The digits dataset result

	Misclassified samples	Accuracy	Running Time(seconds)
Perceptron	39	0.9278	0.0499
SVM (Linear)	12	0.9778	0.0444
SVM (Kernel)	10	0.9815	0.1023
Decision Tree	250	0.5370	0.0064
KNN	8	0.9852	0.0652
Logistic Regression	20	0.9630	0.1639

REALDISP Activity Recognition Dataset Data Set result

	Misclassified samples	Accuracy	Running Time(seconds)
Perceptron	875	0.9607	18.2811
SVM (Linear)	68	0.9969	44.6815
SVM (Kernel)	107	0.9952	91.6749
Decision Tree	7395	0.6677	3.5507
KNN	353	0.9841	154.2335
Logistic Regression	567	0.9745	149.8537

For the SVM Linear has the best result (misclassified samples) on digits dataset but the SVM kernel has the best result in REALISP. The reason may be the digits dataset is more linear separable. Also, the Decision Tree always gives the worst result but has the fastest running time due to low mathematic calculation. The Logistic Regression and KNN has more than 96% accuracy in both test but it toke the longest time. The high demand mathematic calculation requires a lot of process time but they are not as good as SVM.

For the Decision Tree, (Pre-Prunning Early Stopping Rule)

Two Strategies:
1. Minimum of instance is less than specific threshold
2. The node doesn't not improve impurity for a specific threshold. A node will be split if this split induces a decrease of the impurity greater than or equal to specific threshold.

In the main.py, Line 129
tree = DecisionTreeClassifier (criterion='gini',max_depth=4,random_state=1, min_samples_split = 4, min_impurity_decrease = 0.005)

I used the min_samples_split =4 for the minimum of instance strategy, the default is 0
I also used the min_impurity_decrease = 0.005, the default is 0.

The main.py takes 5 arguments for the extra data set:
1. Number of instance to calculate
2 Number of feature to calculate
3. The datafile name
4. index of column start for the feature
5. index of column where is the label located

Those arguments are only used for the extra data set.
If we want to run the test on digits dataset and READLISP, just type main.py 0 0 0 0 0 .