Python_N6_20161105_AaronYu

November 5, 2016

0.1 Summary

Learn Scikit-learn and Machine Learning

0.1.1 Intro to Machine Learning

- 1.Types of Machine Learning
  - Supervised vs Unsupervisied Learning
  - Regression vs Classification
- 2.Basic terminology
- 3.A roadmap for building machine learning systems
- 4.Recap

0.1.2 Intro to Scikit-learn a.k.a. sklearn

- 1.Official Documents
- 2.Example_1 - Linear Regression
- 3.Example_2 - KNN

```
In [1]: import numpy as np
        import pandas as pd
        import matplotlib.pyplot as plt
        import sklearn
        import warnings
        warnings.filterwarnings('ignore')
        %matplotlib inline
```

```
In [2]: from IPython.display import Image
        from IPython.core.interactiveshell import InteractiveShell
        InteractiveShell.ast_node_interactivity = "all"
```

0.2 Intro to Machine Learning

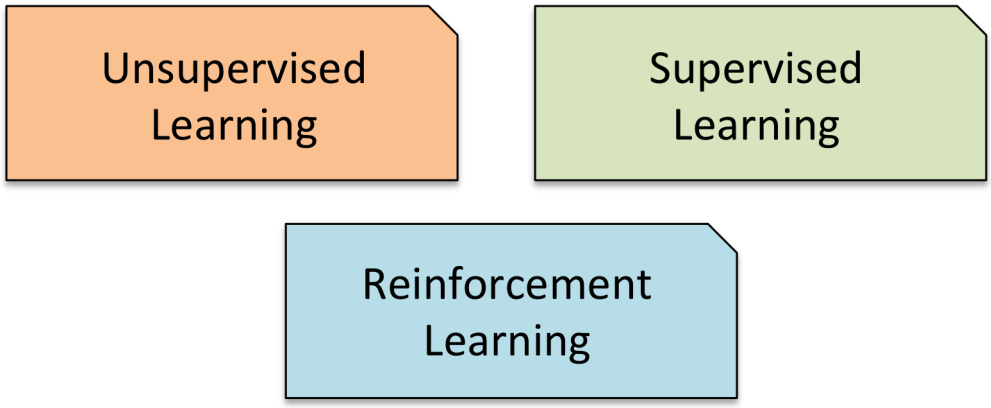
0.2.1 Three different types of machine learning

- Supervised learning - Making predictions about the future
- Unsupervised learning - Discovering hidden structures

- Reinforcement learning - Solving interactive problems

```
In [3]: Image(filename='./Image/01_01.png', width=500)
```

```
Out[3]:
```

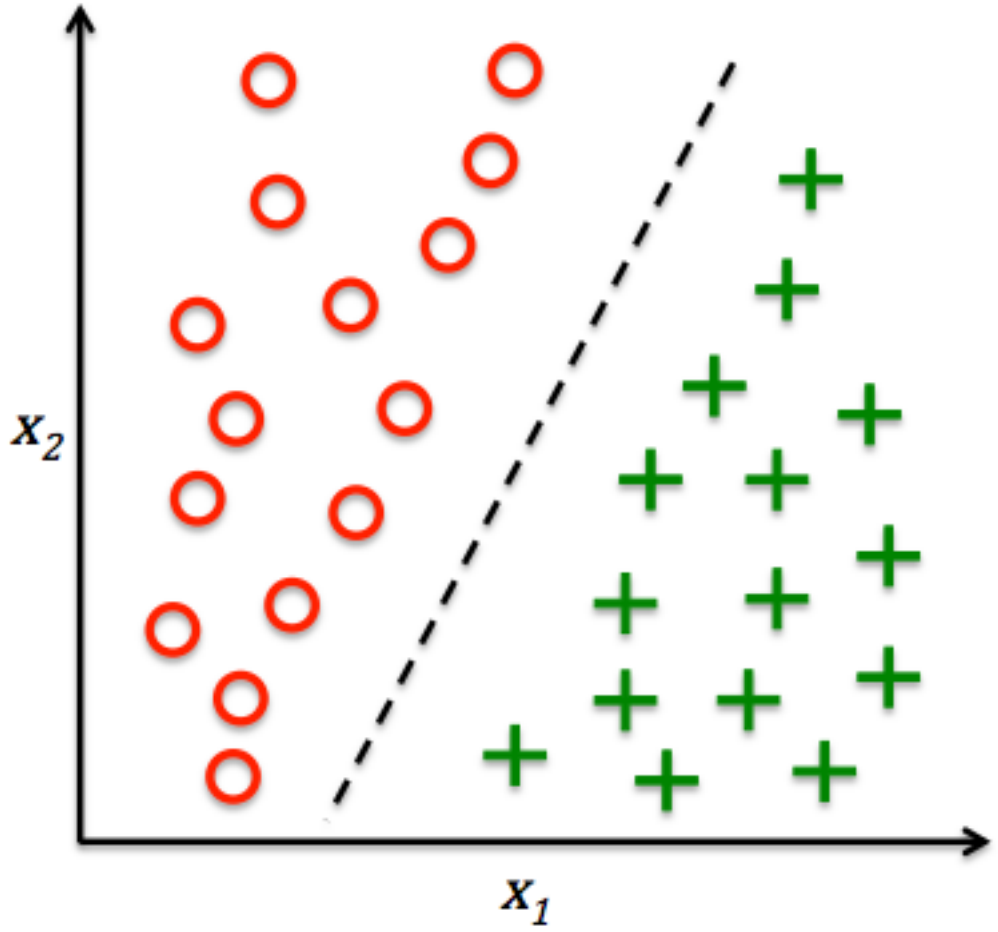


0.2.2 Making predictions about the future with supervised learning

- Classification for predicting class labels

```
In [4]: Image(filename='./Image/01_03.png', width=400, height=50)
```

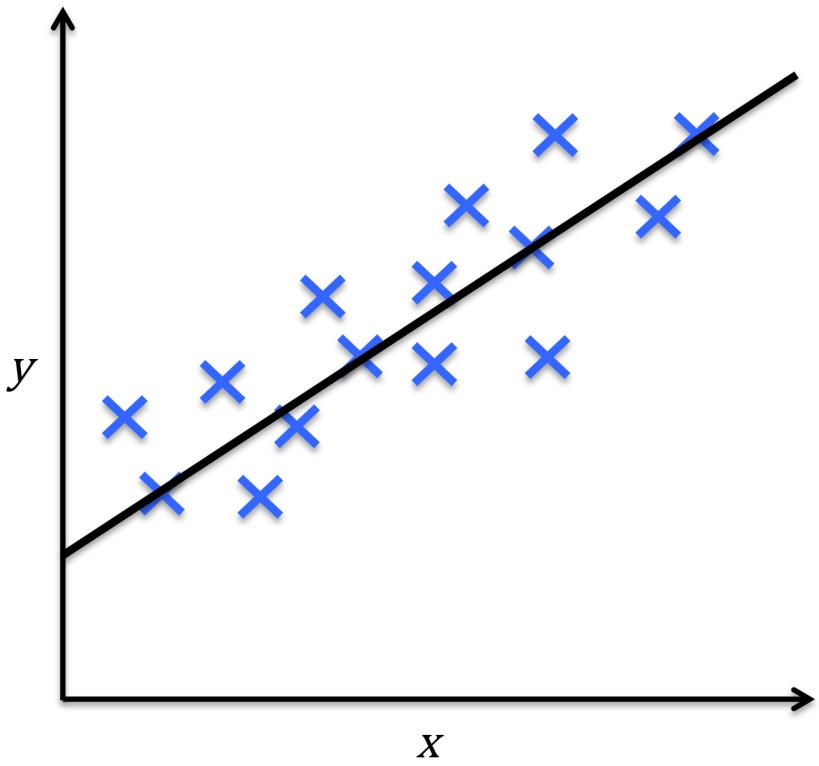
```
Out[4]:
```



- Regression for predicting continuous outcomes

```
In [5]: Image(filename='./Image/01_04.png', width=400, height=50)
```

```
Out[5]:
```

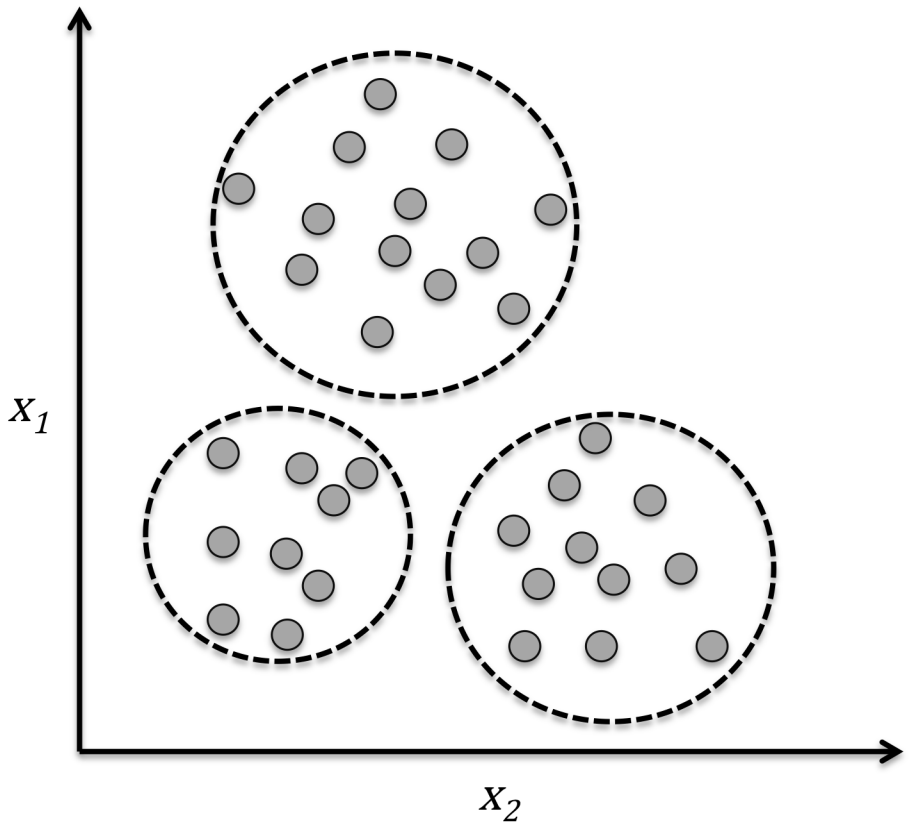


0.2.3 Discovering hidden structures with unsupervised learning

- Finding subgroups with clustering
- Dimensionality reduction

```
In [6]: Image(filename='./Image/01_06.png', width=400, height=50)
```

```
Out[6]:
```



0.2.4 An introduction to the basic terminology

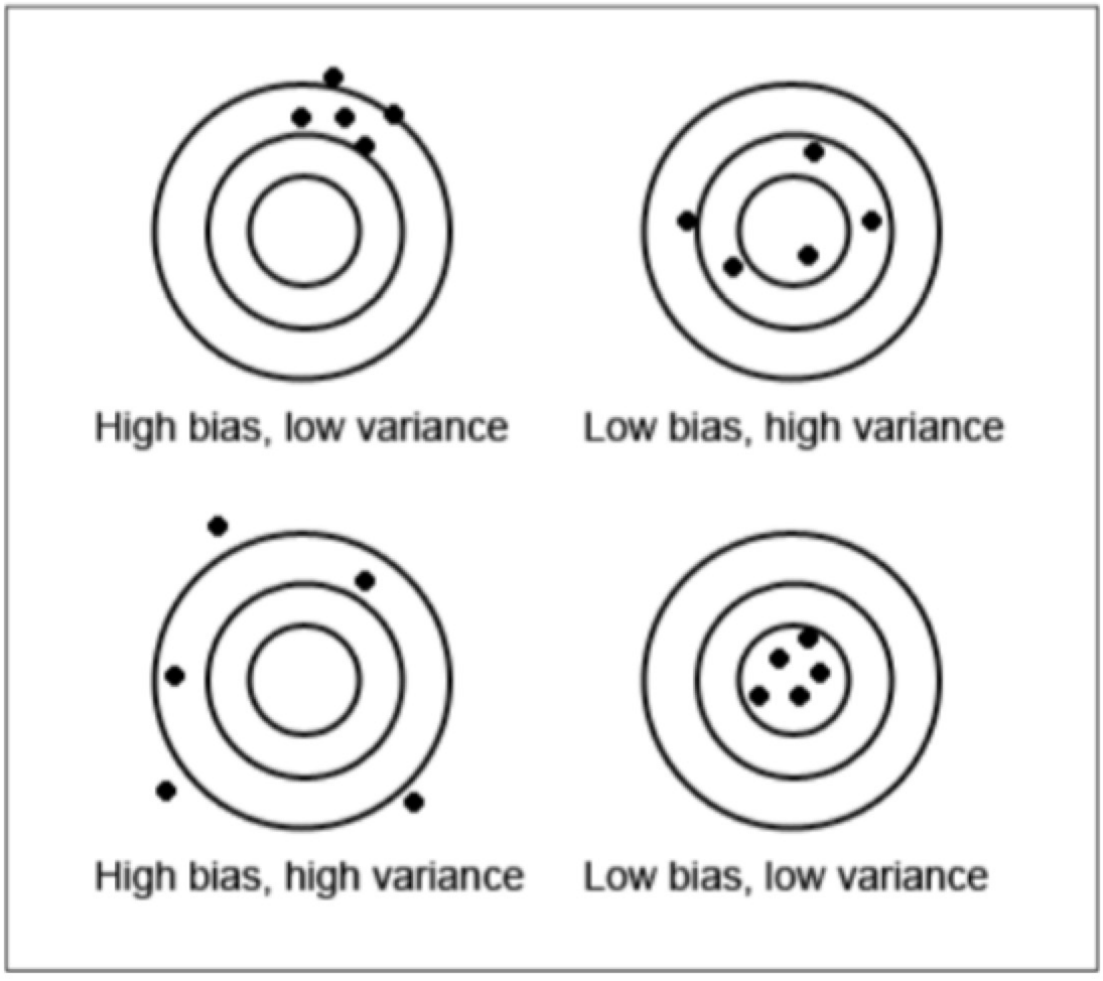
- Input & Output Variables
  - Output variables: response variable, or dependent variables, labels
  - Input variables: explanatory variables, or predictors
  - Response variables and explanatory variables may take real or discrete values.
- Training, Testing and Validating Dataset
  - The collection of examples that comprise supervised experience is called a training set.
  - The collection of examples that is used to assess the performance of a program is called a test set.
  - In addition to the training and test data, a third set of observations, called a validation or hold-out set, is sometimes used to tune variables called hyperparameters, which control how the model is learned.
- Bias & Variance (or Underfitting vs Overfitting)
  - Bias: A model with a high bias will produce similar errors for an input regardless of the training set it was trained with; the model biases its own assumptions about the real relationship over the relationship demonstrated in the training data.

  - Variance: A model with high variance, conversely, will produce different errors for an input depending on the training set that it was trained with.
  - A model with high bias is inflexible, but a model with high variance may be so flexible that it models the noise in the training set. That is, a model with high variance over-fits the training data, while a model with high bias under-fits the training data.

- Performance Measures
  - Regression: MSE, MAE
  - Classification: Accuracy Score, Precision, Recall, F1, ROC, AUC

```
In [7]: Image(filename='./Image/bias-variance.png', width=400, height=50)
```

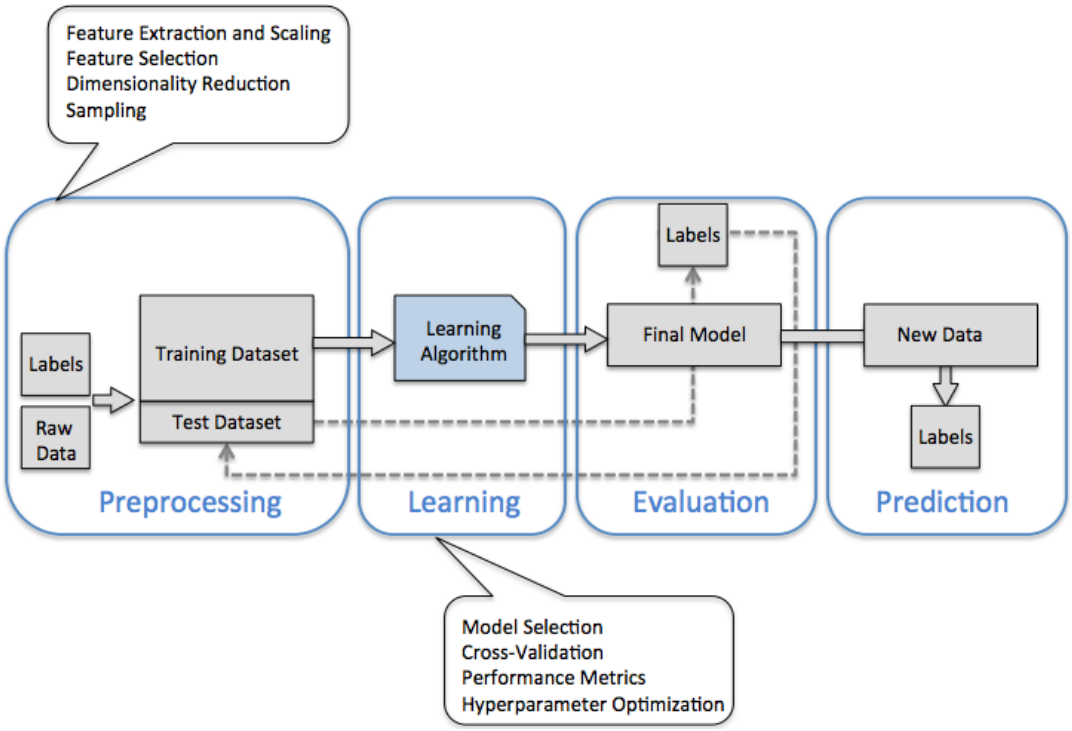
```
Out[7]:
```



0.2.5 A roadmap for building machine learning systems

```
In [8]: Image(filename='./Image/01_09.png', width=600, height=50)
```

Out[8]:



0.2.6 Recap

- Linear Regression
- Logistic Regression
- K Nearest Neighbours
- Decision Tree
- K-Means

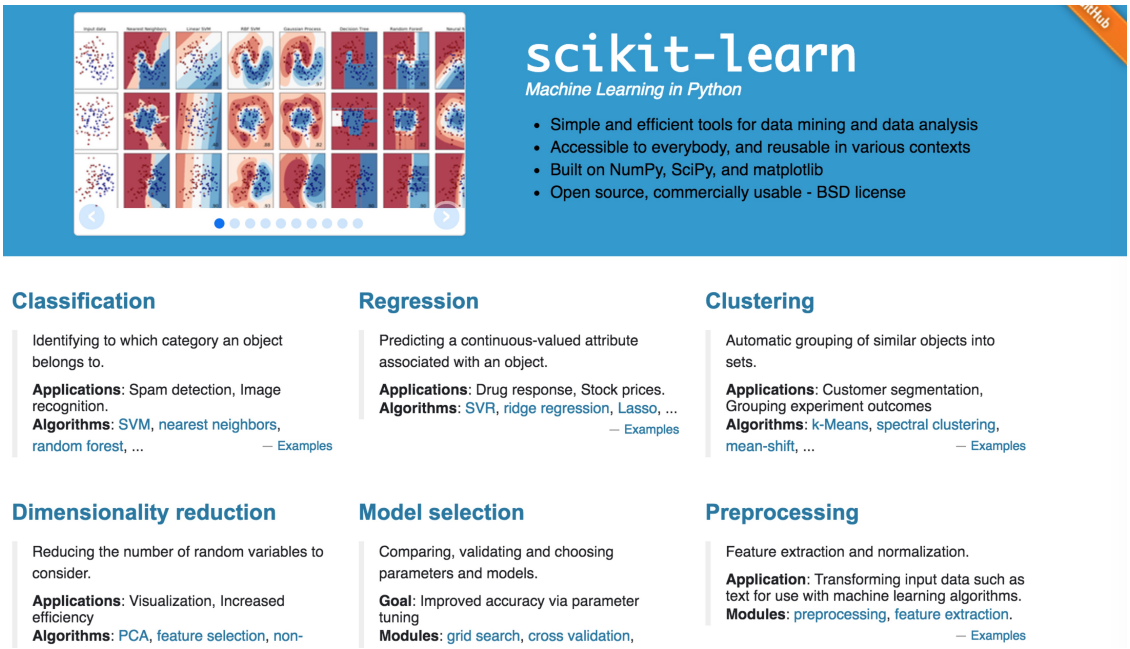
0.3 Intro to Scikit-learn a.k.a. sklearn

0.3.1 Official Documents

http://scikit-learn.org/stable/index.html

```
In [9]: Image(filename='./Image/sklearn.png', width=800, height=50)
```

Out[9]:



0.3.2 Example_1 - Linear Regression

```
In [10]: from sklearn import datasets, linear_model
         from sklearn import model_selection, metrics

In [11]: diabetes = datasets.load_diabetes()
         y = diabetes.target
         x = diabetes.data[:,0]

In [12]: x_train, x_test, y_train, y_test = model_selection.train_test_split(x, y,

In [13]: lr = linear_model.LinearRegression()

In [14]: lr.fit(x_train.reshape(-1,1), y_train)

Out[14]: LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1, normalize=Fals

In [15]: metrics.mean_squared_error(y_test, lr.predict(x_test.reshape(-1,1)))

Out[15]: 5332.6970984690006
```

0.3.3 Example_2 - KNN

```
In [16]: from sklearn import datasets, neighbors

In [17]: digits = datasets.load_digits()
         X_digits = digits.data
         y_digits = digits.target
```

```
n_samples = len(X_digits)

X_train = X_digits[:.9 * n_samples]
y_train = y_digits[:.9 * n_samples]
X_test = X_digits[.9 * n_samples:]
y_test = y_digits[.9 * n_samples:]
```

```
In [18]: knn = neighbors.KNeighborsClassifier()
```

```
In [19]: knn
```

```
Out[19]: KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                              metric_params=None, n_jobs=1, n_neighbors=5, p=2,
                              weights='uniform')
```

```
In [20]: knn.fit(X_train, y_train).score(X_test, y_test)
```

```
Out[20]: 0.96111111111111114
```

```
In [21]: neighbours = range(1,6)
         test_score = []
         for n in neighbours:
             knn = neighbors.KNeighborsClassifier(n_neighbors=n)
             score = knn.fit(X_train, y_train).score(X_test, y_test)
             test_score.append(score)
```

```
In [22]: plt.plot(neighbours, test_score)
```

```
Out[22]: [<matplotlib.lines.Line2D at 0x11ba27908>]
```