

Ordinal Classification

Background

In this exercise we implemented a Meta algorithm which conducts ordinal classification.

The implementation was carried out in Python 2.7, and was based on the article:

Frank, Eibe, and Mark Hall. "A simple approach to ordinal classification." *European Conference on Machine Learning*. Springer Berlin Heidelberg, 2001.

The Data Sets:

We used 5 datasets in our project:

#	Data Set	# instances	# features	target	order	Data
1	The Boston dataset	506	14	house prices	["low", "medium", "high"]	Concerns housing values in suburbs of Boston.
2	The Facebook metrics dataset	500	19	Total Interactions	["low", "medium", "high"]	Facebook performance metrics of a renowned cosmetic's brand Facebook page.
3	The Bike Sharing dataset	17389	16	daily bike rental count	["low", "medium", "high"]	hourly and daily count of rental bikes between years 2011 and 2012 in Capital bikeshare system with the corresponding weather and seasonal information.
4	The Online News Popularity dataset	39797	61	number of shares in social networks	["low", "medium", "high"]	heterogeneous set of features about articles published by Mashable in a period of two years.
5	The Student Grades dataset	649	33	student performance	["low", "medium", "high"]	student achievement in secondary education of two Portuguese schools. The data attributes include student grades, demographic, social and school related features.

Data set links:

#	Data Set	link
1	The Boston dataset	https://archive.ics.uci.edu/ml/datasets/housing
2	The Facebook metrics dataset	https://archive.ics.uci.edu/ml/datasets/Facebook+metrics
3	The Bike Sharing dataset	https://archive.ics.uci.edu/ml/datasets/Bike+Sharing+Dataset
4	The Online News Popularity dataset	https://archive.ics.uci.edu/ml/datasets/Online+News+Popularity
5	The Student Grades dataset	https://archive.ics.uci.edu/ml/datasets/Student+Performance

Our Project

The program consists of the following files

1. **"OrdinalClassifier.py"** - A Python file which consists of 2 classes:
 - The ordinal classifier
 - The binary classifier (regular classifier for ordinal data)

2. **"Utils.py"** - A Python file which consists of helper functions and Dataset load functions
3. **"test.py"** - A Python file which holds the main program.
4. **"datasets"** - A folder with csv files of 4 datasets.
5. **"results.csv"** - A csv file with the accuracy results of both types of classifiers on 5 datasets using 3 classifiers: Gradient Boosting, Random Forest, CART Decision Tree.

Every instance in a chosen dataset has a number of features and a target to classify. Initially the target value was numeric and we transformed it to an ordinal type by a specific order for that dataset with the "pandas.qcut" command. An example for an order is: ["Low", "Medium", "High"]

Every dataset was shuffled and split into a train (70%) and test (30%) dataset.

We built two main classifier classes: Ordinal classifier, Binary classifier. The ordinal classifier takes into account the order of the classes while the binary classifier doesn't. Each main classifier is initiated with the chosen dataset and with a secondary classifier and its parameters. The secondary classifier is one of the next classifiers:

- Gradient Boosting
- Random Forest
- CART Decision Tree

The Ordinal classifier has three stages:

- a. Fit the train data by the chosen model (Gradient Boosting, Random Forest, CART) First, the original ordinal target data $\mathcal{Y}()$ with k degrees ($1, 2 \dots k$) is transformed into a binary array of $k-1$ classes.

$$class\ i = \begin{cases} 1 & y > i \\ 0 & else \end{cases} \quad 1 \leq i < k$$

Secondly, A secondary classifier model is fitted for every class i .

This stage results in k models which all give a probability for an instance to belong to a certain class i .

- b. First, we predict the probability for every instance in the test dataset to belong to every class i (this means that the ordinal target is bigger than i).

test instances	y>low	y>midium
1	0.9	0.5
2	0.4	0.1
3	0.8	0.1


Secondly, we predict the probability of an instance x to belong to the original ordinal target data class ($y \in \{1, 2 \dots k\}$).

$$P(y = i) = \begin{cases} 1 - P(y > 1) & i = 1 \\ P(y > i - 1) - P(y > i) & 1 < i < k \\ P(y > k - 1) & i = k \end{cases}$$

test instances	low	midium	high
1	0.1	0.4	0.5
2	0.6	0.3	0.1
3	0.2	0.7	0.1

- c. In the last stage we assign every instance in the test dataset to the original ordinal target data class which scored the highest probability in phase b.

test instances	low	midium	high
1	0.1	0.4	0.5
2	0.6	0.3	0.1
3	0.2	0.7	0.1



test instances	low	midium	high
1	0	0	1
2	1	0	0
3	0	1	0

The Binary classifier has three stages:

- a. Fit the train data by the chosen model (Gradient Boosting, Random Forest, CART) First, the original ordinal target data (\mathcal{Y}) with k degrees ($1, 2, \dots, k$) is transformed into a binary array of k classes.

$$class\ i = \begin{cases} 1 & \text{if } y = i \\ 0 & \text{else} \end{cases} \quad 1 \leq i \leq k$$

Secondly, A secondary classifier model is fitted for every class i .


This stage results in k models which all give a probability for an instance to belong to a certain class i .

- b. We predict the probability for every instance in the test dataset to belong to every class i (this means that the ordinal target equals i).

test instances	low	midium	high
1	0.1	0.4	0.5
2	0.6	0.3	0.1
3	0.2	0.7	0.1

- c. In the last stage we assign every instance in the test dataset to the original ordinal target data class which scored the highest probability in phase b.

test instances	low	midium	high
1	0.1	0.4	0.5
2	0.6	0.3	0.1
3	0.2	0.7	0.1



test instances	low	midium	high
1	0	0	1
2	1	0	0
3	0	1	0

Evaluation

For each of the 5 datasets, we ran the Ordinal and Binary main classifiers. Each main classifier fitted the data to every class using 3 classifiers: Gradient Boosting, Random Forest, CART Decision Tree. For every model prediction (main classifier, secondary classifier, dataset) an accuracy score was calculated:

- Let n be the number of instances in the chosen dataset.
- Let \hat{y}_i be the predicted ordinal target value for the i 'th instance.
- Let y_i be the true ordinal target value for the i 'th instance.

$$accuracy(\hat{y}, y) = \frac{1}{n} \sum_{i=0}^{n-1} 1(\hat{y}_i = y_i)$$

We ran the program twice with different ordinal ranges:

1. Range [0,1,2], (this order will be referred to as "order_3")
2. Range [0,1,2,3,4], (this order will be referred to as "order_5")

We evaluated the Ordinal classifier by comparing its accuracy scores to that of the Binary classifier.

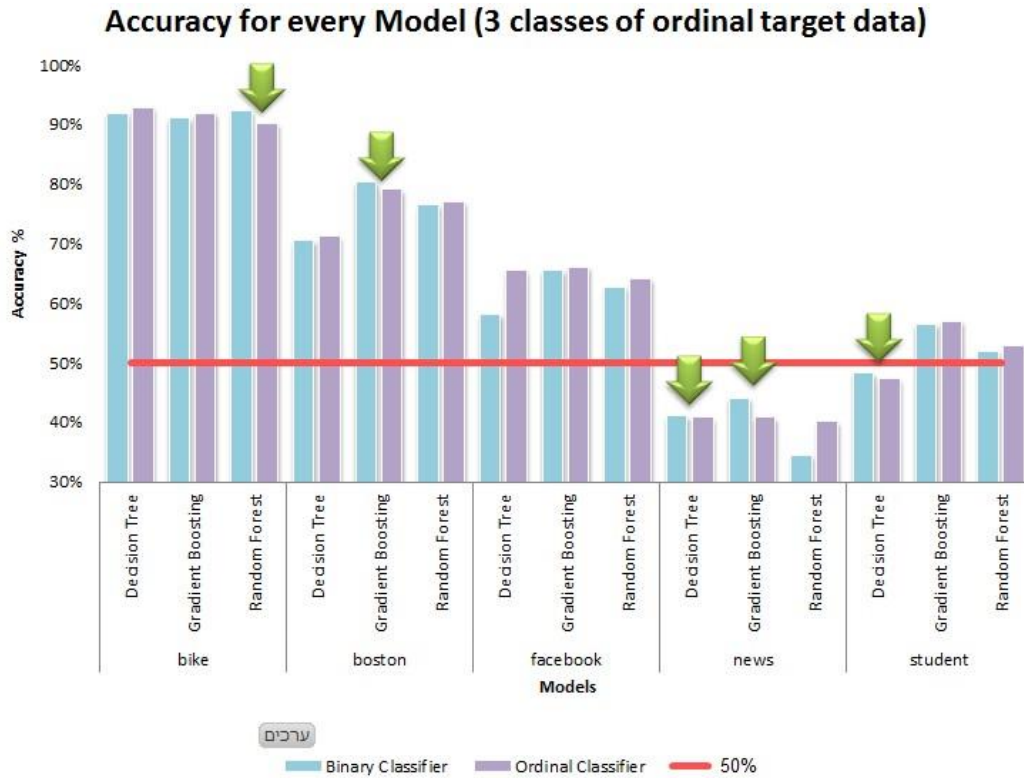
Results

The table below presents the results of our project with **order_3** :

#	Binary classifier accuracy	Ordinal classifier accuracy	dataset	secondary classifier
0	80.26%	78.95%	boston	Gradient Boosting
1	76.32%	76.97%	boston	Random Forest
2	70.39%	71.05%	boston	Decision Tree
3	65.33%	66.00%	facebook	Gradient Boosting
4	62.67%	64.00%	facebook	Random Forest
5	58.00%	65.33%	facebook	Decision Tree
6	90.91%	91.82%	bike	Gradient Boosting
7	92.27%	90.00%	bike	Random Forest
8	91.82%	92.73%	bike	Decision Tree
9	44.00%	40.67%	news	Gradient Boosting
10	34.33%	40.00%	news	Random Forest
11	41.00%	40.67%	news	Decision Tree
12	56.41%	56.92%	student	Gradient Boosting
13	51.79%	52.82%	student	Random Forest
14	48.21%	47.18%	student	Decision Tree

** Values in bold and yellow represent the best accuracy scores.

The chart below presents the results of our project with **order_3** :



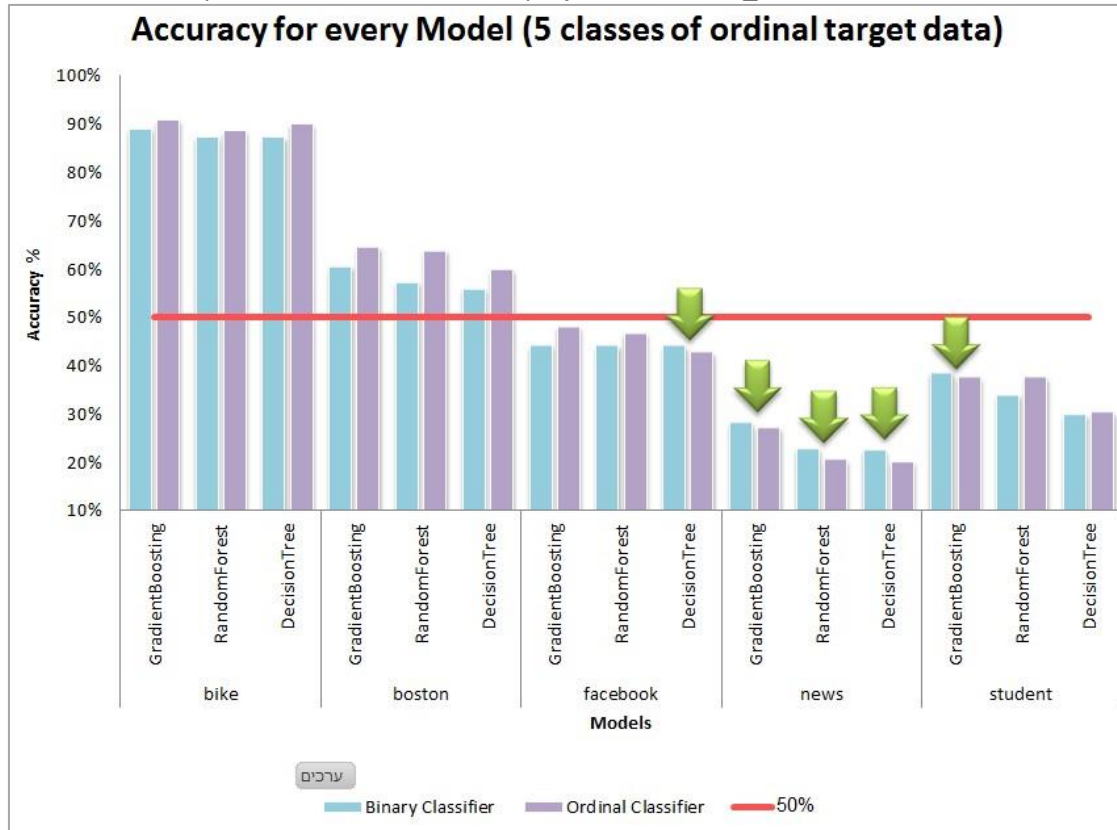
** Green arrows indicate that the binary classifier's accuracy was better than the Ordinal classifier's accuracy.

The results of our project with **order_5**:

#	Binary classifier accuracy	Ordinal classifier accuracy	dataset	secondary classifier
0	60.53%	64.47%	boston	Gradient Boosting
1	57.24%	63.82%	boston	Random Forest
2	55.92%	59.87%	boston	Decision Tree
3	44.00%	48.00%	facebook	Gradient Boosting
4	44.00%	46.67%	facebook	Random Forest
5	44.00%	42.67%	facebook	Decision Tree
6	89.09%	90.91%	bike	Gradient Boosting
7	87.27%	88.64%	bike	Random Forest
8	87.27%	90.00%	bike	Decision Tree
9	28.00%	27.00%	news	Gradient Boosting
10	22.67%	20.33%	news	Random Forest
11	22.33%	20.00%	news	Decision Tree
12	38.46%	37.44%	student	Gradient Boosting
13	33.85%	37.44%	student	Random Forest
14	29.74%	30.26%	student	Decision Tree

** Values in bold and yellow represent the best accuracy scores.

The chart below presents the results of our project with **order_5** :



**** Green arrows indicate that the binary classifier's accuracy was better than the Ordinal classifier's accuracy.**

For both the **order_3** and for the **order_5** runs, 10 out of 15 models scored higher accuracy using the Ordinal classifier (over the binary classifier).

We note that for the **order_3** run the differences between the accuracy scores of most of the models are low (less than 5% change). For the models: (Facebook, Decision Tree) and (news, Random Forest) the accuracy difference between the Ordinal and the Binary classifier is larger than 5% in favor of the ordinal classifier.

In **order_3** the highest accuracy score for the news dataset is 44% (Binary classifier with Gradient Boosting).

We note that for the **order_5** run the differences between the accuracy scores of most of the models are low (less than 5% change). Only the model: (Boston, Random Forest) exhibits an accuracy difference between the Ordinal and the Binary classifier that is larger than 5% in favor of the ordinal classifier.

In **order_3** the highest accuracy score for the news dataset is 28% (Binary classifier with Gradient Boosting).

All of the models scored a higher accuracy score in the **order_3** run than the **order_5** run.

The Table below shows the runtime in seconds for every model with order_5:

Data set	Secondary classifier	Run time Ordinal classifier (sec)	Run time Binary classifier (sec)	difference (sec)
boston	Gradient Boosting	0.074	0.113	0.039
boston	Random Forest	0.432	0.487	0.055
boston	Decision Tree	0.019	0.038	0.019
facebook	Gradient Boosting	0.074	0.118	0.044
facebook	Random Forest	0.378	0.428	0.050
facebook	Decision Tree	0.015	0.035	0.020
bike	Gradient Boosting	0.082	0.148	0.066
bike	Random Forest	0.401	0.459	0.058
bike	Decision Tree	0.019	0.049	0.030
news	Gradient Boosting	0.264	0.330	0.066
news	Random Forest	0.562	0.606	0.044
news	Decision Tree	0.097	0.157	0.060
student	Gradient Boosting	0.085	0.125	0.040
student	Random Forest	0.405	0.477	0.072
student	Decision Tree	0.017	0.044	0.027

** Values in bold and yellow represent longest run time.

Conclusions & Discussion

For most of the models (10 out of 15) the Ordinal classifier outperformed the Binary classifier.

As described in the results, the differences between the accuracy of the main classifiers were low for the majority of the models (<5% accuracy difference). This can be related to poor model parameters which weakened our initial classifier models in both main classifiers. We suggest that more experiments should be conducted with Hyper-parameter optimization for the secondary classifiers. We used the same secondary classifier parameters for all the datasets. In future work we propose to adjust the parameters to every dataset and gain higher accuracy.

As evident from the results all of the accuracy scores of the order_3 run were higher than the order_5 run. It is possible that an ordinal range of three is more fitting for the target values than an ordinal range of five for the chosen datasets. We believe that more data sets should be tested. It is preferred that the target value of those data sets would be ordinal from the start.

In both order_3 and order_5 runs the accuracy of all the news dataset's models were below 50%. This might indicate that the ordinal range used was insufficient. Another possibility is that more/better features are needed for the classification task.

The runtime of the Binary classifier is longer than the Ordinal classifier for all the models in the order_5 run. The differences are very small (less than a tenth of a second). This can be attributed to our implementation of the multiclass classifier. We suspect that using the built id multiclass classifier in the secondary classifiers might speed up this run time.

Even though the accuracy differences were low (<5% accuracy difference) they still exist. Therefore, we conclude that for the ordinal classification task it is better to use the Ordinal classifier than the Binary classifier. This makes sense because the ordinal classifier takes in to account more information regarding the target set (which is the order of the class data).