Ordinal Classification Implementation Report

# The Implementation

We implemented the meta algorithm in python using pandas library, we’re assuming receiving dataset and labels for training as pandas tables. We created a class called OrdinalClassifier that receives a classifier and a list or ordered classes, an assumption has been made that this classifier has the method ‘predict\_proba’ because the need of using the probabilities of the different classes in the paper. The OrdinalClassifier has the methods fit(X,y), predict(X), score(X,y) and generateDatasets(y).

fit(X,y) – as regular fit methods in python models, fit the model, in our case it create n-1 datasets with different target values for each classifier as suggested in the paper, fit and keep n-1 models of the estimator with the datasets created, one model for each of the first n-1 classes (the last one can be derived from the results of the previous).

predict(X) – predicts the probabilities of all the fitted models, return the class of the one that gave the best probability.

score(X,y) – a utility method that uses the predict method to automatically calculate the accuracy of the model on given dataset and labels

generateDatasets(y) – create n-1 labels for n-1 datasets, each iteration take the first subset of classes until the index and change their labels to 0, the others to 1.

# How to Run

The code is attached and can also be found in https://github.com/ofrik/ordinal\_classification

In order to use the code of the OrdinalClassifier you should import the class to your code and use the interface described above.

In order to run the comparison, you should execute ‘python OrdinalClassifier\_comparison.py’ in the folder of the file. It assumes you have sklearn, pandas and numpy.

# Results

We compared the Ordinal Classifier with the same classifier we gave him as an estimator to see if the use of the ordinality in the classes improve the results, we used SVC, Random Forest and Multi-Layer Perceptron for that.

As datasets, we used 5 datasets with a numeric target column, we split it into 3 equal parts according to the range of values and changed the original target values to these ordinal categories. In our evaluation of the algorithms we used 5-fold cross validation, all the metrics are in average.

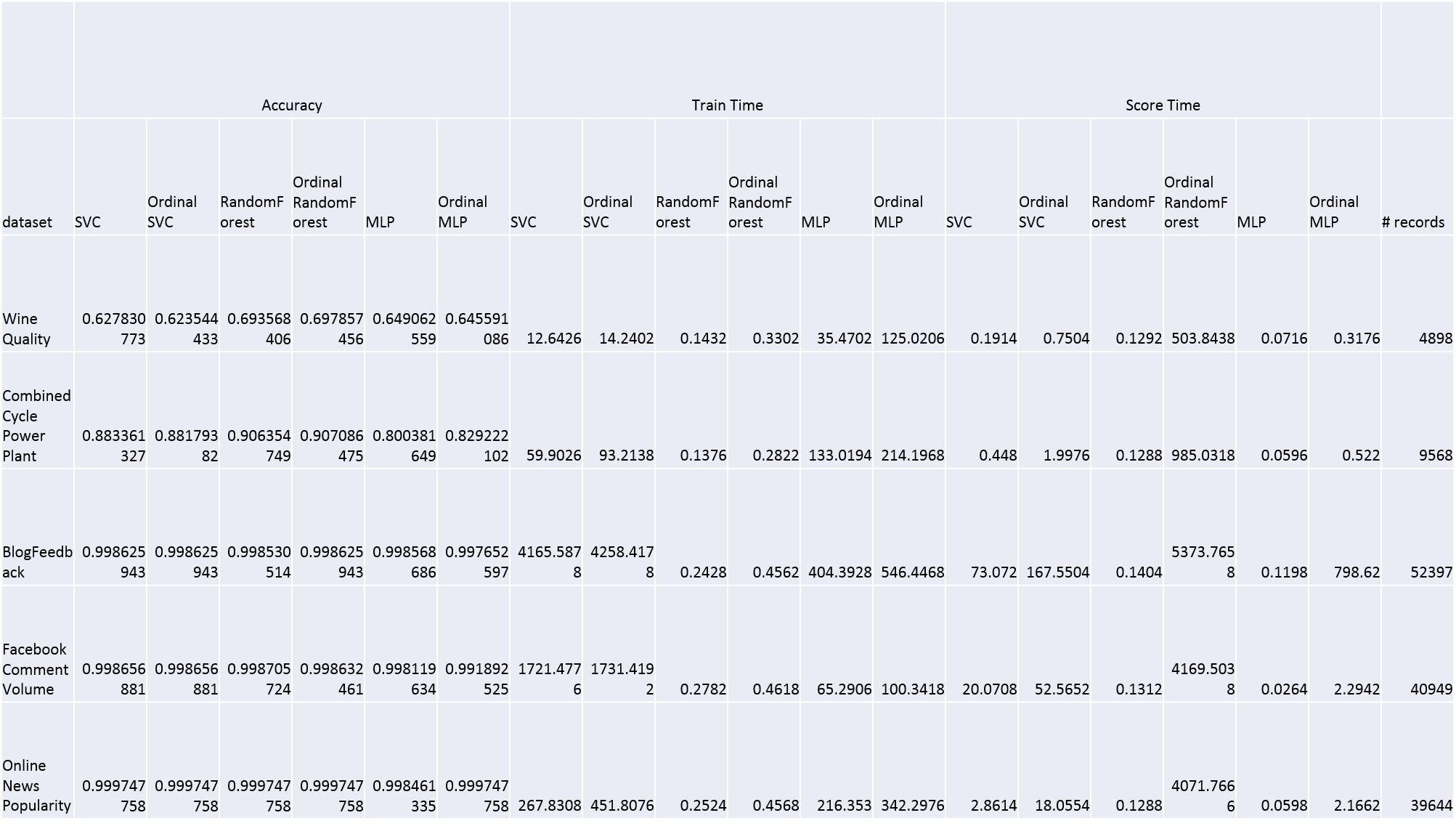


Figure 1 the raw results

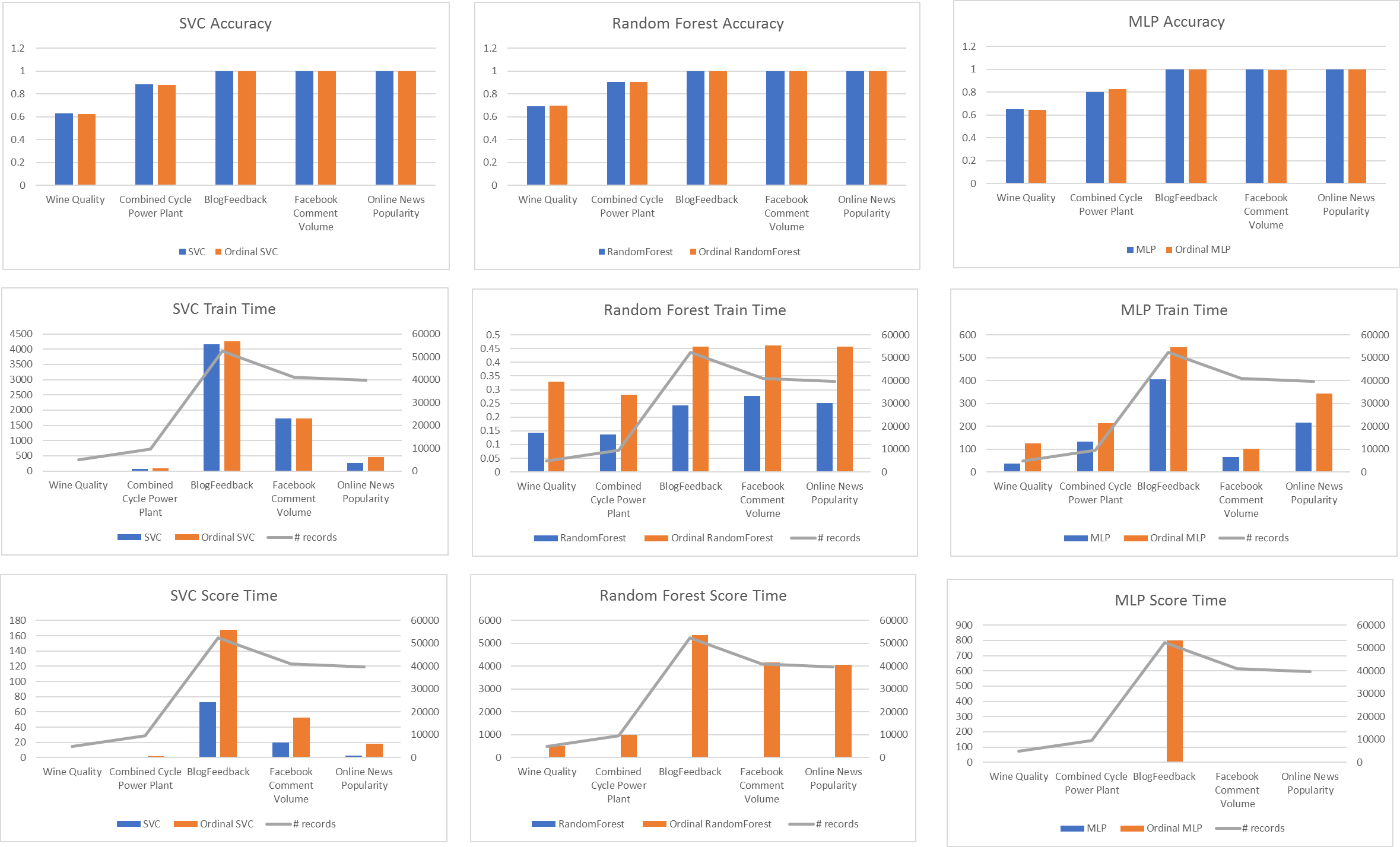


Figure 2 diagrams of the different classifiers and matching ordinal classifiers with different measures

# Conclusions

We can see the ordinal classifier does not perform better in terms of accuracy in most of the cases, but always stay close to the original classifier, from below or above. In terms of train time we can see that, as expected, the ordinal classifier takes more time to train. but, not as bad as we thought, as it’s train time is less than 3 times the original, while still train 3 classifiers instead of one. The score time was exceptionally slow, much more than we expected, in some of the datasets and algorithms the ordinal classifier took more than 10 times to calculate the score than the original classifier. As a conclusion, the this ordinal meta classifier does not improve the accuracy enough in most cases, in the cases it does, in most of the cases does not “worth” the extra train and prediction time.