David Shvimer
Z23430635

Module Order Models

Introduction

In this experiment we test attribute selection methods for Module Order Models (MOM). MOM is a technique that orders modules by some predicted value. MOM is useful when one does not need a predicted value, but needs a set of modules ranked based on that predicted value. When we look at the data relative to itself, we can classify subsets of the data. We will be working with a dataset that describes software modules and attempt to predict which ones have the most faults. A model that can achieve this is useful for any company developing software. A manager can decide where to place efforts to improve code hygiene based on which modules are the most likely to be ridden with bugs. We will test the M5 and Greedy attribute selection methods to see if one performs best with MOM. First, we will predict the number of faults in modules in fit and test sets. Next, we will order the modules and evaluate them using Alberg diagrams and check the performance of each model.

Evaluation

The following two Linear Regression models were used:

M5
FAULTS = - 0.0516 * NUMUORS + 0.0341 * NUMUANDS - 0.0027 * TOTOTORS - 0.0372 * VG + 0.2119 * NLOGIC + 0.0018 * LOC + 0.005 * ELOC - 0.3091

Greedy
FAULTS = - 0.0482 * NUMUORS + 0.0336 * NUMUANDS - 0.0021 * TOTOTORS - 0.0337 * VG + 0.2088 * NLOGIC + 0.0019 * LOC - 0.3255

For both the fit and test set, we predict the faults of software modules with both models and obtain results that look like the following (only a subset of the data is shown):

	NUMUORS	NUMUANDS	TOTOTORS	TOTOPANDS	VG	NLOGIC	LOC	ELOC	FAULTS	F_HAT_M5	F_HAT_GREEDY
187	63.0	633.0	4180.0	3748.0	145.0	30.0	3991.0	607.0	29.0	17.9212	18.0891
186	66.0	1124.0	8606.0	7736.0	448.0	31.0	9163.0	1412.0	29.0	24.8342	24.9720
185	42.0	318.0	1715.0	1477.0	52.0	17.0	3336.0	300.0	22.0	12.9097	12.8690
184	61.0	453.0	2364.0	2023.0	74.0	58.0	3374.0	367.0	20.0	23.0534	23.0179
183	32.0	303.0	1085.0	990.0	34.0	4.0	1323.0	161.0	16.0	8.2117	8.2375

Using these results, we can rank the modules. Once ranked, we find the ratio of faults that various percentiles (c) of the data contain. We compute the ratio of faults (G_c) and predicted faults (G_hat_c) for each model. We are then able to compute the performance of each model relative to the ground truth (phi). The following tables show the results:

M5 Fit

	G_c	G_hat_c	c	modules	phi
0	0.398126	0.340311	95	5	0.854782
1	0.599532	0.542208	90	10	0.904387
2	0.730679	0.666456	85	15	0.912104
3	0.817330	0.743406	80	20	0.909554
4	0.880562	0.810063	75	25	0.919938
5	0.920375	0.855915	70	30	0.929964
6	0.941452	0.893989	65	35	0.949586
7	0.964871	0.932000	60	40	0.965932
8	0.985948	0.962468	55	45	0.976185
9	1.000000	0.987903	50	50	0.987903

M5 Test

	G_c	G_hat_c	c	modules	phi
0	0.419087	0.220549	95	5	0.526259
1	0.630705	0.389471	90	10	0.617516
2	0.751037	0.474230	85	15	0.631433
3	0.821577	0.525389	80	20	0.639489
4	0.883817	0.576532	75	25	0.652321
5	0.925311	0.611252	70	30	0.660591
6	0.941909	0.636447	65	35	0.675699
7	0.962656	0.665282	60	40	0.691091
8	0.983402	0.690140	55	45	0.701788
9	1.000000	0.709055	50	50	0.709055

Greedy Fit

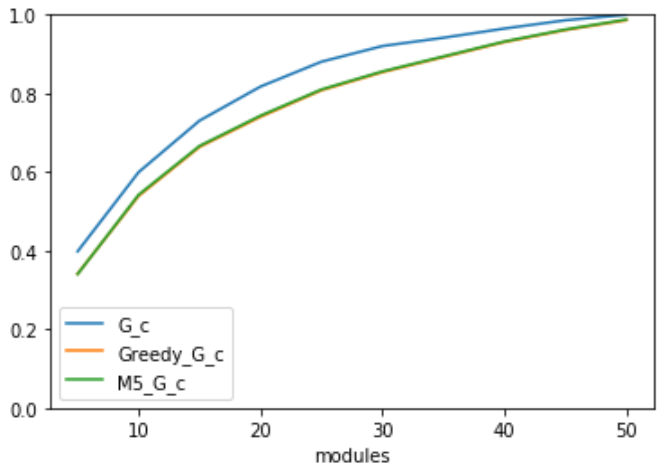
	G_c	G_hat_c	c	modules	phi
0	0.398126	0.341127	95	5	0.856831
1	0.599532	0.538964	90	10	0.898976
2	0.730679	0.663626	85	15	0.908232
3	0.817330	0.740272	80	20	0.905719
4	0.880562	0.807749	75	25	0.917310
5	0.920375	0.853836	70	30	0.927704
6	0.941452	0.892131	65	35	0.947612
7	0.964871	0.930339	60	40	0.964210
8	0.985948	0.960959	55	45	0.974654
9	1.000000	0.986048	50	50	0.986048

Greedy Test

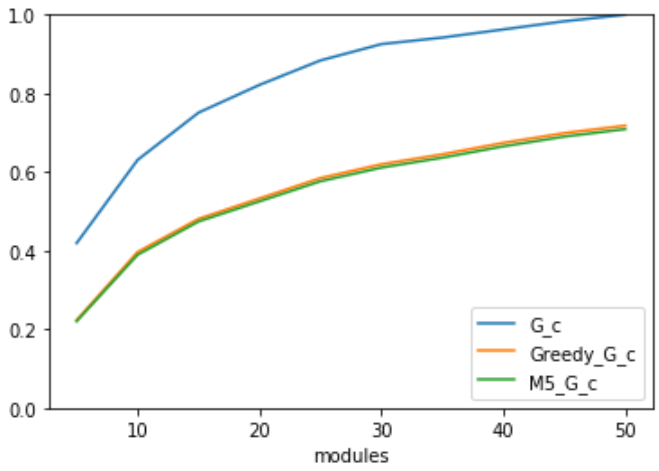
	G_c	G_hat_c	c	modules	phi
0	0.419087	0.223225	95	5	0.532646
1	0.630705	0.396507	90	10	0.628672
2	0.751037	0.480856	85	15	0.640255
3	0.821577	0.532736	80	20	0.648431
4	0.883817	0.584622	75	25	0.661473
5	0.925311	0.619742	70	30	0.669766
6	0.941909	0.645023	65	35	0.684804
7	0.962656	0.674121	60	40	0.700272
8	0.983402	0.698959	55	45	0.710756
9	1.000000	0.718016	50	50	0.718016

At first glance, it seems that both selection methods yield almost identical results across both fit and test data. We can visualize these results by using Alberg diagrams:

Fit

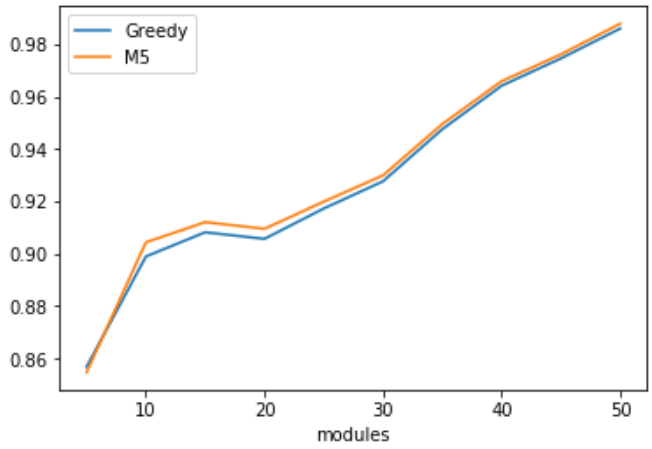


Test

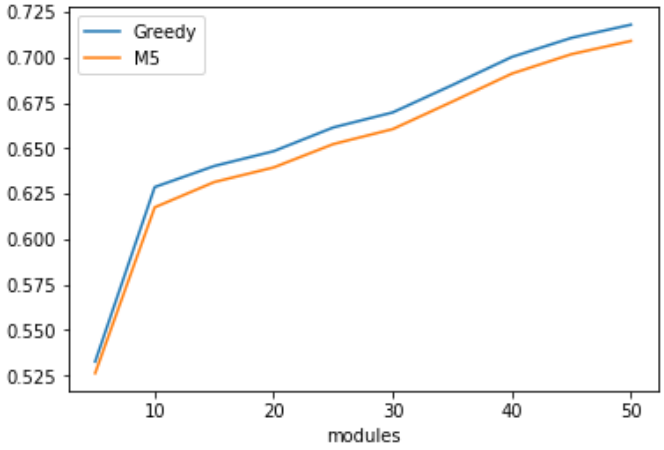


Here we see that both models have similar performance. With the test set, we see that the Greedy method had slightly better results. In order for a model to be considered robust, we want it to have a steady performance across various percentiles. Both models exhibit similar performance across module percentiles. Neither is significantly more robust than the other. We see this by plotting phi.

Fit



Test



Last we can look at the variance of phi to see that the greedy method has slightly less variance in performance:

M5

```
: m5_fit_results['phi'].describe()
```

```
: count    10.000000
  mean      0.931033
  std       0.039865
  min       0.854782
  25%       0.910191
  50%       0.924951
  75%       0.961846
  max       0.987903
  Name: phi, dtype: float64
```

Greedy

```
greedy_fit_results['phi'].describe()
```

```
count    10.000000
mean      0.928730
std       0.039645
min       0.856831
25%       0.906347
50%       0.922507
75%       0.960061
max       0.986048
Name: phi, dtype: float64
```

Conclusion

Two attribute selection methods were compared when used for Module Order Models. Both methods yielded similar results. However, we will choose the Greedy method as the more robust. The Greedy method showed a lower variance in phi, the performance across percentiles of ranked modules. This indicated that the model has more reliable results on various subsets of the dataset. Thanks for stopping by.