# Abstract

Decision tree and Perceptron learning algorithms have been successfully used in expert systems in capturing knowledge. The main task performed in these systems is using inductive methods to the given values of attributes of an unknown object to determine appropriate classification according to decision tree rules or artificial neural network. We examine the decision tree learning algorithm D3 and Perceptron artificial neural network and implemented this algorithms using Python with famous iris dataset.

**Methodology**

The aim of the project is to compare two machine learning algorithms. ID3 decision tree and Artificial neural network Perceptron algorithms using famous Iris Dataset.

For this project I have used 70 percent of date to train algorithms and other 30 percent to test

**The Iris Dataset**

The iris dataset has been used for classification in many research publications. It consists of 50 samples from each of three classes of iris flowers [Frank and Asuncion, 2010]. One class is linearly separable from the other two, while the latter are not linearly separable from each other. There are five attributes in the dataset:

sepal length in cm,  
sepal width in cm,  
petal length in cm,  
petal width in cm, and  
class: Iris Setosa, Iris Versicolour, and Iris Virginica.

**Decision Tree**

What is decision tree: A decision tree is a tree in which each branch node represents a choice between a number of alternatives, and each leaf node represents a decision.

Decision tree are commonly used for gaining information for the purpose of decision -making. Decision tree starts with a root node on which it is for users to take actions. From this node, users split each node recursively according to decision tree learning algorithm. The final result is a decision tree in which each branch represents a possible scenario of decision and its outcome.

What is decision tree learning algorithm?

'Decision tree learning is a method for approximating discrete-valued target functions, in which the learned function is represented by a decision tree. Decision tree learning is one of the most widely used and practical methods for inductive inference.

Decision tree learning algorithm has been successfully used in expert systems in capturing knowledge. The main task performed in these systems is using inductive methods to the given values of attributes of an unknown object to determine appropriate classification according to decision tree rules.

Decision trees classify instances by traverse from root node to leaf node. We start from root node of decision tree, testing the attribute specified by this node, then moving down the tree branch according to the attribute value in the given set. This process is the repeated at the sub-tree level.

What is decision tree learning algorithm suited for:

1. Instance is represented as attribute-value pairs.

2.The target function has discrete output values. It can easily deal with instance which is assigned to a boolean decision, such as 'true' and 'false', 'p(positive)' and 'n(negative)'. Although it is possible to extend target to real- valued outputs.

3.The training data may contain errors. This can be dealt with pruning techniques that we will not cover here.

The 3 widely used decision tree learning algorithms are: ID3, ASSISTANT and C4.5.

Why is Decision Tree Learning an attractive Inductive Learning method

'Purely inductive learning methods formulate general hypotheses by finding empirical regularities over the training examples.

For inductive learning, decision tree learning is attractive for 3 reasons:

Decision tree is a good generalization for unobserved instance, only if the instances are described in terms of features that are correlated with the target concept.

The methods are efficient in computation that is proportional to the number of observed training instances.

The resulting decision tree provides are presentation of the concept that appeals to human because it renders the classification process self-evident

Decision Tree Learning Algorithm — ID3 Basic

ID3 Basic

ID3 is a simple decision tree learning algorithm developed by Ross Quinlan (1983). The basic idea of ID3 algorithm is to construct the decision tree by employing a top-down, greedy search through the given sets to test each attribute at every tree node. In order to select the attribute that is most useful for classifying a given sets, we introduce a metric---information gain.

Entropy --- measuring homogeneity of a learning set

Entropy(S)= - P(positive)log2P(positive) - P(negative)log2P(negative)

P(positive): proportion of positive examples in S

P(negative): proportion of negative examples in S

Information Gain --- measuring the expected reduction in Entropy

The information gain, Gain(S,A) of an attribute A,

Gain(S,A)= Entropy(S) -Sum for v from 1 to n of (|Sv|/|S|) \* Entropy(Sv)

**Perceptron Learning Algorithm**

The perceptron learning rule was originally developed by Frank Rosenblatt in the late 1950s.  Training patterns are presented to the network's inputs; the output is computed.  Then the connection weights **wj** are modified by an amount that is proportional to the product of

the difference between the actual output, **y,**  and the desired output, **d,** and the input pattern, **x**.

The algorithm is as follows:

1. Initialize the weights and threshold to small random numbers.
2. Present a vector **x** to the neuron inputs and calculate the output.
3. Update the weights according to:



1. where

**d** is the desired output,

**t** is the iteration number, and

**eta** is the gain or step size, where 0.0 < n < 1.0

1. Repeat steps 2 and 3 until:

the iteration error is less than a user-specified error threshold or

a predetermined number of iterations have been completed.

Notice that learning only occurs when an error is made, otherwise the weights are left unchanged.

This rule is thus a modified form of Hebb learning.

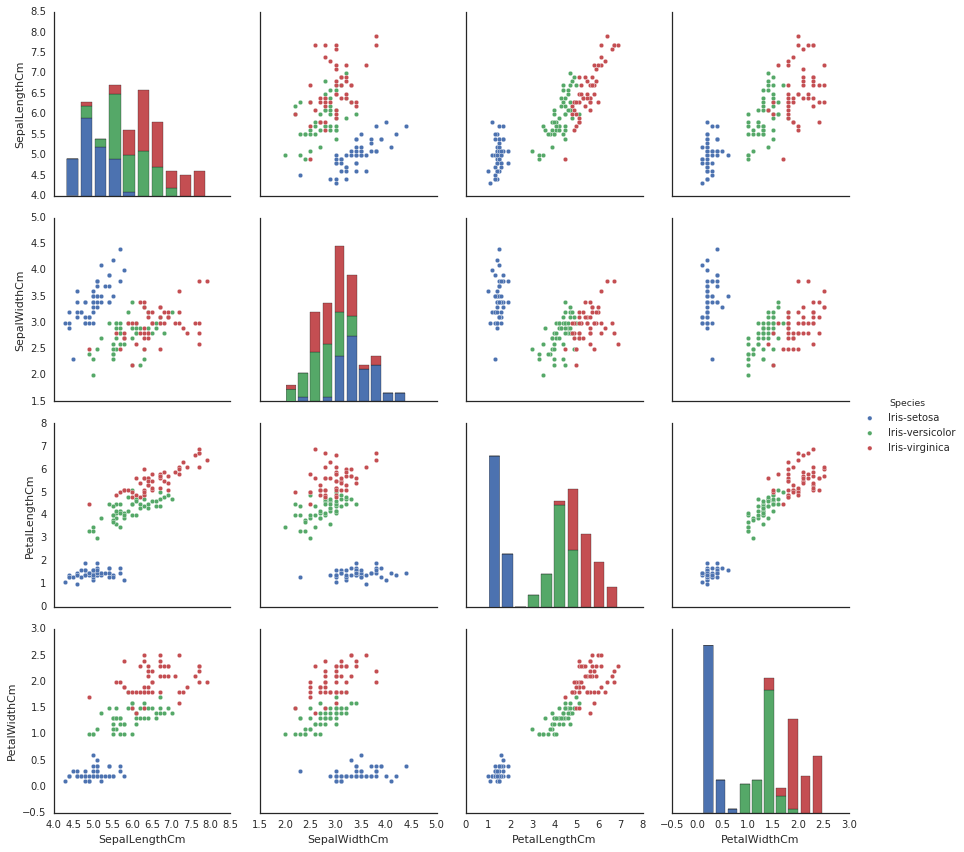
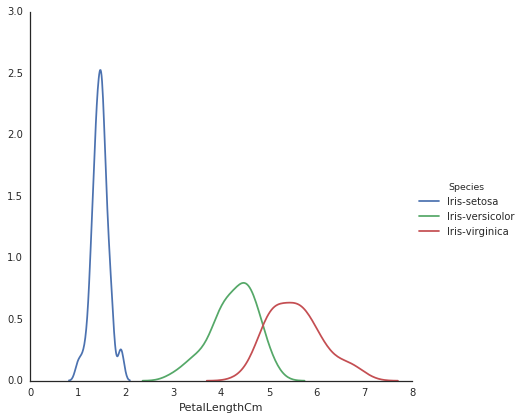
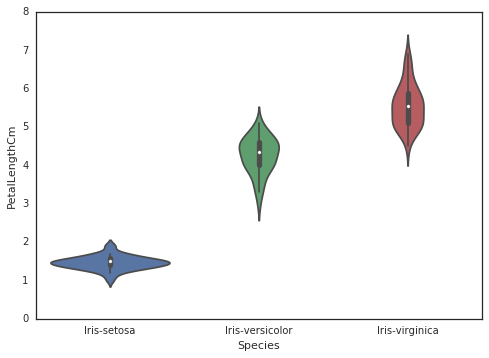
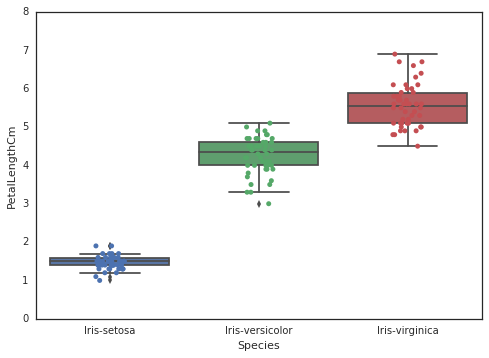
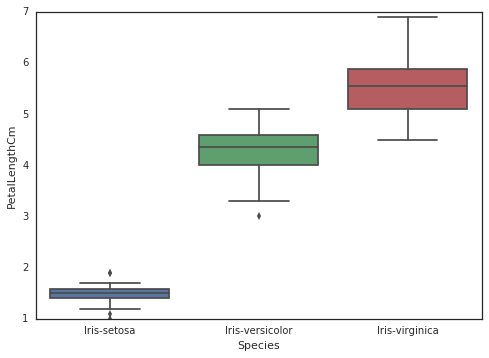
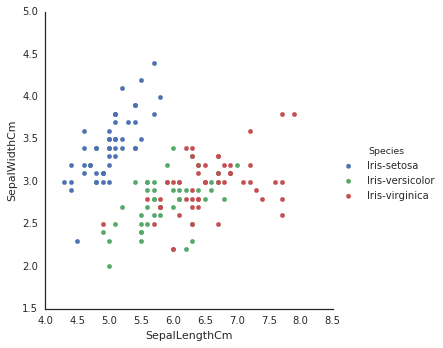
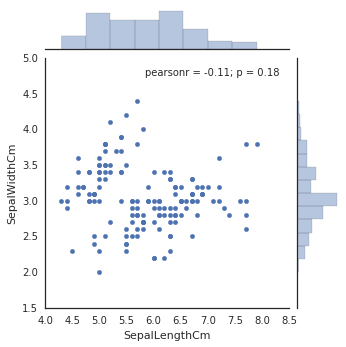
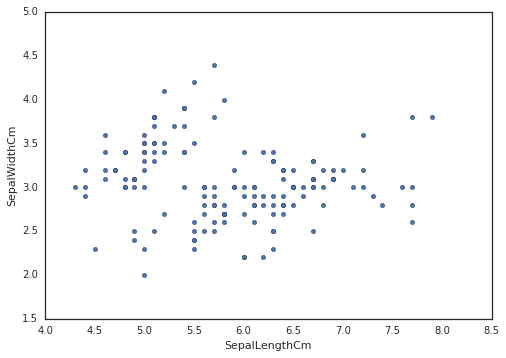
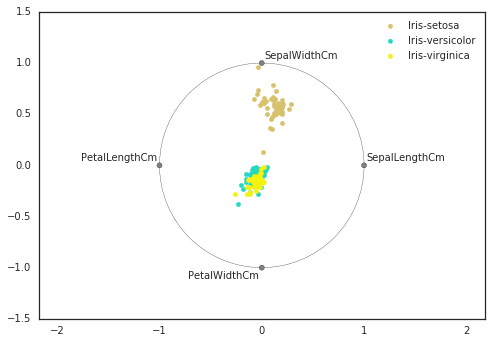
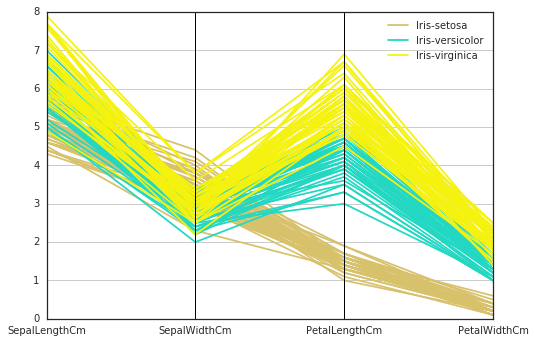
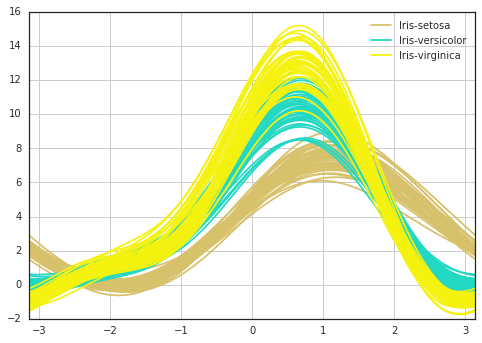
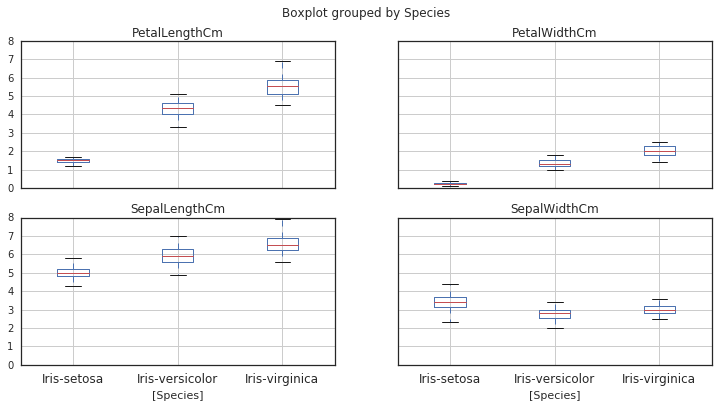
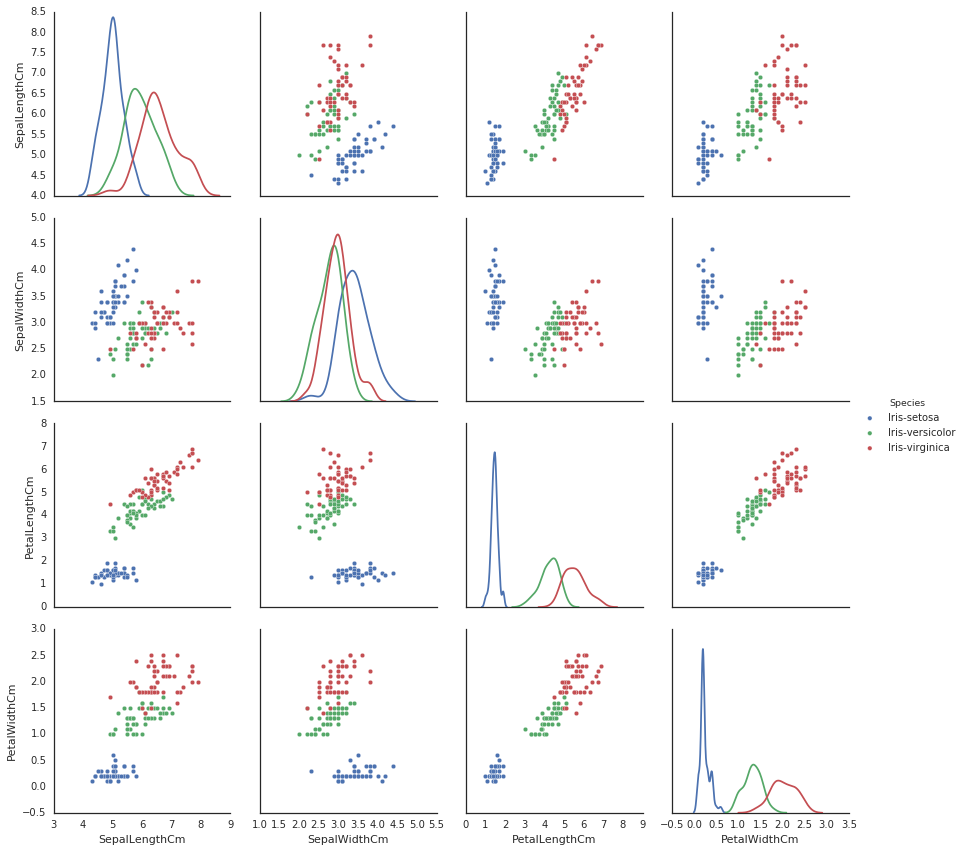
During training, it is often useful to measure the performance of the network as it attempts to find the optimal weight set. A common error measure or **cost function** used is sum-squared error. It is computed over all of the input vector/output vector pairs in the training set and is given by the equation below:



where p is the number of input/output vector pairs in the training set.

# Analysis

## Charting

* 
* 

# Confusion Matrix

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID3 Decicion tree | | Predicted | | |
| Setosa | Versicolour | Virginica |
| Actual class | Setosa | 14 | 0 | 0 |
| Versicolour | 0 | 17 | 0 |
| Virginica | 0 | 1 | 13 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Perceptron | | Predicted | | |
| Setosa | Versicolour | Virginica |
| Actual class | Setosa | 6 | 8 | 0 |
| Versicolour | 0 | 17 | 0 |
| Virginica | 0 | 2 | 12 |

# Conclusion

In the conclusion we trained our perceptron to distinguish between Iris flower types and constructed Decision Tree. At the same time project showed peculiarities and gave some Idea regarding how different are ID3 decision tree and Artificial neural network Perceptron.

Running Perceptron algorithm was 77.77 percent accurate with 35 correct classifications out of 45. ID3 algorithm 97,77 percent accuracy in contrast, was better showing 44 correct classifications with the same both testing data and training data.

# Python Code

#perceptron

from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import perceptron

from sklearn.datasets import load\_iris

iris = load\_iris()

f=iris.data

t=iris.target

q=0

k=0

i1=0

i2=0

i3=0

j1=0

j2=0

j3=0

#i needthose varaibles to calculate accuracy #and for confusion matrix

X\_train, X\_test, y\_train, y\_test = train\_test\_split(f, t, test\_size=0.3, random\_state=10)

#splitting training and test data .7 and .3

net = perceptron.Perceptron(n\_iter=1000, eta0=0.00001 )

net.fit(X\_train, y\_train)

#training our perceptron

for i in range (0,45):

#45 out of 150 is .3

k+=1#to test accuracy q/k\*100

if net.predict(X\_test[i]) == y\_test[i]:

q+=1#to test accuracy q/k\*100

else:

print(net.predict(X\_test[i]),y\_test[i])

#miss classifications

if net.predict(X\_test[i]) == 0:

i1+=1#numer of predicted Setosa

if net.predict(X\_test[i]) == 1:

i2+=1#numer of predicted Versicolour

if net.predict(X\_test[i]) == 2:

i3+=1#numer of predicted Virginica

if y\_test[i]==0:

j1+=1 #actual number of Setosa

if y\_test[i]==1:

j2+=1#actual number of Versicolour

if y\_test[i]==2:

j3+=1#actual number of Virginica

print(k,q,q/k\*100,i1,j1,i2,j2,i3,j3)#ID3 decicion tree

from sklearn.model\_selection import train\_test\_split

from sklearn import tree

from sklearn.datasets import load\_iris

iris = load\_iris()

f=iris.data

t=iris.target

q=0

k=0

i1=0

i2=0

i3=0

#i needthose varaibles to calculate accuracy #and for confusion matrix

X\_train, X\_test, y\_train, y\_test = train\_test\_split(f, t, test\_size=0.3, random\_state=10)

#splitting training and test data .7 and .3

clf = tree.DecisionTreeClassifier(criterion='entropy')

clf = clf.fit(X\_train, y\_train)

#training our Decision tree

with open("huitree.dot", 'w') as file:

h=tree.export\_graphviz(clf, out\_file=file)

#saving our Decision tree

for i in range (0,45):

#45 out of 150 is .3

k+=1#to test accuracy q/k\*100

if clf.predict(X\_test[i]) == y\_test[i]:

q+=1#to test accuracy q/k\*100

else:

print(clf.predict(X\_test[i]),y\_test[i])

#miss classifications

if clf.predict(X\_test[i]) == 0:

i1+=1#numer of predicted Setosa

if clf.predict(X\_test[i]) == 1:

i2+=1#numer of predicted Versicolour

if clf.predict(X\_test[i]) == 2:

i3+=1#numer of predicted Virginica

print(k,q,q/k\*100,i1,i2,i3)