 2017

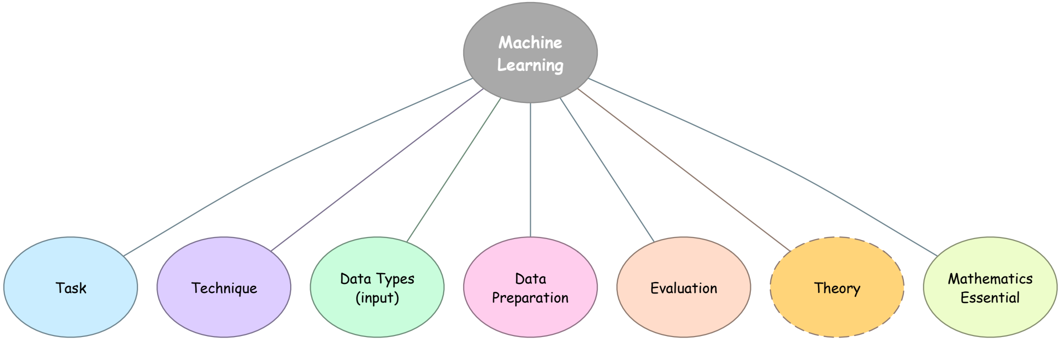
Not What but Why: Machine Learning for Understanding Genomics



https://youtu.be/uC3SfnbCXmw

Introduction

Concepts

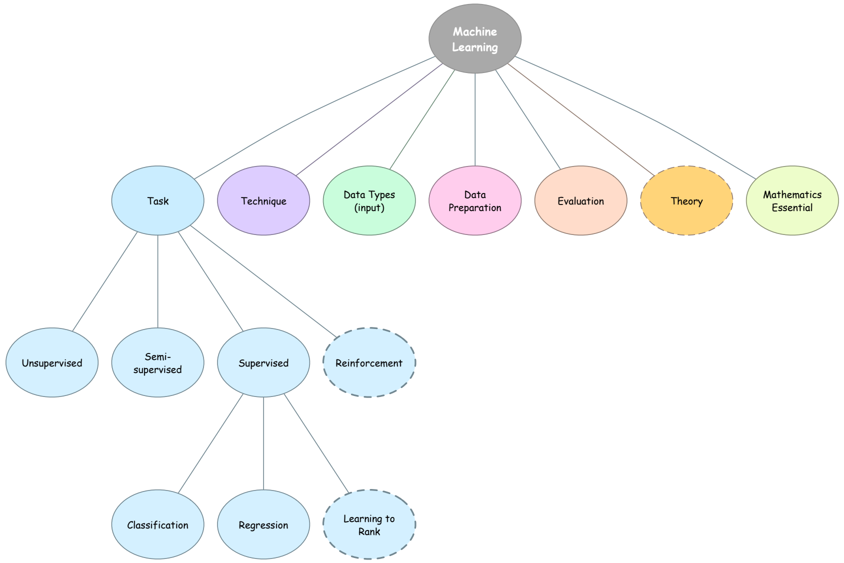


(http://www.site.uottawa.ca/~turcotte/teaching/csi-5180/lectures/04/01/ml_concepts.pdf)

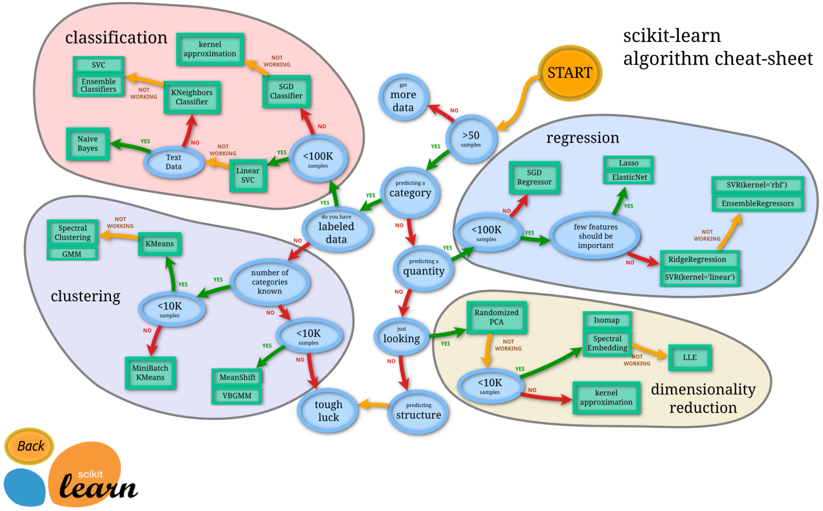
Definition

- Tom M Mitchell. *Machine Learning*. McGraw-Hill, New York, 1997.
  - "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 tasks in T, as measured by P, improves with experience E."

Tasks


Machine Learning
Task
Technique
Data Types (input)
Data Preparation
Evaluation
Theory
Mathematics Essential
Unsupervised
Semi-supervised
Supervised
Reinforcement
Classification
Regression
Learning to Rank

Scikit-Learn Cheat Sheet



https://scikit-learn.org/stable/tutorial/machine_learning_map/index.html

Supervised learning

- **Supervised learning** is the most common type of learning.
- The **data set** (“experience”) is a collection of **labelled** examples.
  - $\{(x_i, y_i)\}_{i=1}^N$
    - Each x_i is a **feature (attribute) vector** with D dimensions.
    - $x_k^{(j)}$ is the value of the **feature** j of the example k, for $j \in 1 \ldots D$ and $k \in 1 \ldots N$.
  - The **label** y_i is either a class, taken from a finite list of classes, $\{1, 2, \ldots, C\}$, or a **real number**, or a more complex object (vector, matrix, tree, graph, etc).
- **Problem**: given the data set as input, create a “**model**” that can be used to predict the value of y for an unseen x.

Supervised learning - an example

Prediction of Chemical Carcinogenicity in Human

- **Input** is a list of chemical compounds with information about their carcinogenicity.
  - Each compound is represented as a feature vector: electrogegativity, octanol-water partition, molecular weight, Pka, volume, dipole, etc.
- **Label**
  - **Classification**: $y_i \in \{\text{Carcinogenic}, \text{Not carcinogenic}\}$
  - **Regression**: y_i is a real number

See: `http://carcinogenome.org`

Unsupervised learning

- **Unsupervised learning** is often the first in a new machine learning project.
- The **data set** ("experience") is a collection of **unlabelled** examples.
  - $\{(x_i)\}_{i=1}^{N}$
    - Each x_i is a **feature (attribute) vector** with D dimensions.
    - $x_k^{(j)}$ is the value of the **feature** j of the eample k, for $j \in 1 \dots D$ and $k \in 1 \dots N$.
- **Problem**: given the data set as input, create a "**model**" that capture relationships in the data. In **clustering**, the task is to assign each example to a cluster. In **dimensionality reduction**, the task is to reduce the number of features in the input space.

Unsupervised learning - problems

- **Clustering**
  - K-Means, DBSCAN, hierarchical
- **Anomaly detection**
  - One-class SVM
- **Dimensionality reducation**
  - Principal Component Analysis (PCA), t-Distributed Stochastic Neighbor Embedding (t-SNE)

Supervised and unsupervised learning

Biomarker discovery - identifying breast cancer subtypes.

- **Input:** gene expression data for a large number of genes and a large number of patients. The data is labelled with information about the breast cancer subtype.
- It would be unpractical to devise a diagnostic test relying on a large number of genes (biomarkers).
- **Problem:** identify a subset of genes (features), such that the expression of those genes alone can be used to create a reliable classifier.
    - PAM50 is a group of 50 genes used for breast cancer subtype classification.

Semi-supervised learning

- The **data set** ("experience") is a collection of **labelled** and **unlabelled** examples.
  - Generally, the are **many more unlabelled examples** than labelled examples. Presumably, the cost of labelling examples is high.
- **Problem**: given the data set as input, create a "**model**" that can be used to predict the value of y for an unseen x. The goal is the same as for **supervised learning**. Having access to more examples is expected to help the algorithm.

Reinforcement learning

- In **reinforcement learning**, the agent “lives” in an **environment**.
- The **state** of the environment is represented as a feature vector.
- The agent is capable of **actions** that (possibly) change the state of the environment.
- Each action brings a **reward** (or punishment).
- **Problem:** learn a **policy** (a model) that takes as input a feature vector representing the environment and produce as output the **optimal** action - the action that maximizes the expected average reward.

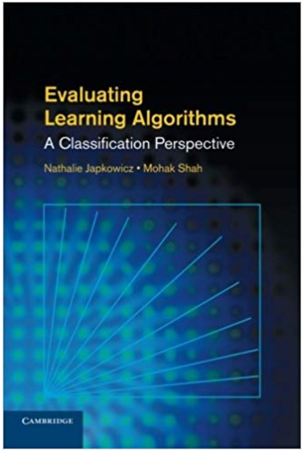
ML for bioinformatics

- In **industry**, **ML** is often used where **hand-coding** programs is complex/tedious.
  - Think about optical caracter recognition, image recognition, or driving an autonomous vehicle.
- In a related way, **ML** is advantageous for situations where the conditions/environment keeps **changing**.
  - Detecting/filtering spam/junk mail.
- In **bioinformatics**, the emphasis might be on the following:
  - Solving complex problems for which no satisfactory solution exists;
  - As part of the discovery process, extracting trends/patterns, leading to a better understanding of some problem.

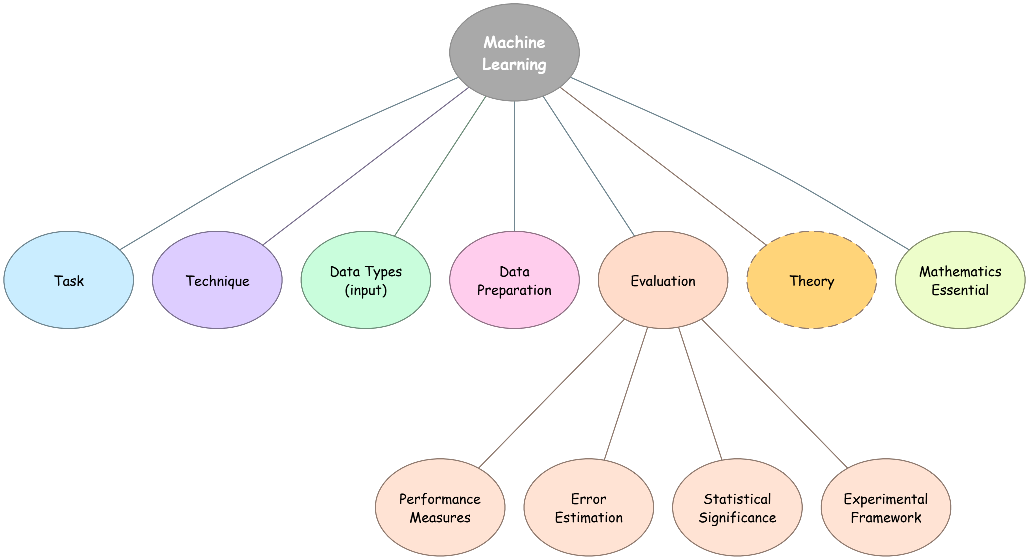
Here is **bibliography of machine learning for bioinformatics**, in **BibTeX** format as well as **PDF**.

Evaluation

Evaluating Learning Algorithms

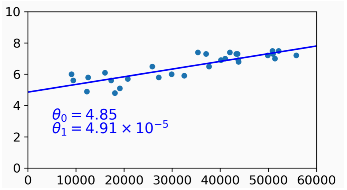


- Nathalie Japkowicz and Mohak Shah. *Evaluating Learning Algorithms: a classification perspective*. Cambridge University Press, Cambridge, 2011.


Machine Learning
Task
Technique
Data Types (input)
Data Preparation
Evaluation
Theory
Mathematics Essential
Performance Measures
Error Estimation
Statistical Significance
Experimental Framework

Words of caution

- **Sound** evaluation protocol
- The **right** performance measure
- We focus on **classification** problems since **regression** is often evaluated using simple measures, such as **root mean square deviation**



Source: Géron 2019, Figure 1.19

Confusion matrix – binary classification

		Predicted	
		Negative	**Positive**
Actual	**Negative**	True negative (TN)	False positive (FP)
	Positive	False negative (FN)	True positive (TP)

- In statistics, FP is often called **type I errors**, whereas FN is often called **type II errors**

Confusion matrix - binary classification

		Predicted	
		Negative	**Positive**
Actual	**Negative**	True negative (TN)	False positive (FP)
	Positive	False negative (FN)	True positive (TP)

- In statistics, FP is often called **type I errors**, whereas FN is often called **type II errors**
- The **confusion matrix** contains all the necessary information to evaluate our result.

Confusion matrix - binary classification

		Predicted	
		Negative	**Positive**
Actual	**Negative**	True negative (TN)	False positive (FP)
	Positive	False negative (FN)	True positive (TP)

- In statistics, FP is often called **type I errors**, whereas FN is often called **type II errors**
- The **confusion matrix** contains all the necessary information to evaluate our result.
- More **concise metrics**, such as **accuracy**, **precision**, **recall**, or **F**$_1$ score, are often more intutive to use.

sklearn.metrics.confusion_matrix

```python
from sklearn.metrics import confusion_matrix

y_actual = [0, 0, 0, 1, 1, 1, 1, 1, 1, 1]
y_pred   = [0, 1, 1, 0, 0, 0, 1, 1, 1, 1]

confusion_matrix(y_actual,y_pred)
```

```
array([[1, 2],
       [3, 4]])
```

```python
tn, fp, fn, tp = confusion_matrix(y_actual, y_pred).ravel()
(tn, fp, fn, tp)
```

```
(1, 2, 3, 4)
```

Perfect prediction

```
from sklearn.metrics import confusion_matrix

y_actual = [0, 1, 0, 0, 1, 1, 1, 0, 1, 1]
y_pred   = [0, 1, 0, 0, 1, 1, 1, 0, 1, 1]

confusion_matrix(y_actual,y_pred)
```

```
array([[4, 0],
       [0, 6]])
```

```
tn, fp, fn, tp = confusion_matrix(y_actual, y_pred).ravel()
(tn, fp, fn, tp)
```

```
(4, 0, 0, 6)
```

Accuracy

How **acccurate** is this result?

$$\text{accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

```
from sklearn.metrics import accuracy_score

y_actual = [0, 0, 0, 1, 1, 1, 1, 1, 1, 1]
y_pred   = [0, 1, 1, 0, 0, 0, 1, 1, 1, 1]

print(accuracy_score(y_actual,y_pred))
```

```
0.5
```

- **Accuracy** is the proportion of (all) your predictions that are correct

sklearn.metrics.accuracy_score

```
y_actual = [0, 1, 0, 0, 1, 1, 1, 0, 1, 1]
y_pred   = [1, 0, 1, 1, 0, 0, 0, 1, 0, 0]

print(accuracy_score(y_actual,y_pred))
```

```
0.0
```

```
y_actual = [0, 1, 0, 0, 1, 1, 1, 0, 1, 1]
y_pred   = [0, 1, 0, 0, 1, 1, 1, 0, 1, 1]

print(accuracy_score(y_actual,y_pred))
```

```
1.0
```

Accuracy can be misleading

```
y_actual = [0, 0, 0, 0, 1, 1, 0, 0, 0, 0]
y_pred   = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

print(accuracy_score(y_actual, y_pred))
```

- **What** is the acccury score?

Accuracy can be misleading

```
y_actual = [0, 0, 0, 0, 1, 1, 0, 0, 0, 0]
y_pred   = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

print(accuracy_score(y_actual,y_pred))
```

- **What** is the acccury score?
  - $(0+8)/10 = \mathbf{0.8}$

Accuracy can be misleading

```
y_actual = [0, 0, 0, 0, 1, 1, 0, 0, 0, 0]
y_pred   = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

print(accuracy_score(y_actual,y_pred))
```

- **What** is the acccury score?
  - $(0+8)/10 = \mathbf{0.8}$
- **Why** is it problematic?

Precision

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

```python
from sklearn.metrics import precision_score

y_actual = [0, 0, 0, 1, 1, 1, 1, 1, 1, 1]
y_pred   = [0, 1, 1, 0, 0, 0, 1, 1, 1, 1]

print(precision_score(y_actual,y_pred))
```

```
0.6666666666666666
```

- **Precision** is the proportion of your positive predictions that are correct

Precision alone is not enough

```
y_actual = [0, 0, 0, 1, 1, 1, 1, 1, 1, 1]
y_pred   = [0, 0, 0, 0, 0, 0, 1, 0, 0, 0]

print(precision_score(y_actual,y_pred))
```

- Given the above example, what is the **precision score**?

Precision alone is not enough

```
y_actual = [0, 0, 0, 1, 1, 1, 1, 1, 1, 1]
y_pred   = [0, 0, 0, 0, 0, 0, 1, 0, 0, 0]

print(precision_score(y_actual,y_pred))
```

- Given the above example, what is the **precision score**?
  - $1/(1+0) = 1.0$

Precision alone is not enough

```
y_actual = [0, 0, 0, 1, 1, 1, 1, 1, 1, 1]
y_pred   = [0, 0, 0, 0, 0, 0, 1, 0, 0, 0]

print(precision_score(y_actual,y_pred))
```

- Given the above example, what is the **precision score**?
  - $1/(1+0) = 1.0$
- **Why** is it problematic?

Precision alone is not enough

```
y_actual = [0, 0, 0, 1, 1, 1, 1, 1, 1, 1]
y_pred   = [0, 0, 0, 0, 0, 0, 1, 0, 0, 0]

print(precision_score(y_actual,y_pred))
```

- Given the above example, what is the **precision score**?
  - $1/(1+0) = 1.0$
- **Why** is it problematic?
  - One could select a small number of high confidence predictions and get a high precision score, but that might not be useful.

Recall (sensitivity or true positive rate (TPR))

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

```
from sklearn.metrics import recall_score

y_actual = [0, 0, 0, 1, 1, 1, 1, 1, 1, 1]
y_pred   = [0, 1, 1, 0, 0, 0, 1, 1, 1, 1]

print(recall_score(y_actual,y_pred))
```

```
0.5714285714285714
```

- **Recall** is the proportion of the true positive that are correctly predicted

F_1 score

$$F_1 \text{ score} = \frac{2}{\frac{1}{\text{precision}} + \frac{1}{\text{recall}}} = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}} = \frac{\text{TP}}{\text{FP} + \frac{\text{FN}+\text{FP}}{2}}$$

```
from sklearn.metrics import f1_score

y_actual = [0, 0, 0, 1, 1, 1, 1, 1, 1, 1]
y_pred   = [0, 1, 1, 0, 0, 0, 1, 1, 1, 1]

print(f1_score(y_actual,y_pred))
```

```
0.6153846153846153
```

- F_1 is the **harmonic mean** of precision and recall

Remarks

- The harmonic mean gives more weight to low values, whereas the arithmetic mean treats all the values equally.
- F_1 score favours classifiers having similar precision and recall.
- Depending on the specific problem, one might want to put more weight on one metric or the other.
    - Imagine classifier producing a list of candidates to be validated experimentally, say a list of **RNA** molecules having a specific motif will be packaged in **exosomes**.
    - A classifier having a high recall might produce a long list of motifs. However, creating a large collection of **knockout** molecules might be expensive.
- Increasing **recall** often occurs at the expense of lowering **precision**, and vice-versa. This called the **precision/recall trade-off**.

Precision/recall trade-off

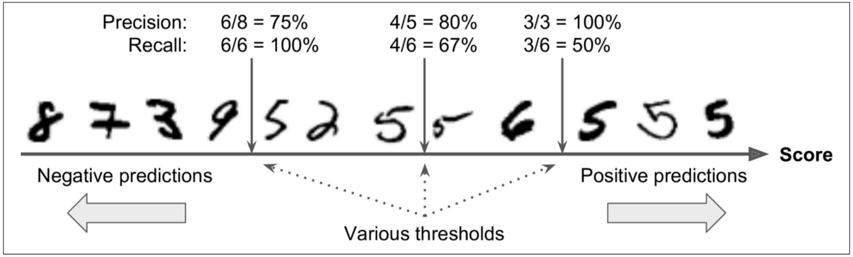
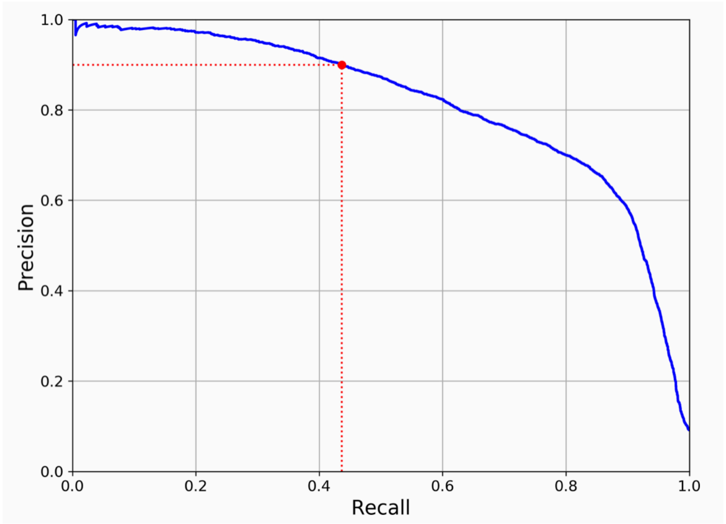


Figure 3-3. In this precision/recall trade-off, images are ranked by their classifier score, and those above the chosen decision threshold are considered positive; the higher the threshold, the lower the recall, but (in general) the higher the precision

Source: Géron 2019, Figure 3.3

Precision/recall trade-off



Source: Géron 2019, Figure 3.5

ROC curve

Receiver Operating Characteristics (ROC) curve

- **True positive rate** (TPR) against **false positive rate** (FPR)

		Predicted	
		Negative	**Positive**
Actual	**Negative**	True negative (TN)	False positive (FP)
	Positive	False negative (FN)	True positive (TP)

ROC curve

Receiver Operating Characteristics (ROC) curve

- **True positive rate** (TPR) against **false positive rate** (FPR)
- An ideal classifier has **TPR** close to **1.0** and **FPR** close to **0.0**

		Predicted	
		Negative	**Positive**
Actual	**Negative**	True negative (TN)	False positive (FP)
	Positive	False negative (FN)	True positive (TP)

ROC curve

Receiver Operating Characteristics (ROC) curve

- **True positive rate** (TPR) against **false positive rate** (FPR)
- An ideal classifier has **TPR** close to **1.0** and **FPR** close to **0.0**
- $\mathrm{TPR} = \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}}$ (recall)

		Predicted	
		Negative	**Positive**
Actual	**Negative**	True negative (TN)	False positive (FP)
	Positive	False negative (FN)	True positive (TP)

ROC curve

Receiver Operating Characteristics (ROC) curve

- **True positive rate** (TPR) against **false positive rate** (FPR)
- An ideal classifier has **TPR** close to **1.0** and **FPR** close to **0.0**
- $\mathrm{TPR} = \frac{\mathrm{TP}}{\mathrm{TP+FN}}$ (recall)
- **TPR** approaches **one** when the number of **false negative** predictions is low

		Predicted	
		Negative	**Positive**
Actual	**Negative**	True negative (TN)	False positive (FP)
	Positive	False negative (FN)	True positive (TP)

ROC curve

Receiver Operating Characteristics (ROC) curve

- **True positive rate** (TPR) against **false positive rate** (FPR)
- An ideal classifier has **TPR** close to **1.0** and **FPR** close to **0.0**
- $\text{TPR} = \frac{\text{TP}}{\text{TP}+\text{FN}}$ (recall)
- **TPR** approaches **one** when the number of **false negative** predictions is low
- $\text{FPR} = \frac{\text{FP}}{\text{FP}+\text{TN}}$ (a.k.a. [1-specificity])

		Predicted	
		Negative	**Positive**
Actual	**Negative**	True negative (TN)	False positive (FP)
	Positive	False negative (FN)	True positive (TP)

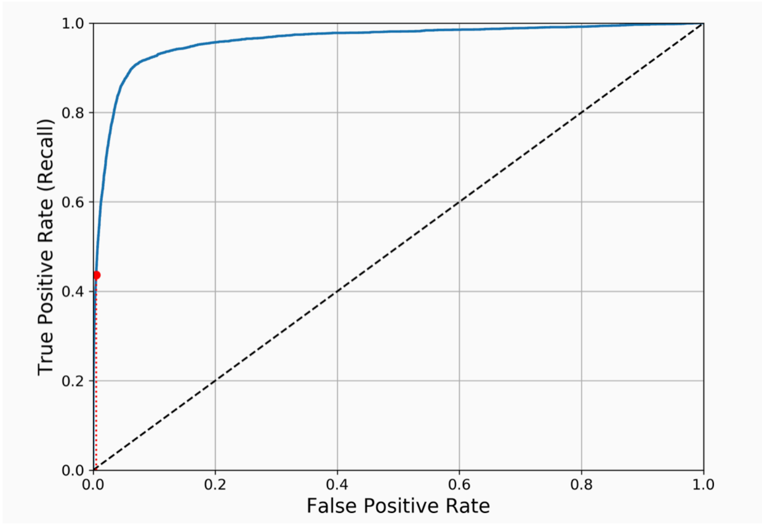
ROC curve

Receiver Operating Characteristics (ROC) curve

- **True positive rate** (TPR) against **false positive rate** (FPR)
- An ideal classifier has **TPR** close to **1.0** and **FPR** close to **0.0**
- $\mathrm{TPR} = \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}}$ (recall)
- **TPR** approaches **one** when the number of **false negative** predictions is low
- $\mathrm{FPR} = \frac{\mathrm{FP}}{\mathrm{FP}+\mathrm{TN}}$ (a.k.a. [1-specificity])
- **FPR** approaches **zero** when the number of **false positive** is low

		Predicted	
		Negative	**Positive**
Actual	**Negative**	True negative (TN)	False positive (FP)
	Positive	False negative (FN)	True positive (TP)

ROC curve



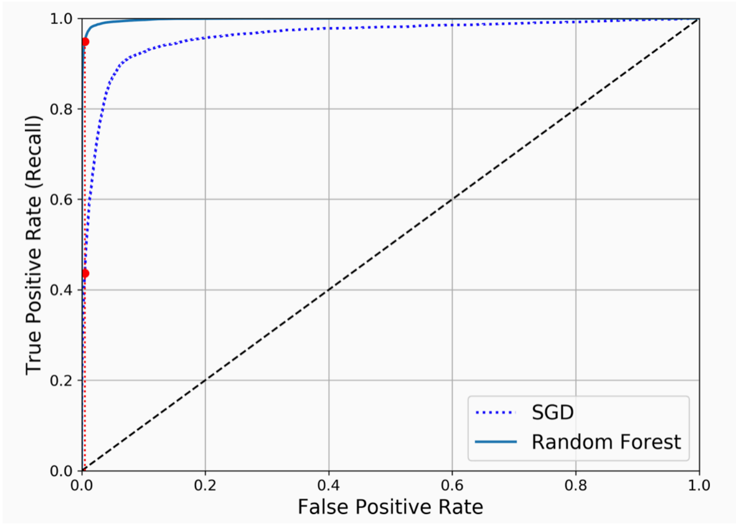
Source: Géron 2019, Figure 3.6

sklearn.metrics.roc_curve

```
from sklearn.metrics import roc_curve

fpr, tpr, thresholds = roc_curve(y_actual, y_pred_scores)
```

Area Under the Curve (AUC)



Source: Géron 2019, Figure 3.7

sklearn.metrics.roc_auc_score

```
from sklearn.metrics import roc_auc_score

roc_auc_score(y_actual, y_pred_scores)
```

- **SGD** has an **AUC** of **0.9611778893101814**
- **Random Forest** has an **AUC** of **0.9983436731328145**

AUC/Bioinformatics

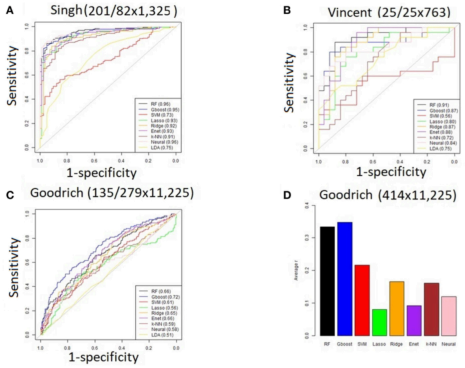


FIGURE 2 | (A–C) ROC curves for each machine learning method using all OTUs. The AUC values are shown in the legend. The size of each dataset (# cases/controls × # OTUs) is shown in the title. **(D)** Bar graph showing the average Pearson correlation (R) between predicted and actual BMI in the Goodrich dataset, using BMI as a continuous trait.

- Zhou, Y.-H. & Gallins, P. A Review and Tutorial of Machine Learning Methods for Microbiome Host Trait Prediction. *Front Genet* **10**, 579 (2019).

Prologue

Summary

- **Unsupervised** and **supervised** learning are the two main types of tasks in machine learning. Other types include **semi-supervised** learning and **reinforcement** learning.

Summary

- **Unsupervised** and **supervised** learning are the two main types of tasks in machine learning. Other types include **semi-supervised** learning and **reinforcement** learning.
- Supervised learning uses **labelled** examples.

Summary

- **Unsupervised** and **supervised** learning are the two main types of tasks in machine learning. Other types include **semi-supervised** learning and **reinforcement** learning.
- Supervised learning uses **labelled** examples.
    - When the label is a class (or a complex object, such as a matrix or a graph), the learning task is called **classification**. Given some unseen example x predict its lable y.

Summary

- **Unsupervised** and **supervised** learning are the two main types of tasks in machine learning. Other types include **semi-supervised** learning and **reinforcement** learning.
- Supervised learning uses **labelled** examples.
  - When the label is a class (or a complex object, such as a matrix or a graph), the learning task is called **classification**. Given some unseen example x predict its lable y.
  - When the label is a real number, the task is called **regression**.

Summary

- **Unsupervised** and **supervised** learning are the two main types of tasks in machine learning. Other types include **semi-supervised** learning and **reinforcement** learning.
- Supervised learning uses **labelled** examples.
  - When the label is a class (or a complex object, such as a matrix or a graph), the learning task is called **classification**. Given some unseen example x predict its lable y.
  - When the label is a real number, the task is called **regression**.
- A **confusion matrix** describes the performance of (classification) learning algorithm.

Summary

- **Unsupervised** and **supervised** learning are the two main types of tasks in machine learning. Other types include **semi-supervised** learning and **reinforcement** learning.
- Supervised learning uses **labelled** examples.
  - When the label is a class (or a complex object, such as a matrix or a graph), the learning task is called **classification**. Given some unseen example x predict its lable y.
  - When the label is a real number, the task is called **regression**.
- A **confusion matrix** describes the performance of (classification) learning algorithm.
  - Performance measure such as **accuracy**, **precision**, **recall**, and $\mathbf{F}_a$ summarize different aspects of the confusion matrix.

Summary

- **Unsupervised** and **supervised** learning are the two main types of tasks in machine learning. Other types include **semi-supervised** learning and **reinforcement** learning.
- Supervised learning uses **labelled** examples.
  - When the label is a class (or a complex object, such as a matrix or a graph), the learning task is called **classification**. Given some unseen example x predict its lable y.
  - When the label is a real number, the task is called **regression**.
- A **confusion matrix** describes the performance of (classification) learning algorithm.
  - Performance measure such as **accuracy**, **precision**, **recall**, and $\mathbf{F}_a$ summarize different aspects of the confusion matrix.
- **ROC** curves allow to visualize the TPR vs FPR tradeoff, whereas **AUC** is useful to compare multiple algorithms or hyperparameters combinations.

Next module

- **Training** learning algorithms

References

Nathalie Japkowicz and Mohak Shah.
Evaluating Learning Algorithms: a classification perspective.
Cambridge University Press, Cambridge, 2011.

Aurélien Géron.
Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow.
O'Reilly Media, 2nd edition, 2019.

Andriy Burkov.
The Hundred-Page Machine Learning Book.
Andriy Burkov, 2019.

Tom M Mitchell.
Machine Learning.
McGraw-Hill, New York, 1997.

Marcel **Turcotte**
Marcel.Turcotte@uOttawa.ca

School of Electrical Engineering and **Computer Science** (EECS)
University of Ottawa