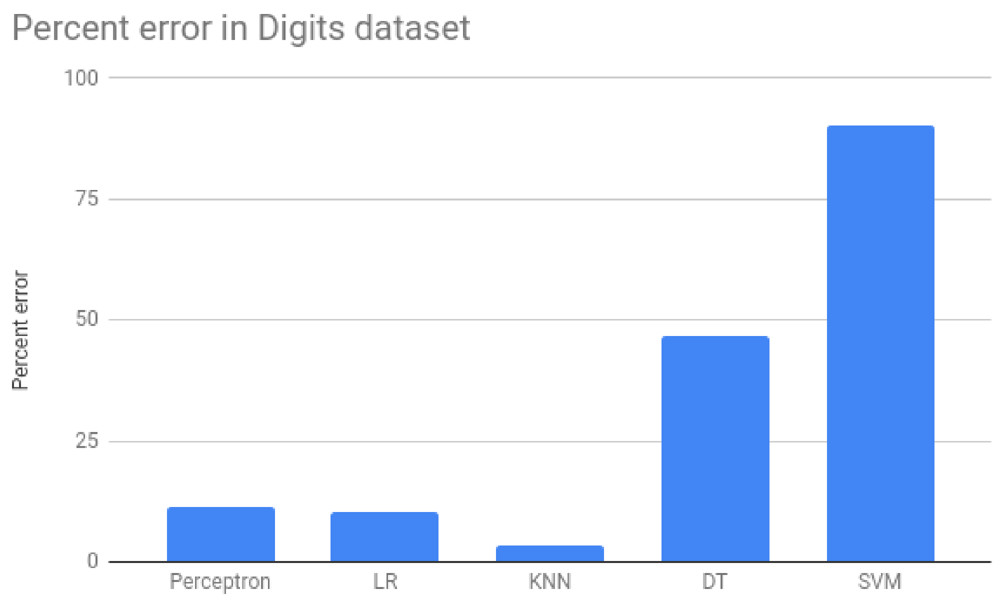
Stepheny Perez
CS 519 Project 3 Report

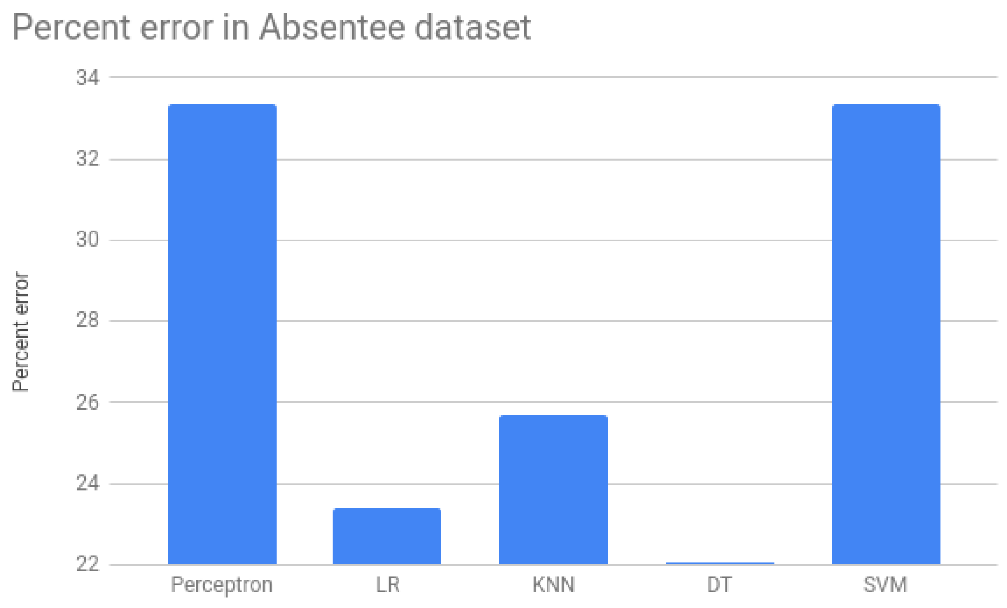
My chosen second dataset is the ‘Absenteeism at work’ dataset from the UCI machine learning repository. This was labeled as ‘Time-Series’ so I assume it follows the requirements of the assignment. There is additional information below on how I interpret the true class on this dataset.

Classifier Analysis

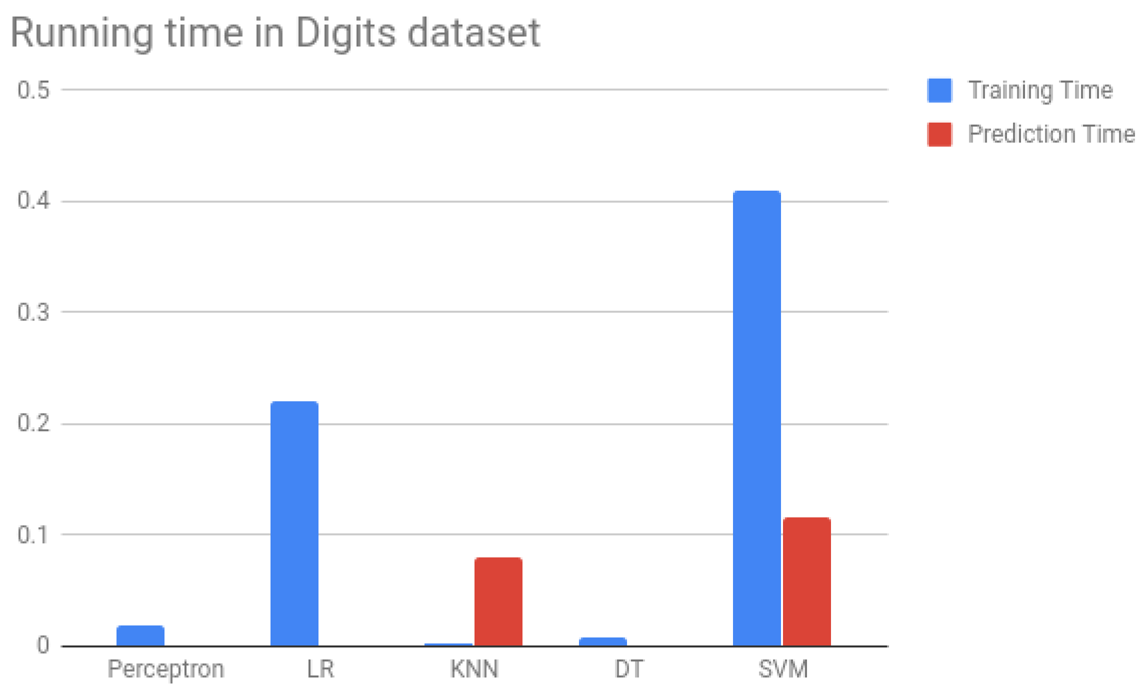
As you can see below, the Decision Tree and Simple Vector Machine classifiers behave very poorly on the digits dataset. Perceptron, LR, and KNN all behave well, with KNN behaving the best with only 3.5% error.

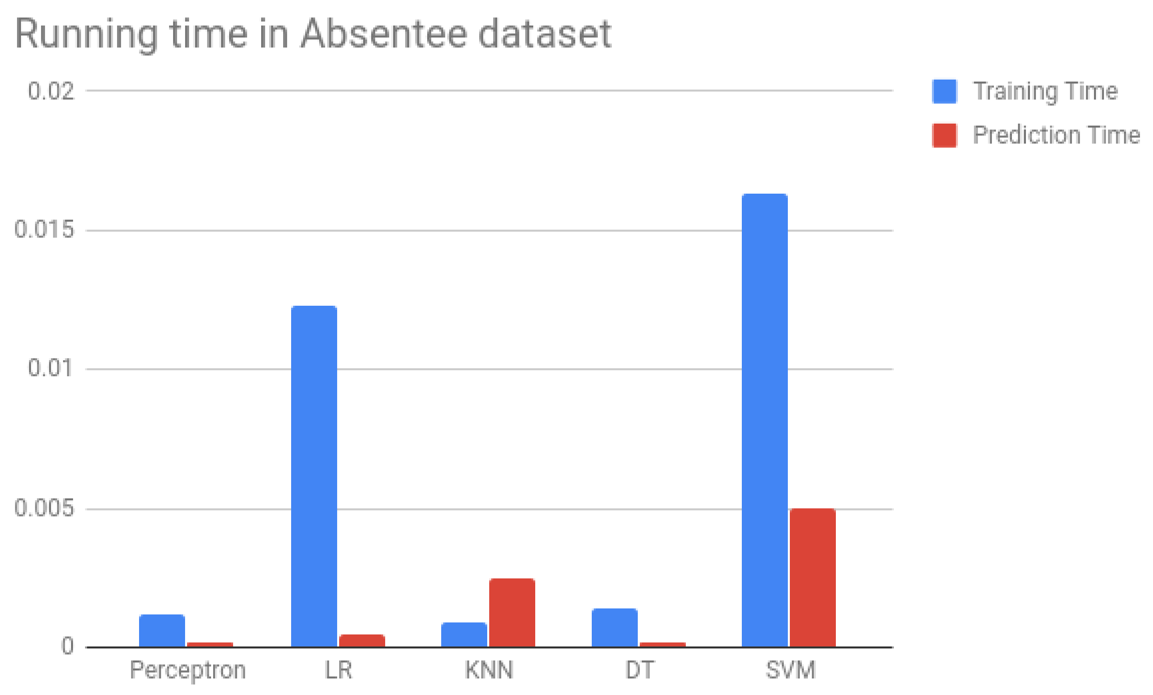


When run on the absentee dataset, all classifiers run fairly well. The axis may be deceiving, but the worst classifier (still SVM) has ~30% error rate, whereas on the digits dataset it was around ~90%. While I don’t know the exact reason, my guess is that the way I defined the ‘true class’ in the absentee data set is loose enough that the classification is more relaxed and therefore has higher results. Because the classification was not straightforward (it was # of hours absent, that could be anything from 0 to N) I classified into two categories: absent for less than 5 hours and absent for at least 5 hours. Perhaps with a more strict classification the performance would not be so high. My last theory for this huge difference is the content of the datasets. The absentee dataset has a large number of attributes, which may explain the high accuracy of the 5 classification algorithms



The runtime between the two datasets is pretty consistent -- SVM and LR both have very high training times in both datasets, and KNN exhibits behavior where the prediction time is higher than the training time. This is of course expected, because of the way KNN works (there is no training). Overall, DT and perceptron and consistently fastest so in time-conscious applications these would be the best solutions.





DecisionTreeClassifier Source Analysis

According to the class notes, extremely deep trees which lead to overfitting can be avoided by setting a limit for the maximum depth of the tree. I would consider this to be pre-pruning, because it's preventing the tree from ever reaching the state mentioned above. It is clearly stated in the sklearn docs that there is a parameter called max_depth. This value is used in the source code around line 171.

The second place pruning is used is similar to the first, and that's the “max_leaf_nodes” default parameter. As you can see on line 348, if no value is provided then DFS is used. With DFS, it's sometimes possible to have a very tall and unbalanced tree. With no max_leaf_nodes set and no max_depth set, it's possible to have a huge tree and overfitting issues.