Machine Learning for Big Data

Project by:

Moustafa Dafer B00772009
Ms450806@dal.ca

Data

We are using a dataset that contains tracking information for user transportation which include sampled coordinates of the trajectory traveled, userId, transportation mode, and timestamps. The goal of this project is to discover the potential of being able to predict new instances' transportation mode.

The transportation present in the dataset are: Bus, Car, Motorcycle, Run, Subway, Taxi, Train, and Walk.

I visualized the data on a map using Tableau just to be able to detect any visually obvious issues.

The completed tasks are as follows:

A- 1- Visualize the data and extract features grouped by UserId and Date: distance, speed, acceleration, and bearing
2- Create sub-trajectories by class using the daily trajectories and compute the trajectory features as follows:
* Discard sub-trajectories with less than 10 trajectory points.
* For each point feature, compute the minimum, maximum, mean, median and standard deviation. Those 20 values (5 statistical measures x 4-point features) are the trajectory features that should be used for classification.

3- Explore the data and compare the trajectory features values by class.

B- 1- After evaluating the trajectory features, we propose a hierarchy to classify the data. We then provide an overview of the groups we merged on each layer.
2- Compare the results of Decision Tree and Random Forest algorithms between hierarchical structure with and flat structure using ten-fold cross-validation with stratification.
3- Perform a multiclass evaluation and a significance test (e.g. paired t-test) for each classifier.

Part a

We start by exploring the data visually by projecting it on a map using Tableau:

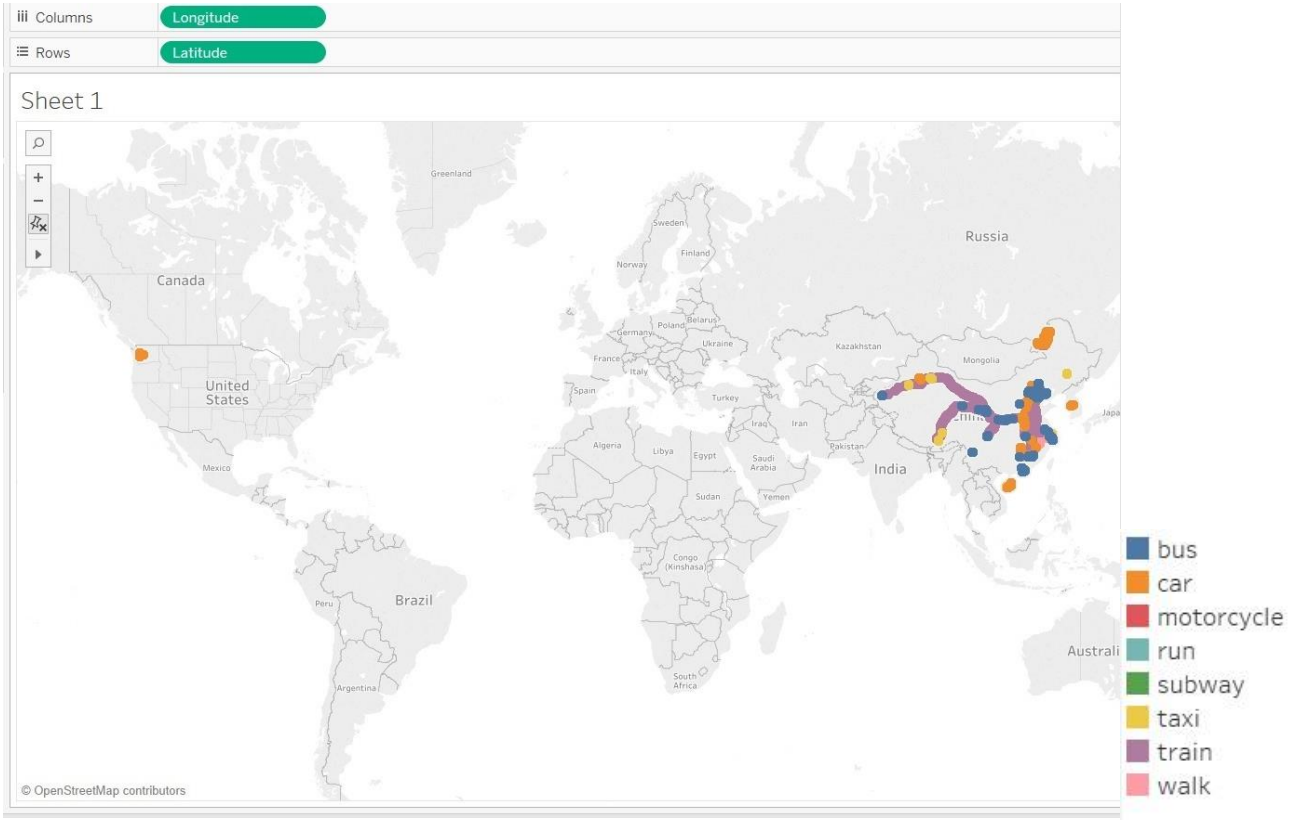

Figure 1: General Visualization of data on map

By filtering the data, we can find that there are under-presented transportation modes which may be considered as outliers:

Figure 2: Transportation Mode: run

Figure 3: Transportation Mode: motorcycle

By setting colors to represent unique users, we also find that there are time gaps for the same user:

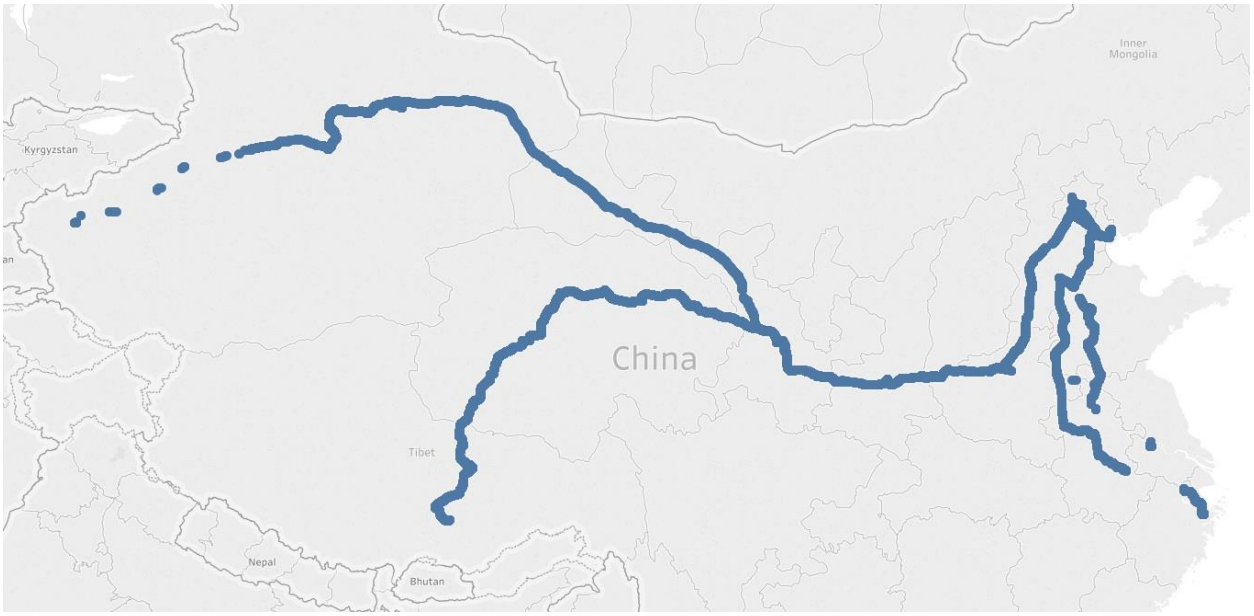

Figure 4 All points of user 10

These gaps can introduce a lot of noise if we group by date and userId without considering separate paths.

For example, assume that the following points were collected in the same day from user 10:

Figure 5: Time Gaps for User 10

Figure 6: Time Gaps Grouped by UserId and Date

As we can see, we cannot complete task A-1 simply by grouping-by date and userId, so we group by userId, date, and transportation mode first; we calculate the features and then we group the results by userId and date using summation and mean as necessary. It's also worth mentioning that even this enhanced way of grouping-by still introduces noise since there are trajectories traversing the boundaries of a day eg. 11:30PM to 12:05AM will be split into two trajectories; I propose to use time difference threshold instead of 24-hour day. Moreover, distance difference can be taken into consideration to split trajectories and preprocess data for better results.

Running the program generates the following files:

out_acc.csv: mean of acceleration per user per day

out_bearing: mean of bearing per user per day

out_speed: mean of speed per user per day

out_distance: total distance per user per day

out_time: total time difference per user per day

along with 2 preprocessing results and an out2.csv which contains the results of A-2.

In addition to the generated files, the program generates the following plots:

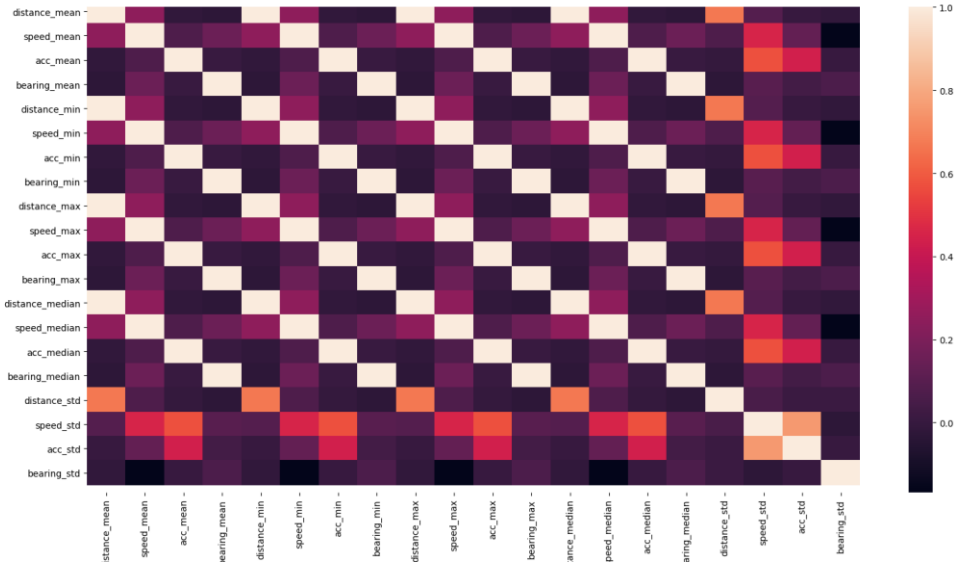

Figure 7: Features Correlation Heatmap

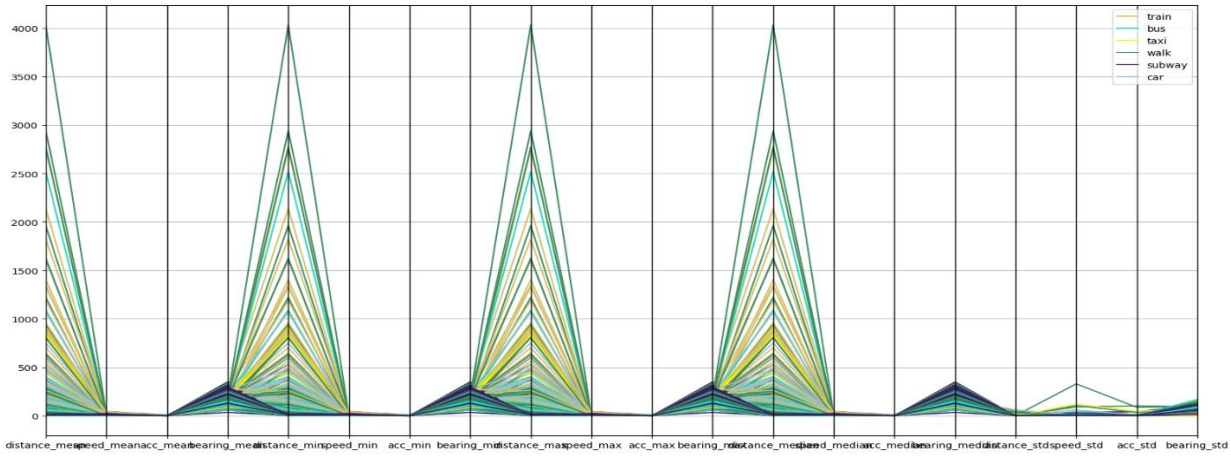

Figure 8: Parallel Coordinates for all features

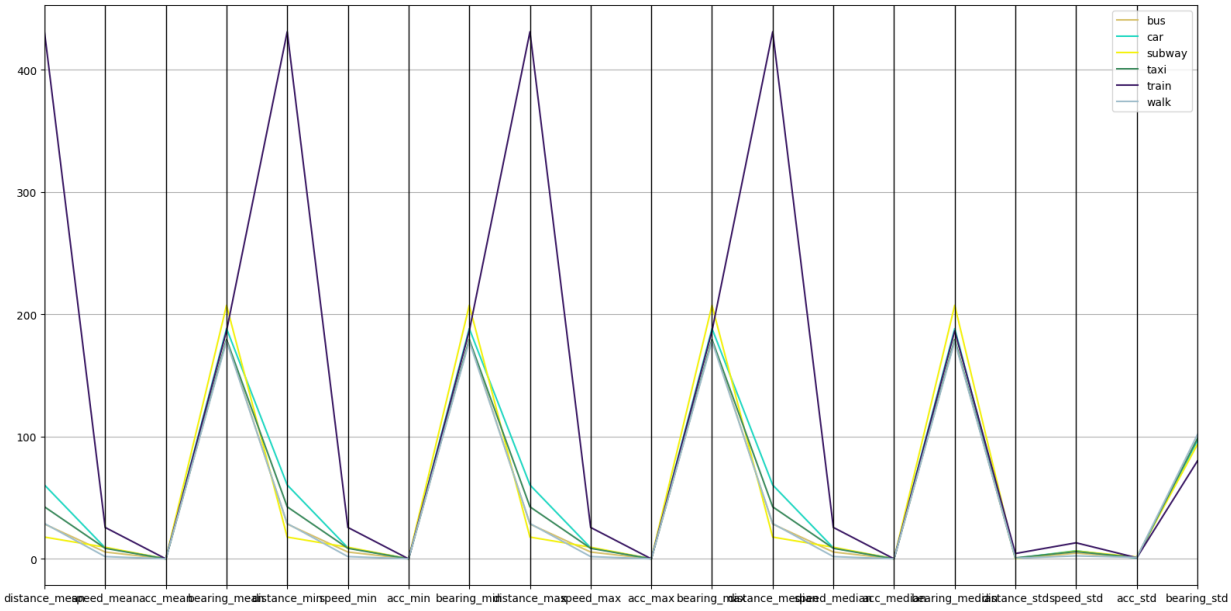

Figure 9: Parallel Coordinates of Means of Features

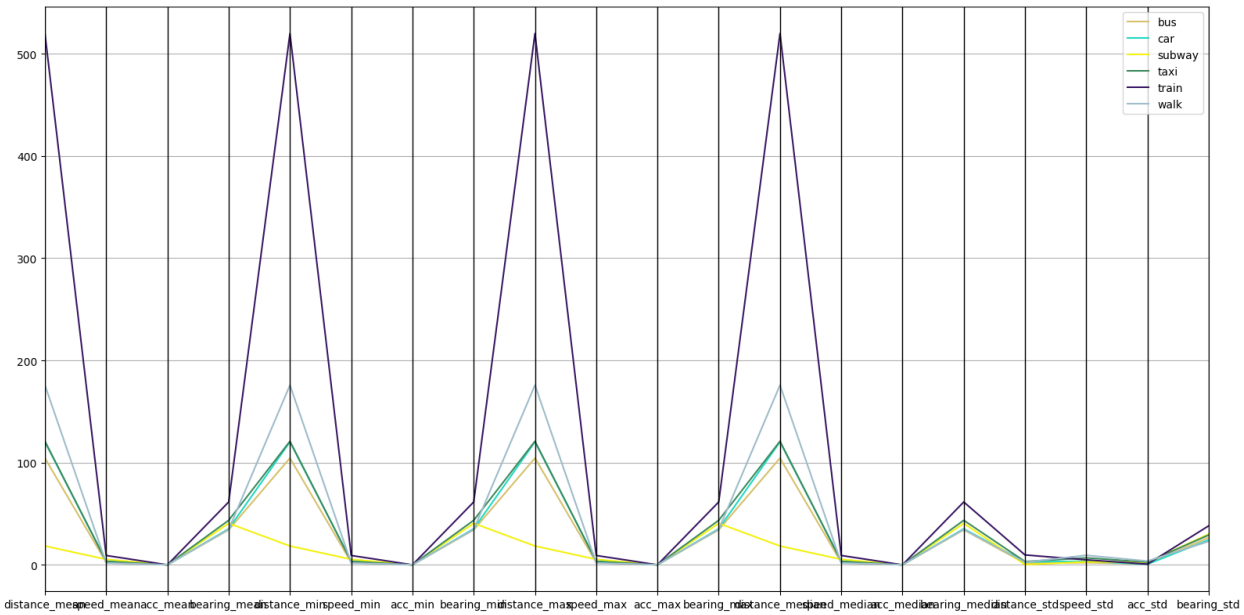

Figure 10: Parallel Coordinates of Std of Features

Part b

From figures 9 and 10, we can conclude that Train is clearly separable from other classes, the same thing goes for the unique pattern of subway, so we choose separating them as the root of our hierarchy, especially that they are on the opposite vertical ends of the plots.

Since the root identifies 2 classes, it dictates that the next level be the bag of Train and Subway; on the other hand, Figure 10 shows that the standard deviation for distance walked is noticeably greater than that travelled by other means, so we set the other part of level2 as Walk or other.

Because the plots show a close relation between the remaining classes, I set the 3rd level to bag all 3 remaining classes.

So our Hierarchy can be represented as follows:

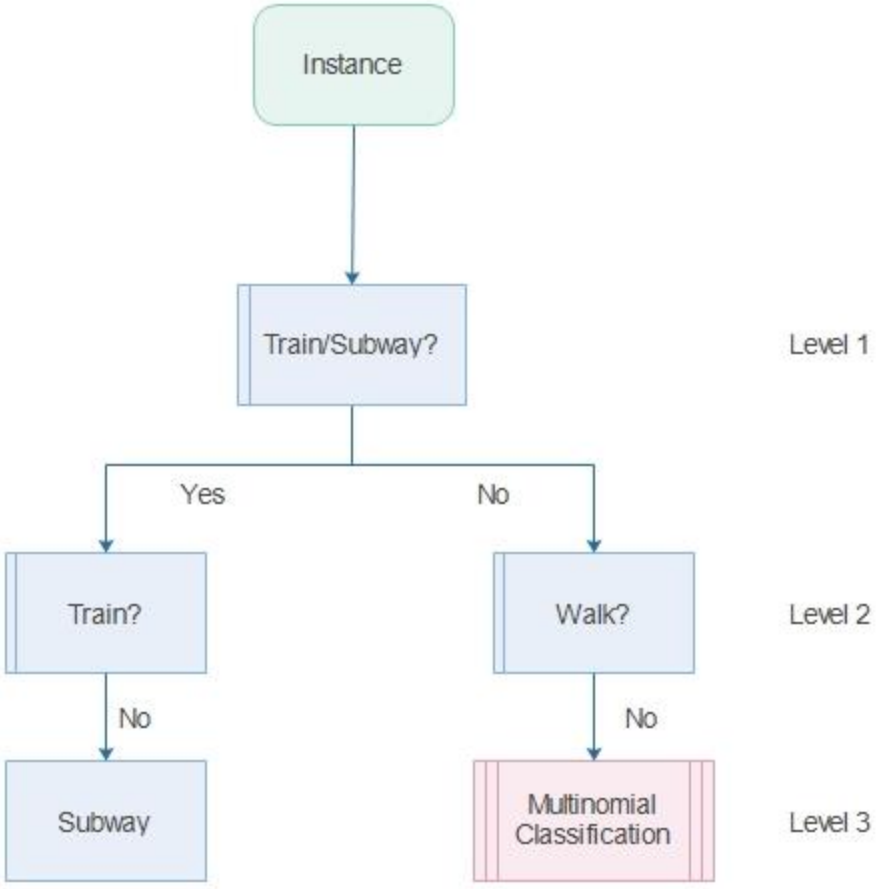

Figure 11: Proposed Hierarchy

By comparing the classification accuracy between the Hierarchical and the flat approach we get the following results:

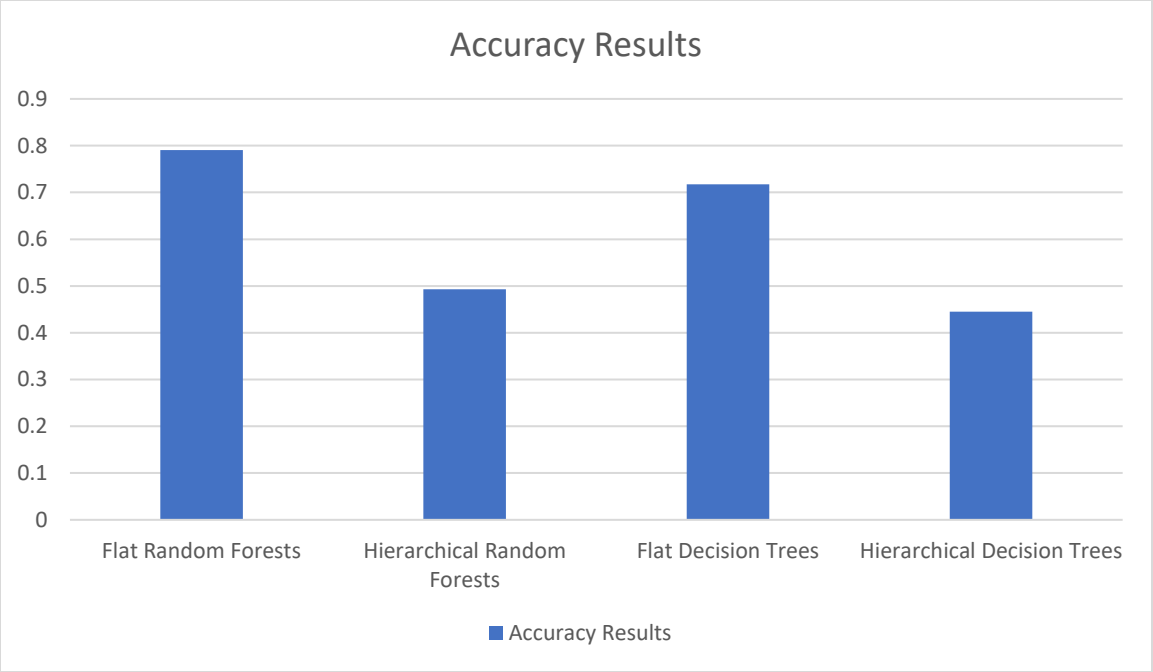

Figure 12: Accuracy of Classifiers

As expected, Random forests performed better than decision trees in both flat and hierarchical implementations, this is because decision trees are ensembles; However, flat algorithms performed better than hierarchical ones; this is also expected because both decision trees and random forests are hierarchical by nature, so after all, all 4 implementations are actually hierarchical, this leads us to deduce that there is a better hierarchy that we could have proposed.

We then run T-Tests between flat and hierarchical implementations of same algorithms and the results are as follows:

Classifier	P-value	Null Hypothesis (a=0.05)
Random Forests	1.1266482e-13	rejected
Decision Trees	4.7223396e-17	rejected

Table 1: T-Tests between flat and hierarchical implementations

I also tried to explore the dataset furthermore by testing other classifiers using Weka and below are the results:

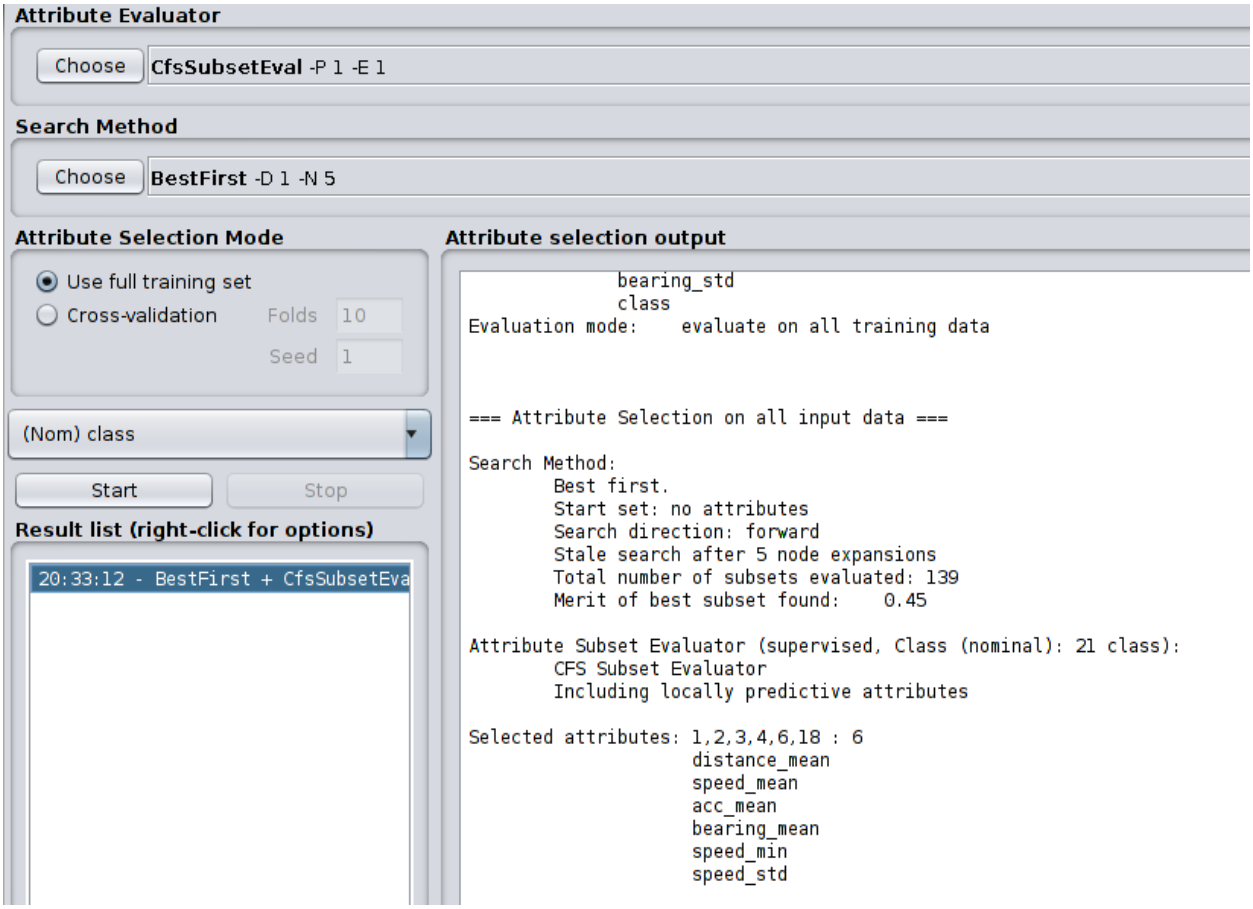

Figure 13: Suggested Best 6 Attributes

Figure 13 suggests that some attributes may be introducing noise, the best 6 attributes are shown in the figure according to the CfsSubset Evaluator using BestFirst method.

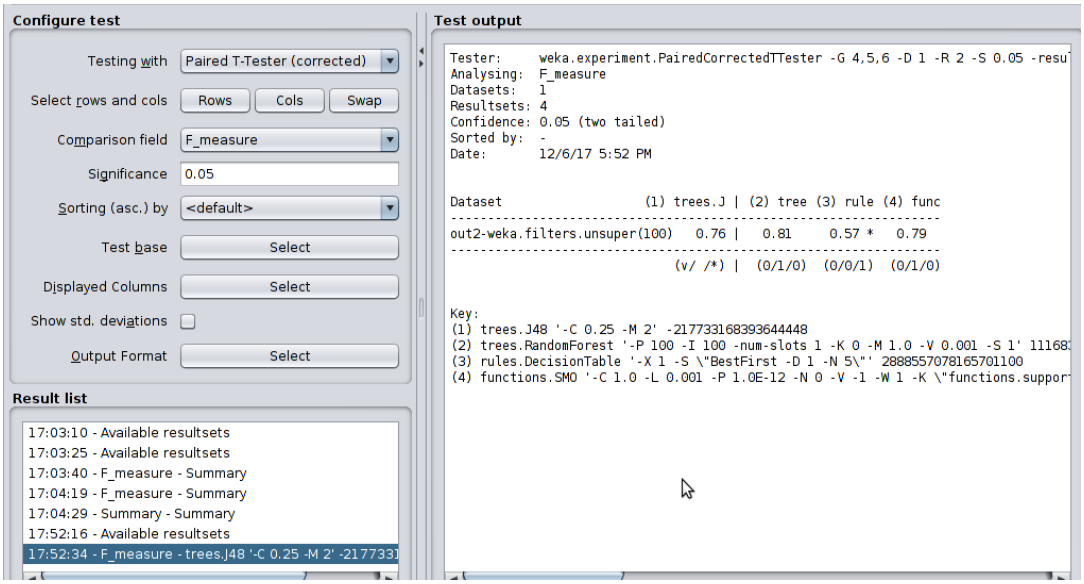

Figure 14: F-Measure of several classifiers

Figure 14 shows that Random Forests performed better than Decision Trees, Decision Table, and SVM.

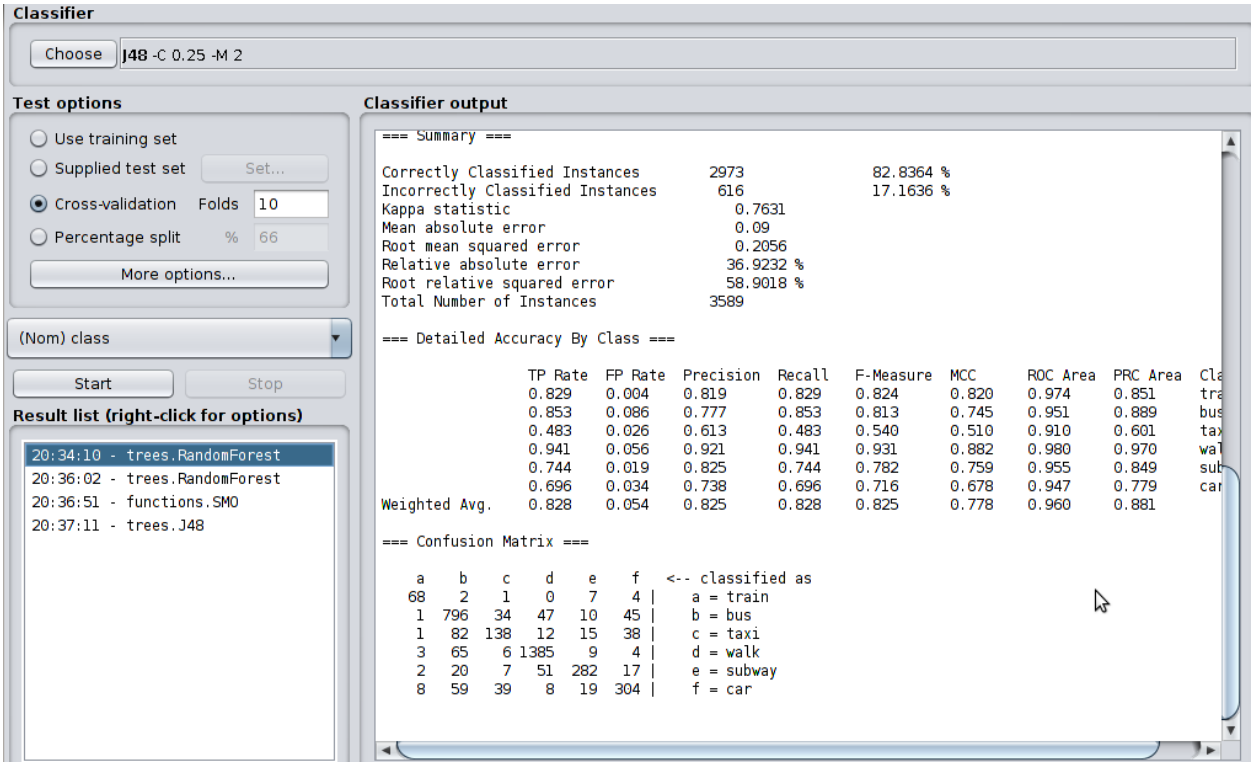

Figure 15: J48 results after trimming some attributes

Figure 15 Shows that trimming some attributes actually improves the performance of the classifiers.