DATA 1030 Final Report
Predicting Galaxy Distances with Photometric Redshift via Regression

Adrian Lam
Data Science Initiative, Brown University, Providence, RI 02912
Github repository: https://github.com/adrian-lam/data1030_project

I. INTRODUCTION

Of all the natural sciences, astronomy is a little different from the rest, in that it is not really an experimental science, but rather an observational science. Unlike physics, chemistry, and biology, you can't physically manipulate any of the things you study in astronomy. All astronomers can do is observe these objects from earth and record measurements in the form of data. Only from this data can astronomers glean information about astronomical phenomena. So really, in addition to being an observational science, astronomy is also a data science.

In this project, I will be working with astronomical data of galaxies in the galaxy cluster Abell 2457, with the goal of training a regression model to predict the distances of these galaxies from Earth.

In astronomy circles, distances in this context is referred to as *redshift*. The farther an object is from us, the more redshifted it is. Now, the redshift of a small subset of these galaxies are determined through an independent (and more expensive) method, known as spectroscopy. This set of redshifts will be our training label, and is denoted "zspec" in the DataFrame in figure 1. Because of how time-intensive it is to obtain spectroscopic redshifts, the number of such measurements are typically quite low. For our particular cluster, 638 spectroscopic redshifts was obtained (after some preliminary filtering). By extension, that limits the number of data points we have to 638.

The features of our model are the brightness of each galaxy at various wavelengths. In astronomy circles, the measure of brightness is given by the dimensionless quantity *magnitude*, and our features consist of the magnitudes of each galaxy through five different colored filters u, g, r, i, and z. Accompanying each magnitude measurement is a measurement uncertainty. These correspond to the columns with a "d" prepended to the filter name. It turns out that the uncertainty could also be a useful feature in our model, as it is not uncommon in machine learning for some features to not have direct influence on the target variable, but act as proxies that reflect the influence of other variables that do. This gives us a total of 10 features, as shown in the DataFrame in figure 1.

Given our features and target label, we can train a suitable regression model, and use that to estimate the distances to the remaining majority of the galaxies in the cluster, without the need to use the more expensive method over the entire population of galaxies.

	r	dr	g	dg	u	du	i	di	z	dz	zspec
0	20.8498	0.087124	20.8860	0.005956	19.9325	0.014263	20.0788	0.035860	19.8511	0.075540	2.505000
1	20.2984	0.046205	20.6447	0.004217	19.2952	0.008835	20.0889	0.034942	20.4333	0.078268	1.895000
2	20.8883	0.064217	21.7938	0.012222	22.1299	0.092288	20.5905	0.044032	20.0958	0.099445	3.405000
3	15.3928	0.001871	16.4288	0.000498	16.1900	0.002139	15.0507	0.001141	14.7237	0.002326	0.039055
4	21.1361	0.085443	21.3559	0.007474	20.6440	0.029433	20.8778	0.056025	21.2004	0.124193	2.365000
...	...	...	...	...	...	...	...	...	...	...	...
662	19.1186	0.027805	21.0347	0.013256	21.4664	0.091210	18.4442	0.011597	18.1855	0.023093	0.410812
663	19.6923	0.042599	21.9020	0.022806	22.8041	0.140533	18.8218	0.017058	18.7771	0.039564	0.499613
664	20.4720	0.058682	20.5128	0.003799	19.5003	0.009951	20.3356	0.046609	20.1929	0.103614	2.574073
665	20.4289	0.074319	22.5736	0.033274	22.4091	0.127780	19.6188	0.029984	19.3738	0.063926	0.689085
666	17.4448	0.007564	18.8868	0.002270	19.0618	0.013087	16.9325	0.003545	16.7861	0.008904	0.191646

FIG. 1: Pandas DataFrame of our galaxy data.

II. EXPLORATORY DATA ANALYSIS

Figure 1 shows the dataset we are working with. It is IID, and all the features, along with the target variable, are continuous.

Figure 2 shows stacked histograms of the magnitude of galaxies, through various filters. The key takeaway is that the magnitudes are approximately normally distributed, and that different filters have different means.

Figure 3 shows a semi-log histogram of the redshift of galaxies. The vast majority of galaxies have redshift below 1, with a small amount of galaxies having redshifts from 1 to 5. The semi-log plot is to make clear the existence of higher redshift galaxies in our dataset, because their presence was almost indiscernible with a linear y-axis.

Finally, let's take a look at the relationship between features and the target variable. Fig. 4 shows a scatter plot of the redshift of galaxies vs. their magnitude in the r-filter. We clearly see that there is a correlation between the two variables. Fig. 5 shows a scatter plot of redshift vs. the uncertainty in magnitude in the r-filter.

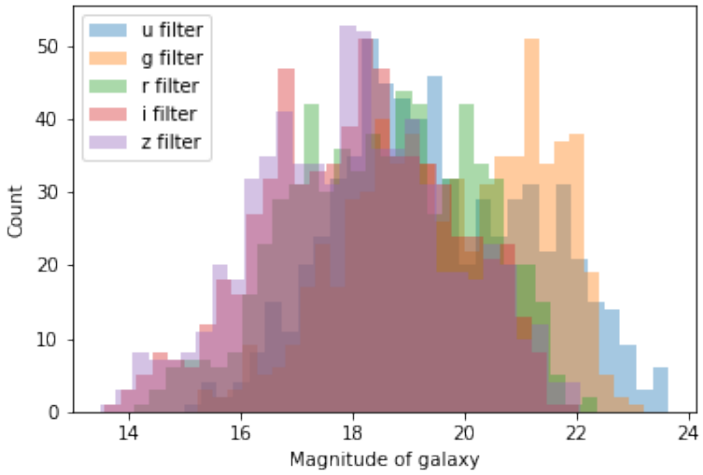

Figure 2. Historgrams of the magnitude of galaxies measured through various filters. The magnitudes through each filter is approximately normally distributed, with the mean varying between filters.

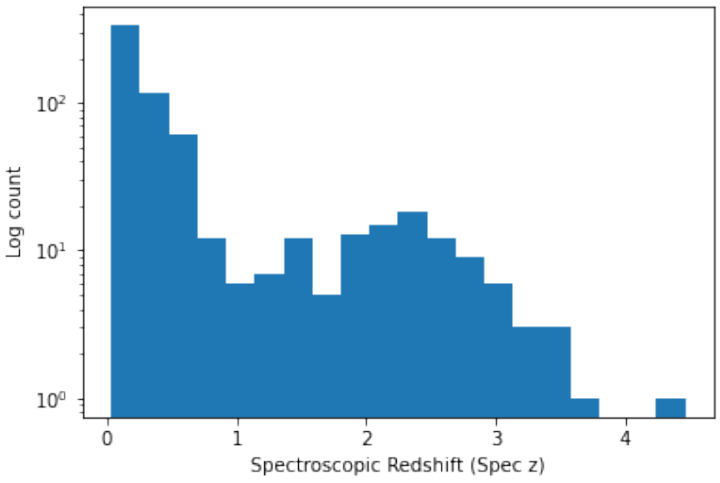

Figure 3. Semi-log histogram of the redshift of galaxies, measured by spectroscopy. The vast majority of galaxies are low redshift (z<1), with a small number of galaxies having higher redshifts, up to almost z = 5.

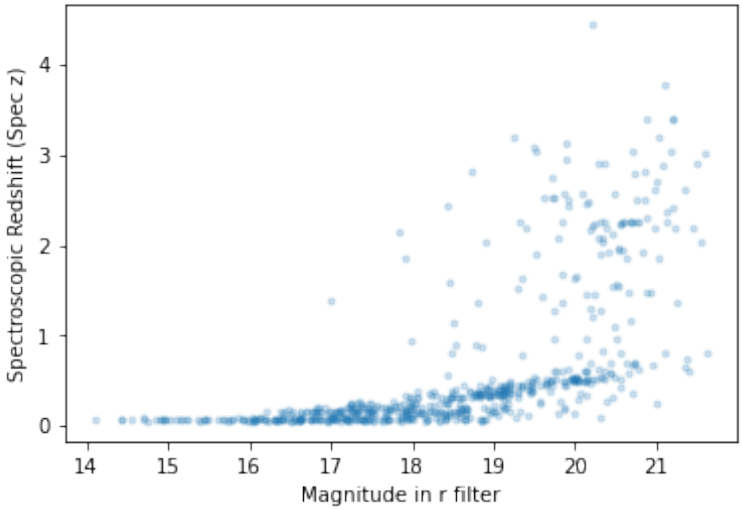

Figure 4. Scatter plot of the redshift of galaxies vs its magnitude in the r-filter. There is a clear correlation, but it isn't simply a linear relationship.

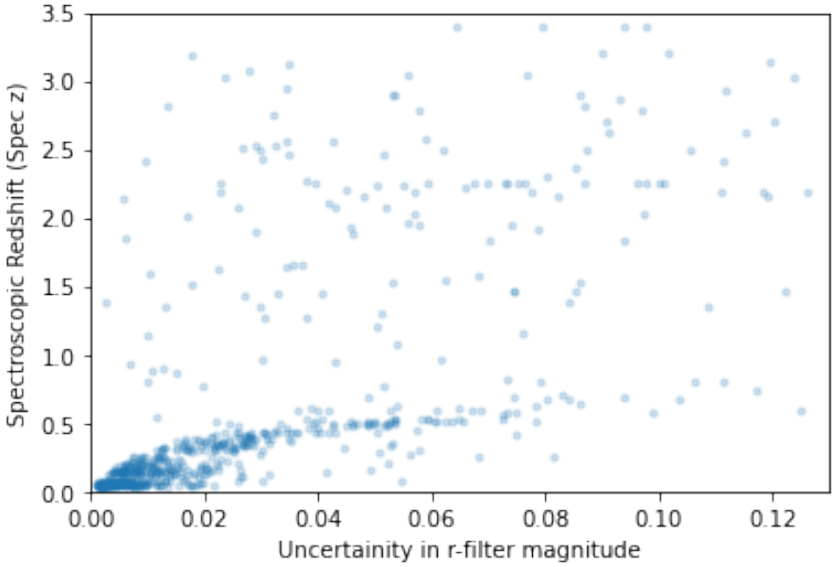

Figure 5. Scatter plot of the redshift of galaxies vs the uncertainty of its measured magnitude in the r-filter

III. METHODS

We start off with our splitting strategy, which warrants careful considerations considering the small size of our dataset and the imbalanced nature of our target variable.

With 638 data points, our dataset is on the small side, so a traditional 60/20/20 split was made. Because the values of our target variable is imbalanced as we saw in figure 3, we make sure to use stratification when applying all our splits. We will stratify the target variable into three bins: [0,1], [1, 2.5], and [2.5, 5].

To reduce our models' sensitivity to specific random splits, we will apply 4-fold cross-validation to our training and validation sets. That way, all observations have a chance to be used for both training and validation, and each observation will be used for validation exactly once. We are thus using stratified four-fold split.

Since all 10 of our features are continuous, we simply apply Standard Scaler to all of them (MinMaxEncoder doesn't work here because none of our features have strict upper-bounds). Our target variable, being also continuous, does

not require any preprocessing. After this simple preprocessing procedure, we are still left with 10 features.

The evaluation metric was chosen to be the mean squared error (MSE), which is a safe choice for regression and easy to interpret. The closer each predicted value is to the true value, the smaller the MSE. Thus, the best validation score will be the one where the MSE is *minimized*. A natural baseline model would be one that simply predicts the mean of the target values in the training set for all points.

Putting together all the aforementioned pieces, we can construct our machine learning pipeline. For each of the ten random states, we loop through all the hyperparameter combinations, and find the corresponding mean val_score of the 4 folds. The hyperparameter combination with the best (ie. lowest) mean val_score in that random state will be stored, and used to make predictions on the test set, to then obtain a test score. We thus end up with 10 best models and 10 corresponding test scores. Averaging these 10 tests scores would give a fairly robust metric of how predictive our model is.

Using this pipeline, three different non-linear machine learning algorithms were trained: support vector regressors, random forest regressors, and XGBoost regressors. Relevant hyperparameters were tuned for each algorithm, and the range of hyperparameters were always chosen to be wide enough so that we see both the high-bias and high-variance regimes. This ensures the we will find the optimal model somewhere in between.

For the support vector regressor, we tune gamma and C, trying values [1e-5, 1e-3,1e-2, 1e0,1e2, 1e3, 1e5], and [1,10,100,1000] respectively.

For the random forest regressor, we tune the max_depth and max_features, trying values [1, 3, 5, 10, 30, 50] and [0.5, 0.6, 0.7, 0.8, 0.9, 1.0] respectively.

For the XGBoost regressor, we tune the max_depth as well as the alpha and lambda parameters. We will try max_depth of [1, 3, 5, 10, 30, 50], as well as alpha and lambda values of [0, 1e-2, 1e-1, 1e0, 1e1, 1e2].

In the following section we will explore the results we get from each of these machine learning algorithms.

IV. RESULTS

A. Support vector regression

We know SVMs underfit at low gamma and low C, and overfit at high gamma and C, with the optimal values somewhere in between. For our dataset, the bias-variance tradeoff is much more sensitive to gamma. Table I below summarizes the best results we get for each random state.

Random state	**Best gamma**	**Best C**	**Best validation score**	**Test score**
1	0.01	100	0.181	0.175
2	0.01	1000	0.167	0.229
3	0.01	1000	0.170	0.202
4	0.01	1000	0.170	0.202
5	0.01	1000	0.180	0.138
6	0.01	1000	0.158	0.293
7	0.01	1000	0.180	0.118
8	0.01	1000	0.163	0.224
9	0.01	1000	0.163	0.205
10	0.01	1000	0.163	0.160

TABLE I: Summary of the best hyperparameter combination for each random state from training an SVR, based on the 4-fold cross-validation score. The evaluation metric is the MSE.

It is worth pointing out that our test scores fluctuate quite a bit between some random states. This is in large part due to the small dataset we have. With only slightly over a hundred data points in the test set, a couple of predicted outliers can really throw off the MSE.

The mean test score comes out to be 0.195, with a standard deviation of 0.0476. The baseline model gives a mean MSE of 0.663, so our SVR does better than the baseline by about (0.663-0.195)/0.0476 = 9.8 standard deviations.

Figure 6 shows a scatter plot of the true target labels versus the predicted labels of the test set, using our best SVR model. We see that our model does reasonably well, surprisingly even for high redshifts. It's mainly a couple of low redshift points that are poorly predicted as mid-ranged that is throwing the score off.

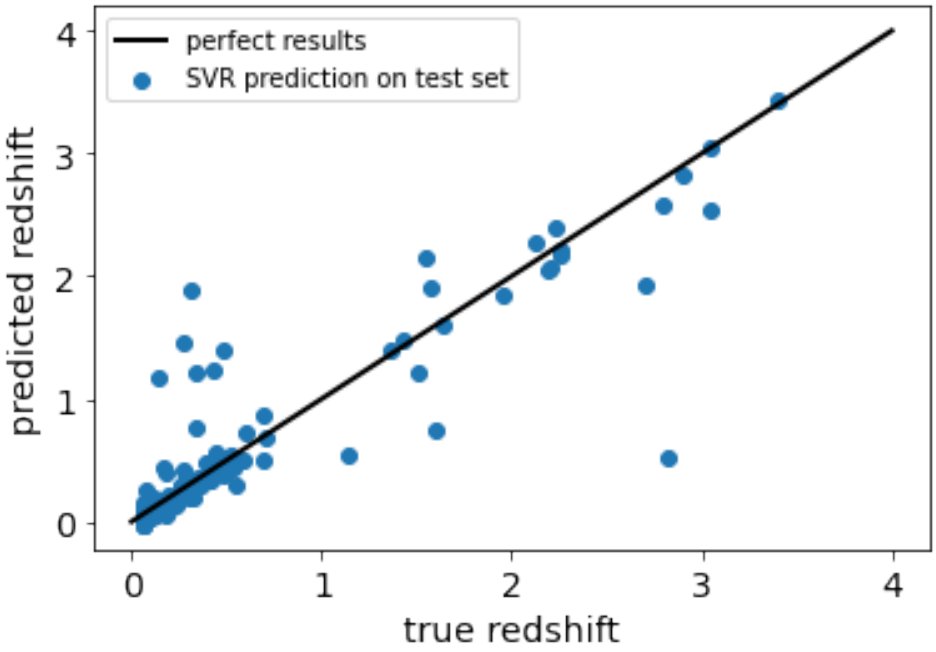

Figure 6. Scatter plot of the true redshift versus the predicted redshift from the test set, using one of our best support vector regression models. Also plotted for comparison is the line true redshift = predicted redshift, corresponding to perfect results.

B. Random forest

We know random forests with small tree depth will be high bias, while a random forest with high tree depth and high percentage of features to be high variance. We again expect the optimal model to contain parameters somewhere in between these two extremes. This is indeed the case, but there is a bit of variation here depending on the random state used.

Random state	Best max features	Best max depth	Best validation score	Test score
1	0.5	5	0.177	0.185
2	0.6	10	0.166	0.247
3	0.5	5	0.197	0.153
4	0.7	5	0.202	0.204
5	0.5	30	0.183	0.201
6	0.8	10	0.163	0.263
7	0.5	30	0.187	0.141
8	0.7	10	0.164	0.240
9	0.5	10	0.175	0.202
10	0.6	30	0.174	0.203

TABLE II: Summary of the best hyperparameter combination for each random state from training a random forest, based on the 4-fold cross-validation MSE score.

The mean test score comes out to be 0.204, with a standard deviation of 0.0368. The baseline model gives a mean MSE of 0.663, so our random forest does better than the baseline by about (0.663-0.204)/0.0368 = 12 standard deviations.

From figure 7, we again see that most of the points are predicted reasonably well, and again it is a few low redshift points that are incorrectly predicted as mid-ranged that is causing the most problems.

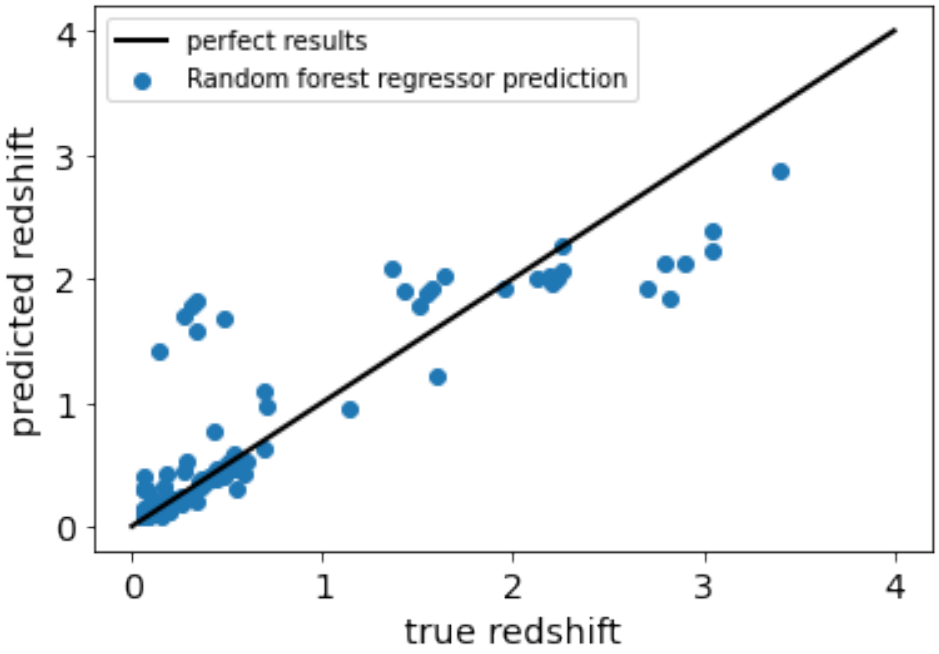

Figure 7. Scatter plot of the true redshift versus the predicted redshift from the test set, using one of our best random forest regression models. Also plotted for comparison is the line true redshift = predicted redshift, corresponding to perfect results.

C. XGBoost regressor

XGBoost is also a tree-based method, but while a random forest is a bagging method, XGBoost is a gradient boosting method. XGBoost in particular offers regularization, so there are additional alpha and lambda parameters to tune.

Random state	Best max depth	Best alpha	Best lambda	Best validation score	Test score
1	10	1	1	0.181	0.218
2	3	0.1	0.01	0.174	0.266
3	3	1	0.01	0.187	0.150
4	3	0.1	10	0.199	0.230
5	3	1	0	0.188	0.220
6	3	1	0.01	0.171	0.237
7	3	1	0.01	0.179	0.179
8	3	0.01	1	0.163	0.219
9	3	0	1	0.178	0.230
10	30	0	0	0.175	0.198

TABLE III: Summary of the best hyperparameter combination for each random state from training an XGBoost regressor, based on the 4-fold cross-validation MSE score.

The mean test score comes out to be 0.218, with a standard deviation of 0.0303. Given the baseline model mean MSE of 0.663, our random forest does better than the baseline by about (0.663-0.204)/0.0368 = 12 standard deviations. While XGBoost has a slightly lower overall test score compared to the SVR and random forest, it has the lowest spread, making it score higher than the baseline model by more standard deviations.

From figure 8, we see again that most of the points are predicted reasonably well, but with a few low redshift points that are incorrectly predicted as mid-ranged. This seems to be a common artifact amongst all three algorithms, so it's likely it has something to do with some underlying bias. We will address this more in the outlook section.

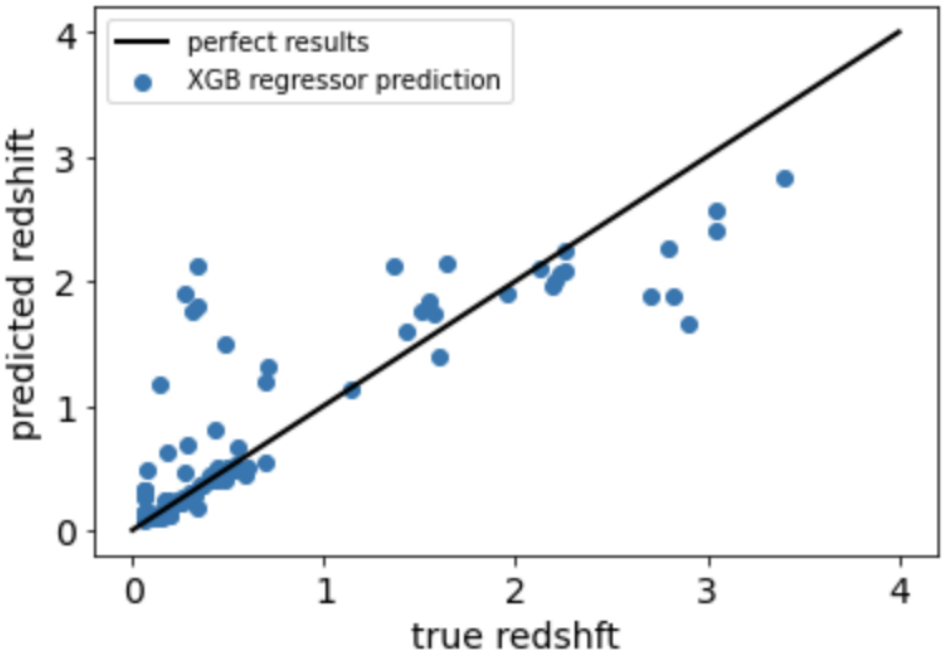

Figure 8. Scatter plot of the true redshift versus the predicted redshift from the test set, using one of our best XGB regression models. Also plotted for comparison is the line true redshift = predicted redshift, corresponding to perfect results.

D. Feature importance

To get a sense of feature importance, we can perform a permutation feature importance calculation. This will show which feature influences the final prediction the most. Figure 9 shows this on one of our SVR models.

This is a surprising result. Intuitively, we expect the features 'u', 'g', 'r', 'i', 'z' to be the most important, as they give information that is directly related to redshift, and we would expect their uncertainties to play a smaller role. This is partially seen in figure 9, as the 'g', 'r', and 'u' filter influences the test score a lot. However, what's very surprising is the large influence of 'dg'. There is certainly no a priori reason that the uncertainties of a particular feature would have such a huge impact on the test score. Perhaps there is something very wrong with the data or measurements made for the 'g' filter and 'dg'. This will be something worth looking into in further work.

V. OUTLOOK

From the surprising and strange result in permutation feature importance, the first step to consider in future work is to investigate the data of features 'g' 'dg'– there must be a reason that is causing the erratically huge influence 'dg' has on our results.

Second, we note that a common theme amongst all three algorithms is that there seem to always be the same few low redshift points incorrectly predicted as mid-range redshifts (refer to figures 6-8). It could be due to some inherent bias in our model– we could mitigate that by possibly considering more features, such as considering what type of galaxy each observation is.

We also note the small number of data led to some randomness in our results. Unfortunately obtaining more data is not viable– measurements for the true y labels are hard to obtain, which is precisely why we're trying to train a model to estimate it in the first place.

Finally, in future work, two additional algorithms I would train would be a polynomial regression and a KNN regression.

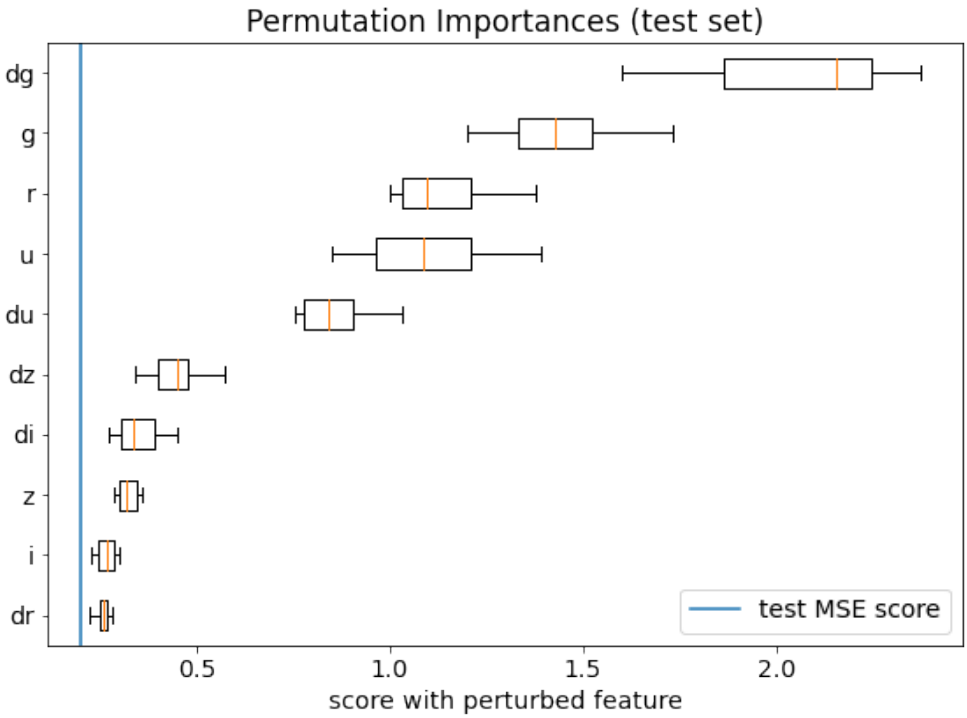

Figure 9. Permutation feature importance performed on one of our SVR models.

VI. REFERENCES

[1] N. Benitez, *The Astrophysical Journal* **536** no. 2, (2000) 571-583, arXiv:astro-ph/9811189.
[2] J. VanderPlas, *Python Data Science Handbook*. 2017.
[3] Data obtained from https://ned.ipac.caltech.edu/ and the Brown University Physics Department.