# Report for Exercise 1

Parsia Basimfar

## Methods

The first task of this question was to determine a baseline performance for the regression analysis. To do so, linear regression (without regularization) is performed on each of the datasets and their $R^2$ scores are reported. Different regularization schemes (l1, l2 and elastic net) are also executed on each of the datasets (using 'GridSearchCV' and 6-fold cross-validation) to see how much they improve the fit to the data and also to choose one of these methods for determining the goodness-of-fit of a model.

For the 2nd task 20 datasets are created using the two different missing value strategies; miss at random and miss by feature. Further the missing datasets are imputed using the two imputation algorithms mentioned in the question. Both imputation methods are implemented in the package 'fancyimpute' and therefore I use the built-in methods of that package to accomplish this task. To determine the shrinkage parameter, I use the method mentioned in the question. 5 values [0.01, 0.1, 0.25, 0.5, 0.75] are chosen which are multiplied by the maximum singular value of the scaled (Biscaler() method) datasets. As this part is computationally expensive I choose 5 random simulations for each dataset using each missing value strategy (a total of 8 dataset categories) and take the average of their $R^2$ scores obtained from testing them with test sets using lasso. The lambda proportion with the highest average value across the 5 chosen simulations is chosen to perform soft imputation on the whole 20 created datasets. The regularized regressions for the imputed data are all implemented by the LassoCV method to ensure the best fit. The residual plots in this report were obtained using the package yellowbrick.

In order to do feature selection I used elastic net and after a bit of tweaking I chose an l1-ratio 0.9 for all datasets (imputed and actual) to get both reliable (meaning enough number of features) and comparable results. I choose and prefer elastic-net to lasso here because we are dealing with some low-rank datasets and for their feature selection elastic net can be less aggressive than lasso and provide us with more features (some of the covariates included) which could be informative. I use cross-validation to get the best alpha which is the constant multiplied by the penalty terms in 'sklearn'. Moreover, I bootstrap the simulated samples 200 times for the small datasets and 100 time for the large datasets (in the case of baseline I bootsrap every dataset 200 times). After a bit of trial and error I choose those variables as important that were chosen on 90% bootstrap samples. This is done for the actual data and the imputed data. For the imputed data I run this on the whole set of 20 simulations and after getting the results I count the features being selected on each simulated dataset. The features with a higher number of counts (20 being the maximum) are chosen as the selected features.

## Results

Linear regression without regularization does not harbor good results on the small data sets, whereas regularized regression improves the regression results on the small ones significantly. This improvement is observed with respect to the big datasets as well but is less significant. The figure below summarizes the regression findings on the given datasets.

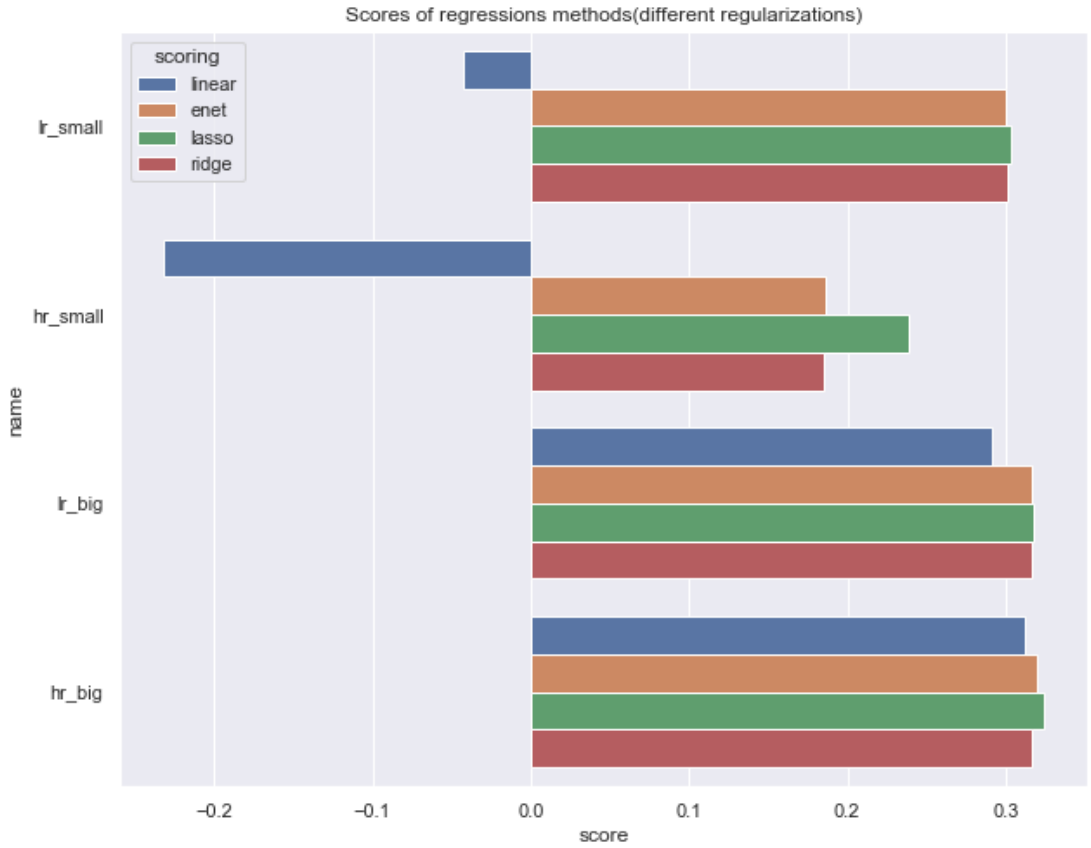


*Figure 1. The y axis denotes the different datasets and x axis are the $R^2$ scores. 4 different regularization schemes were worked out as designated by the legend. The results clearly show that regularization improves the regression performance, especially for the smaller datasets. Of all the regularization schemes, lasso seems to have the greatest impact.*

Figure 1 suggests that the choice of regularization should be lasso although there does not exist a huge difference and the only noticeable difference is within the hr_small group. Consequently, I choose lasso for other tasks that require regularization in this exercise.

The results of regression on the imputed data are further discussed. The results for the linear regression using the mean imputation method are shown in figure 2.

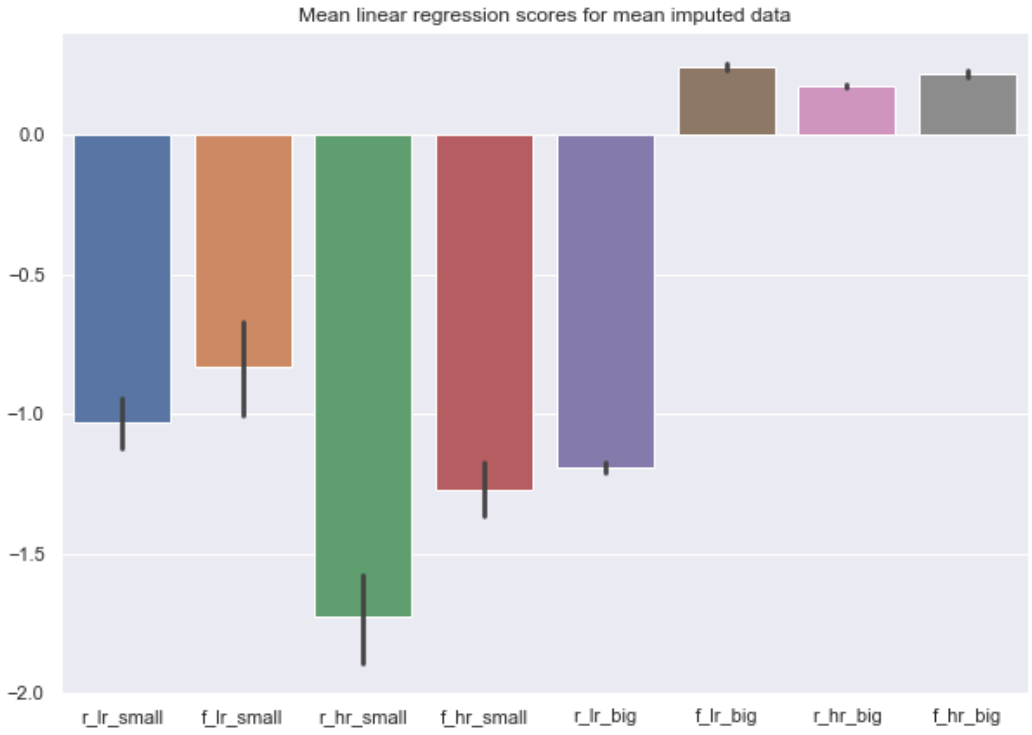


*Figure 2. The graphs show the mean $R^2$ score for the different datasets. The lines on the bars show the standard deviation of the scores. Those dataset names preceded by an r_ denote the missing at random strategy and those preceded by f_ denote the missing by feature strategy.*

The results of linear regression on mean imputed data is not surprising. They are almost everywhere worse, but some datasets even more so such as the small datasets. The larger datasets, except for the randomly missing data do not show a vast difference. Next step is regularization and the results of lasso for mean imputed data are shown in Figure 3 and 4 in the next pages.

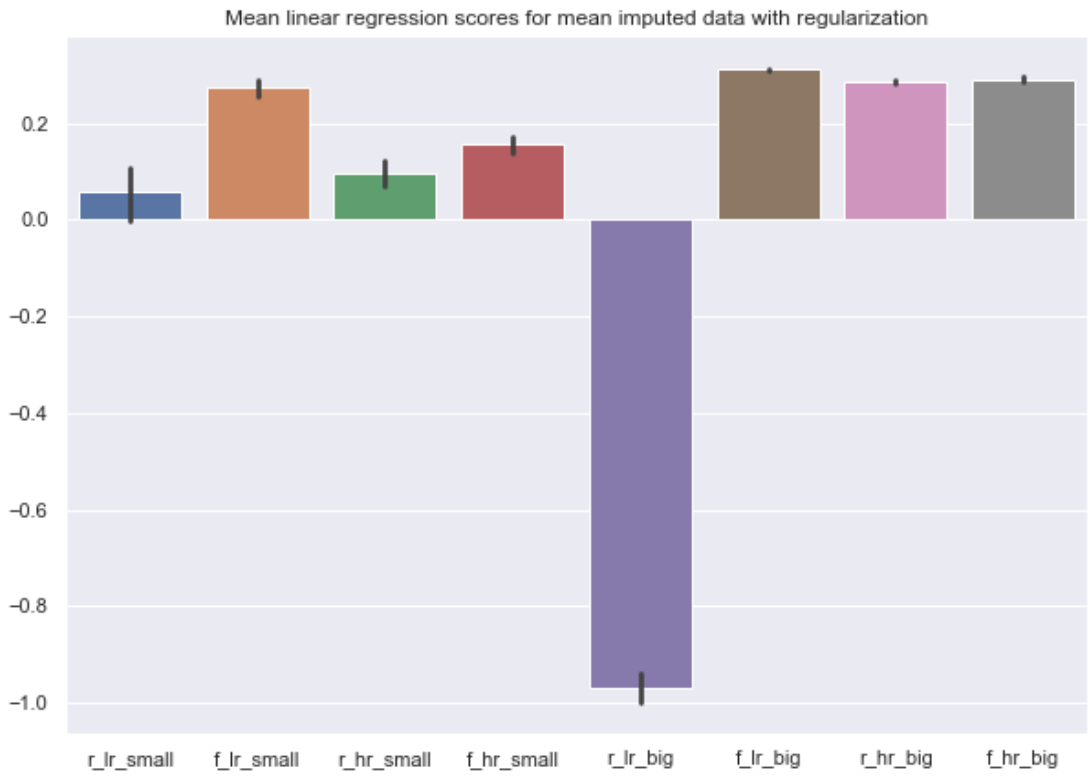


*Figure 3. This figure is identical to figure 2 except that here the scores correspond to lasso.*

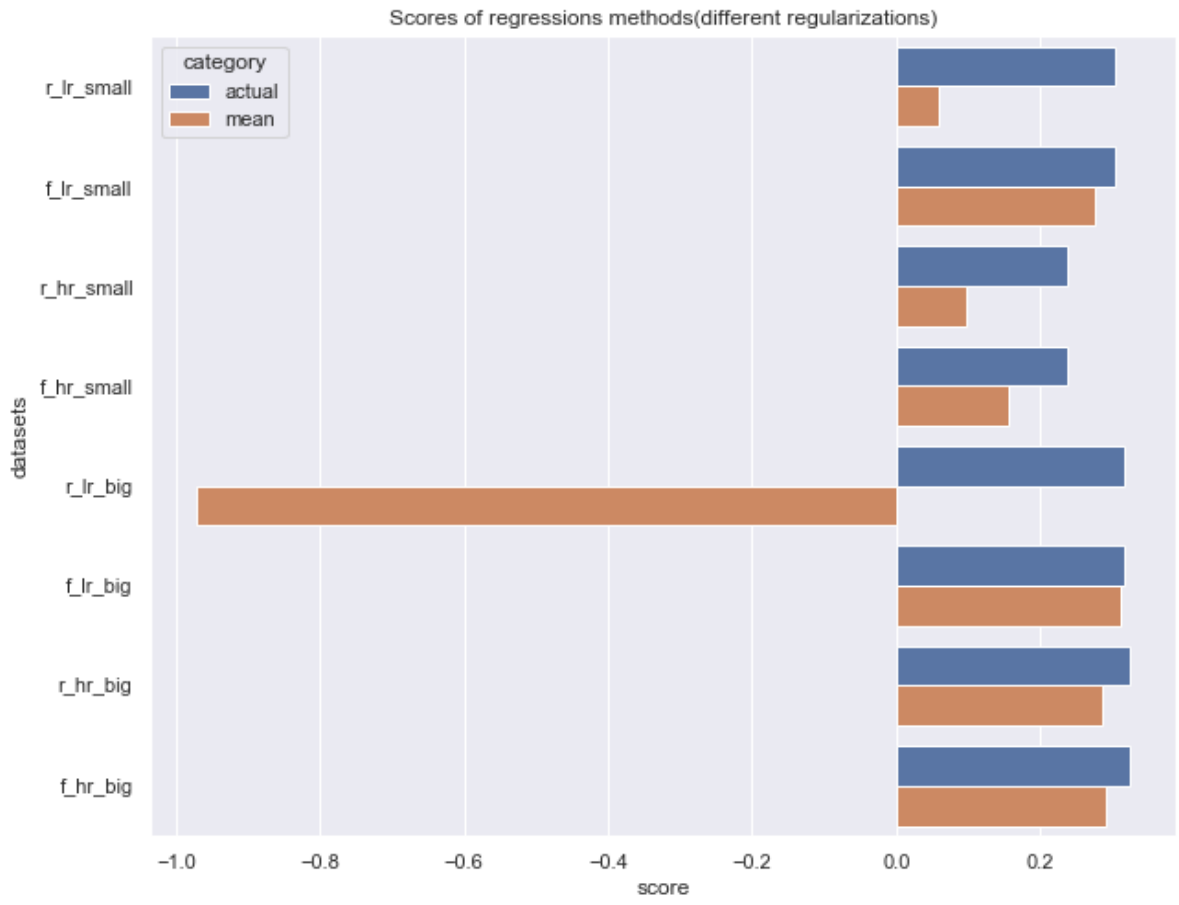


*Figure 4. The comparison between the two methods show that mean impute is worse especially in the smaller dataset cases but has relatively good performance for the larger datasets, except for the lr_big missing at random case.*

Next we analyze the performance of the soft impute algorithm using the same criteria as before; first using linear regression and then using regularized regression. The soft impute was executed using the best shrinkage parameter in accordance with the procedure described in the methods section. Figure 5 below shows a comparison of shrinkage parameters on different datasets. After choosing the best shrinkage parameter, I test its performance on the different datasets in the two aforementioned missing settings. The results are summarized in figures 6-7 below. You might notice that there is a difference in figures 4 and 6 with respect to the mean imputed random lr_big dataset. This is due to different hypermarameters being used in figure 6 for that dataset and I'll elaborate on it in the discussion section.


*Figure 6. This plot suggests that the shrinkage parameter might have either small or huge effects on the performance of the soft impute. For all datasets only the best shrinkage parameter was used for regression analysis.*

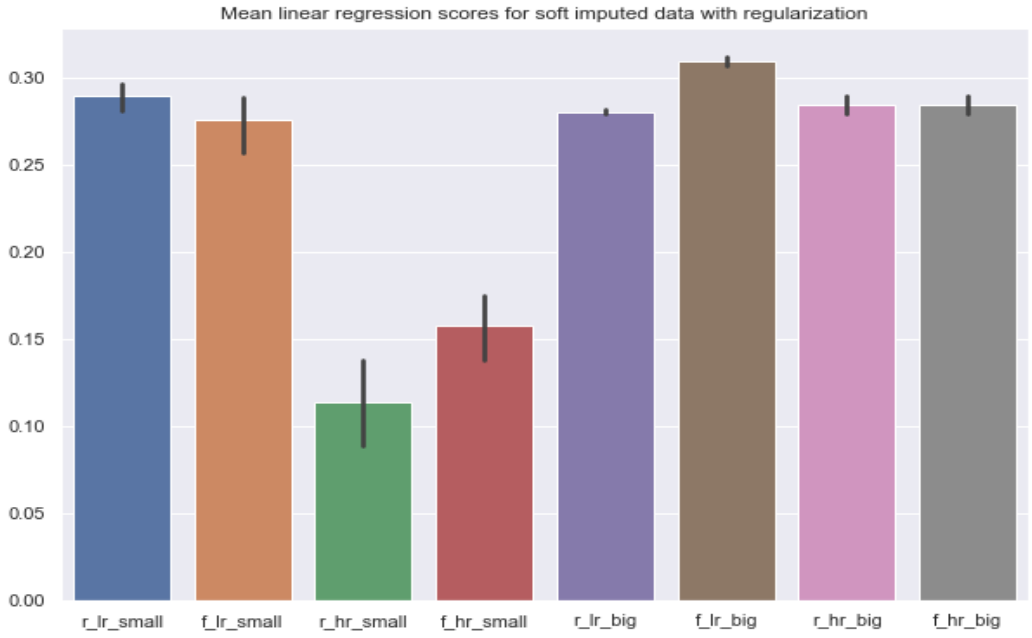


*Figure 5. $R^2$ scores as achieved by performing the lasso on each of these datasets*

I have saved the results of the feature selection for the discussion part as I think it is more relevant there.

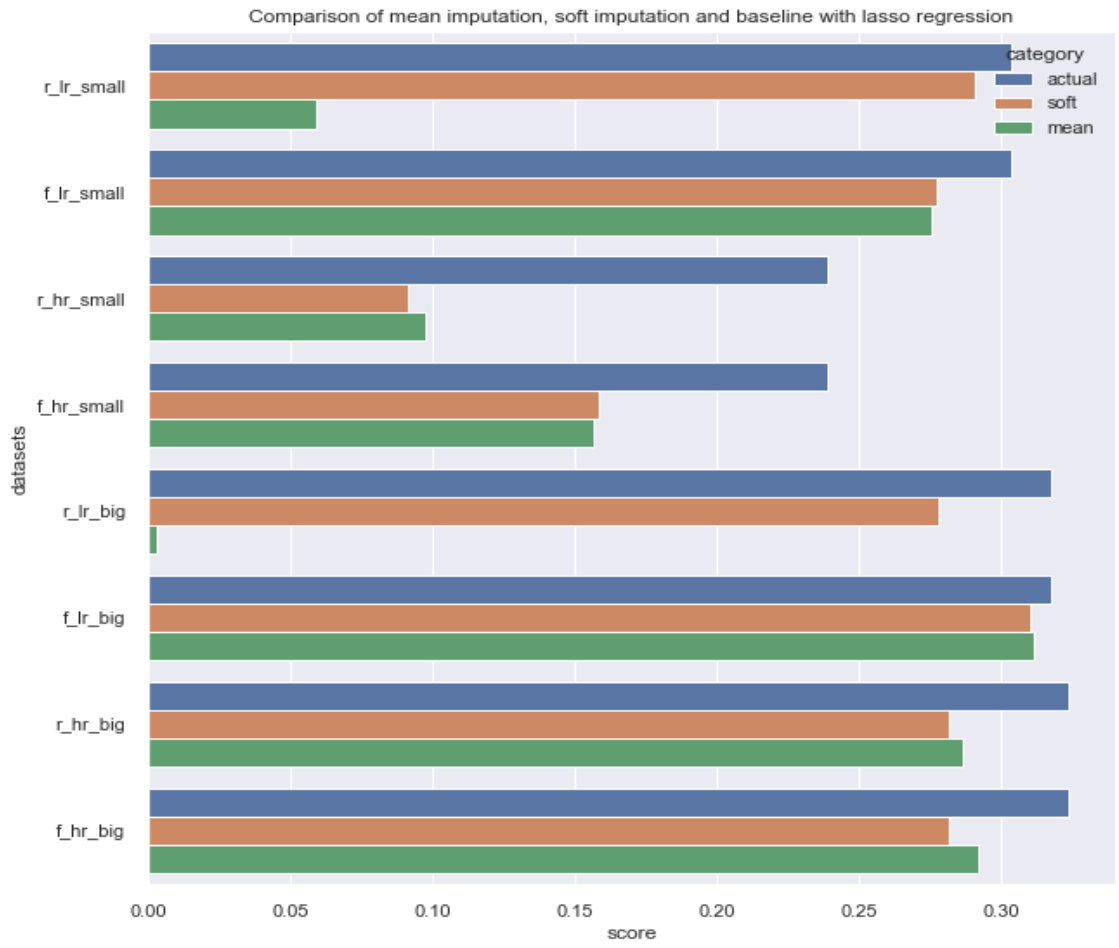


*Figure 7. The results attest to the superiority of soft impute in most cases although in the large datasets, except for the random lr_big, it seems that there is no difference between the two imputation algorithms. Mean impute even seems slightly better.*

# Discussion

## Regression Results with and without imputation

As expected, regularization improves the regression fit no matter which regularization method is used. This is especially evident in the smaller datasets. The smaller datasets do not provide as much information for a good fit to be established. In the small datasets, the low-rank one has a better fit than the high-rank one. Since the data is collinear in the low-rank case, less information is needed to find a fit for the data and this is why the low-rank dataset has a better $R^2$ score. In the larger datasets, the effect of existence or non-existence of collinearity is less pronounced as there enough observations for the regression to find a good fit.

Along the same lines, we can see why regularization improves the regression fit, especially in the smaller datasets. By removing unimportant features, regularization can overcome noise in the data and prevent overfitting. The latter procedure is done when using elastic net or lasso but not ridge regression and it might explain why the first two perform better than the other one. Ridge does a proportional shrinkage while lasso multiplies each coefficient by a constant and truncates at zero.

Another point to note here is that, scaling does not help with improving the performance. As seen from the data (figure 8), features have a normal or near-normal distribution around different means with different standard deviations. Scaling the data seems to have no effect on the $R^2$ score but this is not unexpected. In order to further show that standardization is not conducive towards improving the model I have plotted the residuals for the lr_small dataset in figure 9. Also since we want to compare between low rank and high rank data it is crucial that we conserve the correlations that exist between the variables. Standardization might affect this. Standardization is best when we have polynomial terms being fit to the data, since we are doing linear regression, standardization seems rather unnecessary. But this might explain why ridge perfroms worse than lasso since ridge involves $2^{nd}$ order terms. Lastly, since we don't have domain knowledge we can't be really sure whether scaling would result in better interpretability. All in all I choose not to standardize the dataset as it seems to not be suitable here, or make any difference at best.


Distribution of some randomly selected features in the lr_small dataset
117
108
30
78
45
177
182
170
12
132
12
10
8
6
4
2
0
−15
−10
−5
0
5
10
15

*Figure 7*

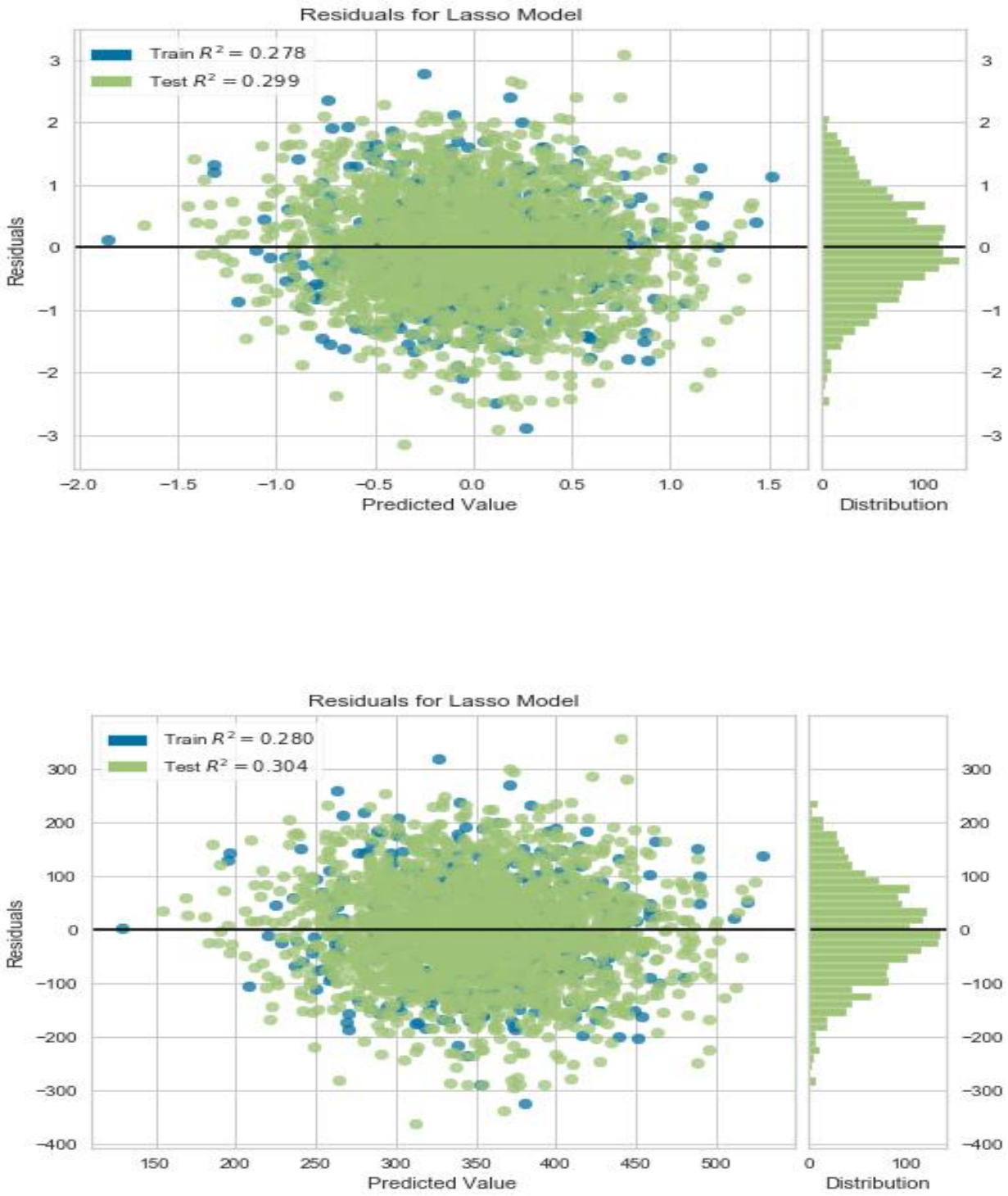


*Figure 8 A comparison between the residual distributions shows that they have the same mean and standard deviation, before standardization (top figure) and after it (bottom figure). The models were fitted using the best hyperparameters.*

## Mean Imputation

The first method of imputation executed in this exercise is mean imputation, so the missing entries are replaced with the mean of the entries given a certain feature. We will look at mean imputation's performance with respect to small and large n, low and high rank and the different missing strategies. This will be done by mostly referring to figures 3 and 4. We see that the regression results on the imputed data have a higher standard deviation (variability) for the small datasets compared to the large ones. This might be due to an averaging effect (70% 5000 is more than 70%  500) and better average values are attained in large datasets to be replacing missing values. Also more observations usually translates to more robust and consistent predictions. There is more variability in performance of low-rank data sets vs their high-rank counterparts. Missing values have higher impact on low-rank dataset performances here because if one of the important features is compromised, there are not many options to compensate for it, where as in high-rank ones there are plenty other important features. Also random strategy has more variability than feature strategy because in random strategy every feature is missing by some extent although by varying degrees while in feature strategy some features are not changed and those that are changed have less missing values.

Another thing to note is that low rank data have worse results compared to their high rank counterparts in the random setting. Mean imputation tends to reduce the variance in the data, thus imposing a bias. $R^2$ is the explained variance of the dependent variables based on the independent variables. Since we are reducing the variance in the training set, the regression model is fit according to that bias. When validated against the test set its performance decreases because there is an amount of variance in the test data that was missing in the imputed data. For the low rank data this effect is more stressed, as the number of independent variables in these datasets are lower. So if they are distorted by chance then they will have a larger impact on how well a regression model explains the test data. Moreover, since mean imputation is looking at the mean values of each feature separately, it is blind towards the multivariate relationships that exist between the features. In other words, it underestimates the correlations that exist between the independent and dependent variables. This is another reason why low rank data have worse performance.

The mechanism described above can also explain why there is a difference between the performances of the two missing strategies. Let us neglect the fact that in the missing by random strategy there are more missing values compared to the missing by feature strategy (70% to 40%). In the missing by feature strategy only some of the features are missing while others remain intact. In the random strategy each feature is likely to miss 70% of its values. This causes the missing by feature strategy to have less bias and keep some of its dependent variables intact. So in general, the random strategy has a poorer performance compared to the feature strategy.

Interestingly in the feature strategy, the low rank datasets have better performance than their high rank counterparts. One reason for this may be the regularized model (lasso) that we fit to

the datasets. Further testing with ridge regression yields similar results, so the issue lies somewhere else. By looking at the residual plots of missing by feature low rank high rank datasets, we see that the residuals are much bigger on training for the high rank datasets, whereas the residuals become small on the test sets. This discrepancy in the residuals suggest that the imputed datasets and the test data do not have close enough values but have similar distributions which results in some performance deterioration but not enormously (Figure 9).

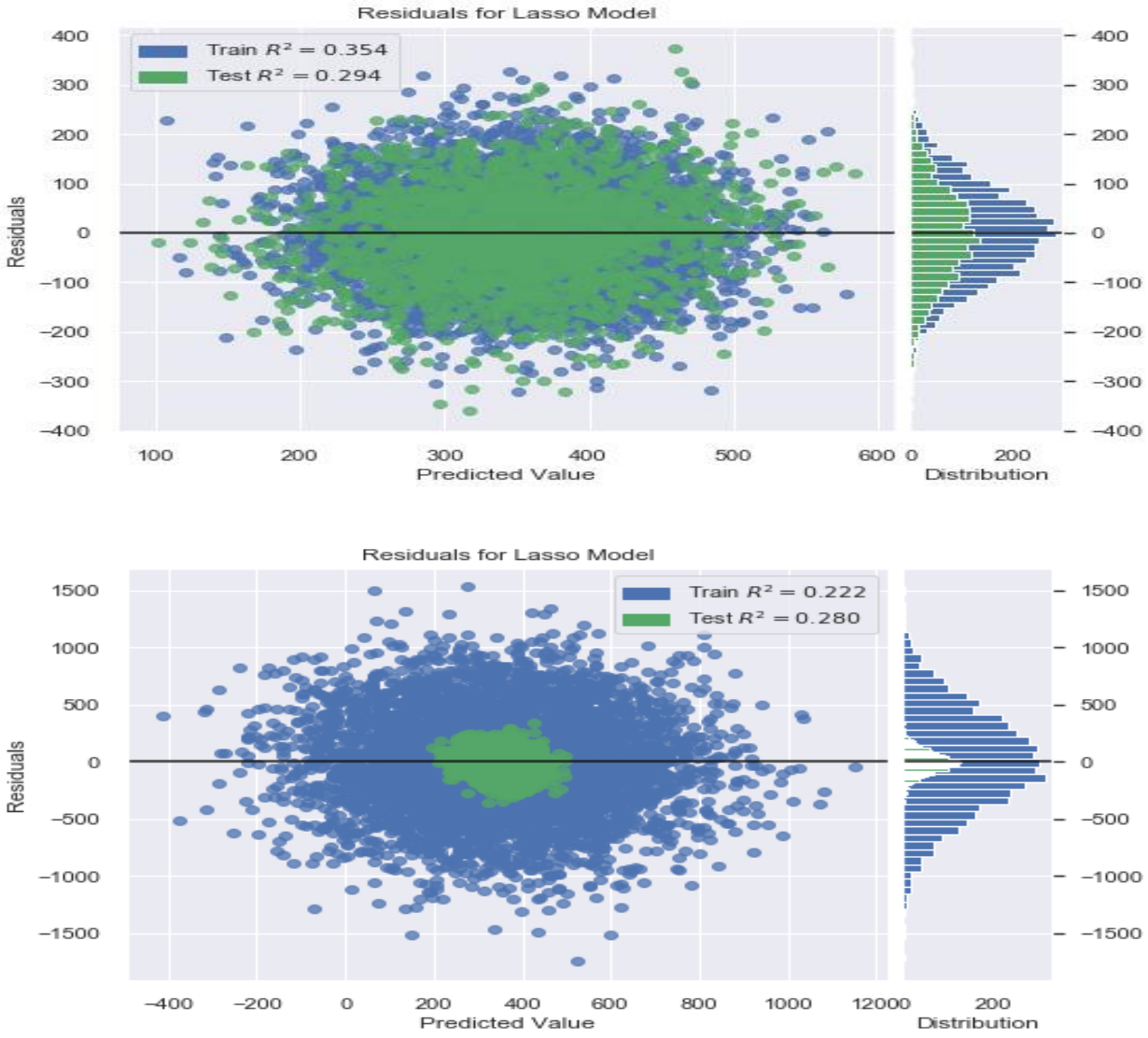


*Figure 9. The figure to the top corresponds to the low rank data residuals in the missing by feature setup and the figure to the bottom corresponds to its high rank counterpart.*

As I mentioned in the results section there is a discrepancy in the case of the random lr_big dataset in figures 4 and 7. The reason is explained in figure 10 below.

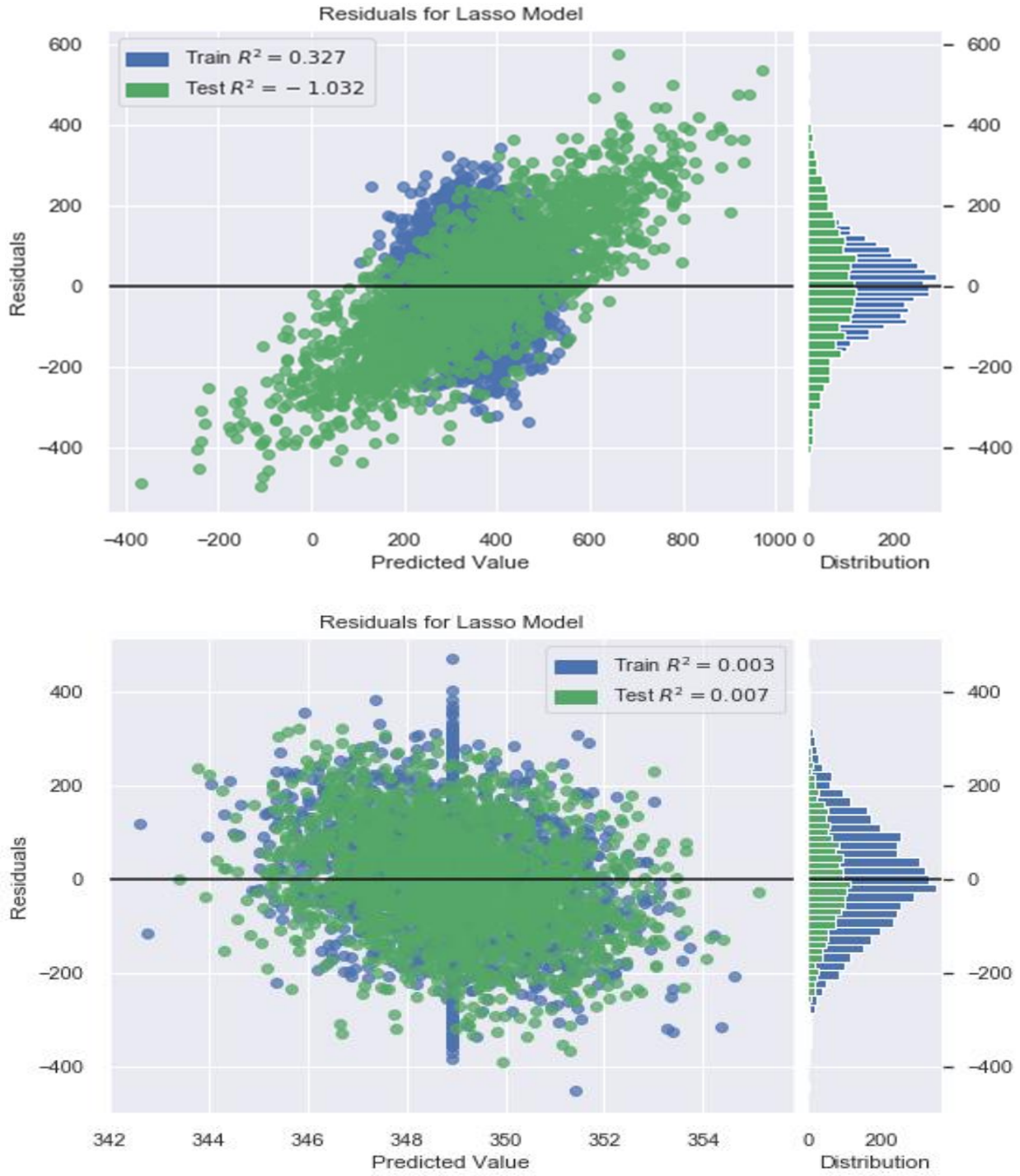


*Figure 10. This figure corresponds to one of the simulation in the random lr_big category. The top figure is using the best training parameter which is an alpha = 0.097 and the figure to the bottom uses an alpha of 10. We see that here the imputed and test datasets have very different distributions. By using a larger alpha and making the regression sparser we are perhaps able to remove some of the misleading imputed features and arrive at a better result. Figure 7 uses an alpha of 10 while figure 4 uses the alpha found in hyperparameter testing.*

## Soft imputation

First I will give an overview of soft imputation. The objective function of the soft impute algorithm tries to minimize the rank of the sparse data. However, it makes a modification of minimizing the nuclear norm (sum of the singular values of the dataset) to make the problem convex. Since soft impute is designed to give an optimal low rank estimation of a dataset with missing values. It makes sense that it has much better performance (compared to mean imputation) for such datasets especially when n is small, and notably in the case of random lr_big dataset. As seen from figure 11 below, low-rank datasets have lower and more zero singular values, as expected. Comparing the performance of the algorithm in the case of small and large n shows that most of the times large n has better or similar performance and as

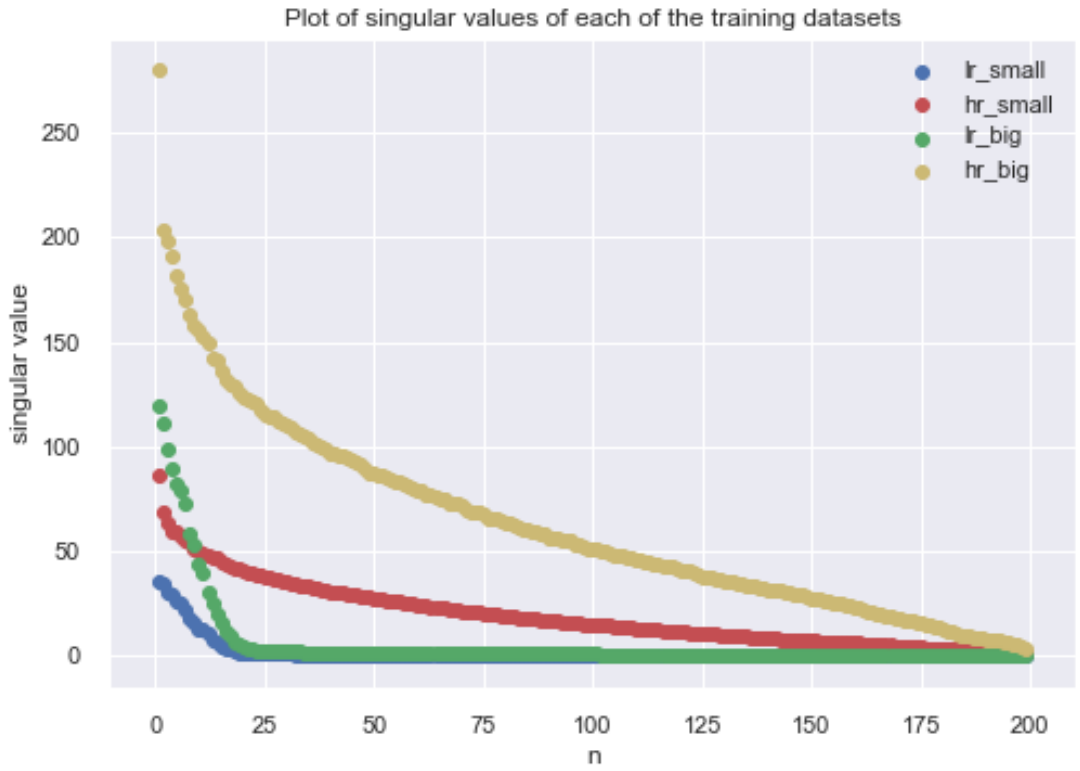


*Figure 11 The singular values of each of the training datasets are plotted in descending order. The maximal singular value was omitted from the plot to make it observable.*

expected large datasets show less variability. In the case of low rank vs high rank, there is a general tendency of low rank datasets achieving better scores than high rank ones. This is due to soft impute trying to minimize the nuclear norm and making a low rank estimation of the missing matrix. Low rank estimation of an otherwise high rank dataset will result in loss of information and creation of bias which promotes worse performance on the test sets. Comparing the two missing strategies missing by feature has similar or better performance than

the random strategy. Since only certain features are lost, there would be less bias in the performance, this is especially evident in the hr_small dataset. In the hr_big, due to large number of observations bias is less pronounced.

The variability in the large vs small cases here is attributable to the same characteristics discussed in the mean imputation part. Contrary to mean imputation, however, soft imputation shows less variability in the low-rank data compared to high rank ones. Depending on which features are affected more in the high-rank datasets, soft impute will try to make them dependent on others (by minimizing the nuclear norm), and this will have detrimental effects to the regression fit. However, with low-rank data, estimating a low-rank matrix is actually closer to their reality so they are affected less. The two missing strategies have more or less similar variability here. This is perhaps due to soft impute looking at the matrices as a whole and estimating missing values but also changing non-missing values to some extent, whereas in mean impute only features which have missing values are changed, for example.

## Feature Selection

The results of the feature selection for the mean and soft imputation are summarized in the tables below.

| | R_lr_small | F_lr_small | R_hr_small | F_hr_small | R_lr_big | F_lr_big | R_hr_big | F_hr_big |
|---|---|---|---|---|---|---|---|---|
| Importance of imputed in baseline model | 167 -> 12 (195)<br>100 -> na (137) *<br>178 -> 13 (194)<br>160 -> 4 (195)<br>144 -> 20 (185) *<br>139 -> 1 (200) | 97 -> 3 (199)<br>91 -> 6 (198)<br>34 -> na (74)<br>149 -> 19 (185)<br>100 -> na (137) *<br>63 -> na (155) * | 148 -> 3 (200)<br>172 -> 7 (200)<br>110 -> 11 (200)<br>39 -> 5 (200)<br>176 -> 8 (200)<br>140 ->23 (198) | Same as r_hr_small with the substation of<br>44 -> 30 (197)<br>Instead of 140 | 199 -> 7 (195)<br>186 –> na (113)<br>184 -> na (165) *<br>178 -> na (105)<br>172 -> 24 (182)<br>167 -> na (171) * | 140 -> 8 (195)<br>185 -> na (91)<br>167 *-> na(171)<br>3 -> na (164)<br>120 -> na(143)<br>100 -> na(143)<br>193 -> 2 (198) | 181 -> 24 (198)<br>176 -> 0 (200)<br>172 -> 8 (200)<br>148 -> 2 (200)<br>137 -> 1 (200)<br>122 -> 31 (196) | Same as r_hr_big with the addition of 140 (198) |
| Importance of baseline in imputed model | 199 -> 13<br>139 -> 13<br>65 -> 15<br>97 -> 2 *<br>160 -> 16<br>84 -> 4 *<br>Max:20 | 199 -> 9 *<br>139 -> 10<br>65 -> 12<br>97 -> 14<br>160 -> 12<br>84 -> 9 *<br>Max:14 | 97 -> 7<br>185 -> 1*<br>55 -> 1*<br>148 -> 19<br>40 -> 3<br>39 -> 14<br>Max:19 | 97 -> 13<br>185 -> 9*<br>55 -> 10*<br>148 -> 19<br>40 -> 12<br>39 -> 17<br>Max:19 | 137 -> 20<br>98 -> 17<br>193 -> 18<br>133 -> 20<br>55 -> 2 *<br>171 -> 19<br>Max:20 | 137 ->12<br>98 -> 7<br>193 -> 14<br>133 -> 10<br>55 -> na*<br>171 -> 11<br>Max:20 | 176 -> 20<br>137 -> 20<br>148 -> 20<br>54 -> 17<br>87 -> 5 *<br>49 -> 20<br>Max:20 | 176 -> 20<br>137 -> 20<br>148 -> 20<br>54 -> 17<br>87 -> 10 *<br>49 -> 20<br>Max:20 |

Table 1. This table summarizes the results of feature selection in the mean imputation setup. Only the 6 most important features were used here for analysis to save time and space. In the first row the numbers before the arrow are the most important features selected in the corresponding imputed dataset, in descending order. In the same row, the number after the arrow shows the importance of the selected feature (in the imputed dataset) in the selected features of the actual dataset and the number in the parentheses is the corresponding number of appearances of that feature as important in the 200 bootstrap samples. In the second row the numbers before the arrow show the important features in the baseline model in descending order. The numbers after the arrow denote the number of appearance of that feature as important in the 20 simulated imputed datasets corresponding to their respective columns. The max value shows the maximum number of times that a feature was selected across the 20 simulations in the corresponding dataset (column name). The red stars show examples of features that were not captured by the majority of the imputed datasets but were present in the baseline dataset. The green stars suggest features that could have substituted those features in the imputed dataset based on the number in the parentheses (the higher this number, the more likely it is a suitable feature to substitute). 'Na' stands for not available.

| | R_lr_small | F_lr_small | R_hr_small | F_hr_small | R_lr_big | F_lr_big | R_hr_big | F_hr_big |
|---|---|---|---|---|---|---|---|---|
| Importance of imputed in baseline model | 167 -> 12 (195)<br>199 -> 0 (200)<br>160 -> 4 (199)<br>144 -> 20 (185)<br>139 -> 1 (200)<br>100 -> na (137) * | 149 -> 19 (185)<br>34 -> na (74) *<br>91 -> 6 (198)<br>97 -> 3 (199)<br>184 -> 14 (193)<br>37 -> 21 (184) | 140.0 -> 23 (198)<br>39.0 -> 5 (200)<br>172.0 -> 7 (200)<br>176 -> 8 (200)<br>110 -> 11 (200)<br>148.0 -> 3 (200) | 172 -> 7 (200)<br>127 -> 21 (198)<br>148 -> 3 (200)<br>39 -> 5 (200)<br>137 -> 12 (200)<br>34 -> 6 (200) | 199 -> 7 (195)<br>184 -> na (165) *<br>178 -> na (105)<br>172 -> 24 (182)<br>171 -> 5 (197)<br>167 -> na (171) * | 125.0-> na (150)<br>139.0 -> na (145)<br>91.0 -> na (154)<br>100.0 -> na (143)<br>144.0 -> 19 (189)<br>167.0 -> na (171) * | 176 -> 0 (200)<br>172 -> 8 (200)<br>148 -> 2 (200)<br>137 -> 1 (200)<br>110 -> 19 (200)<br>97 -> 10 (200) | Same as the R_hr_big dataset |
| Importance of baseline in imputed model | 199 -> 18<br>139 -> 17<br>65 ->16<br>97 ->10<br>160 -> 17<br>84 -> 7 *<br>Max:20 | 199 -> 9*<br>139 -> 11<br>65 -> 12<br>97 -> 14<br>160 -> 12<br>84 -> 9*<br>Max:15 | 97 -> 11<br>185 -> 6 *<br>55 -> 8<br>148 -> 20<br>40 -> 3 *<br>39 -> 17<br>Max:20 | 97 -> 13<br>185 -> 9 *<br>55 -> 11<br>148 -> 17<br>40 -> 11<br>39 -> 17<br>Max:18 | 137 -> 20<br>98 -> 18<br>193 -> 17<br>133 -> 1 *<br>55 -> na *<br>171 -> 20<br>Max:20 | 137 -> 18<br>98 -> 12<br>193 -> 17<br>133 -> 1 *<br>55 -> 11<br>171 -> 11<br>Max:20 | 176 -> 20<br>137 -> 20<br>148 -> 20<br>54 -> na *<br>87 -> na *<br>49 -> 18<br>Max:20 | 176 -> 20<br>137 -> 20<br>148 -> 2 *<br>54 -> 20<br>87 -> na *<br>49 ->16<br>Max:20 |

Table 2. This table summarizes the results of feature selection in the soft imputation setup. Every number and every marking bear the same meaning as table 1.

The last figure in the next page compares certain results about the feature selection. Soft impute attributes less features as important in almost all the cases compared to mean impute. This is expected as soft impute lowers the rank of the matrix itself and accounts for correlations. This results in poorer performance especially in high ranked cases compared to mean impute. Mean impute finds more features in the case of large n, sometimes too much, whereas soft impute is less sensitive to the size of the input matrix, especially in the high rank cases. In the big high rank datasets, mean impute outperforms soft impute by a great margin as mean impute does not bias high rank datasets into low rank ones. The feature selection performs a bit better in the missing by feature setup. Since some of the features remain intact here, they might show better results especially in the low rank cases where already a few number of features are important.

Returning to the tables, one reason that some feature might be in the actual dataset and it does not exist in the imputed ones, might be the degree of missing values in that feature which are imputed. In the mean imputed data some feature might by chance fit better to the data because perhaps it's mean of existing values are close to that of an important feature and thus it arises as a new important feature. In the soft imputed case, if we see some feature in the baseline model which is not selected in the soft imputed data, it might be because that feature did not have a large singular value in the actual model

and it was omitted during the soft impute algorithm, or maybe that feature could be explained by other variables from soft imputes perspective. On the other hand, it is possible that missing values in one of the features in the simulated datasets caused that feature to lose its information in explaining the variance in the data (by for example decreasing its standard deviation) and the soft impute made that feature correlated on others and thus making it become less important.

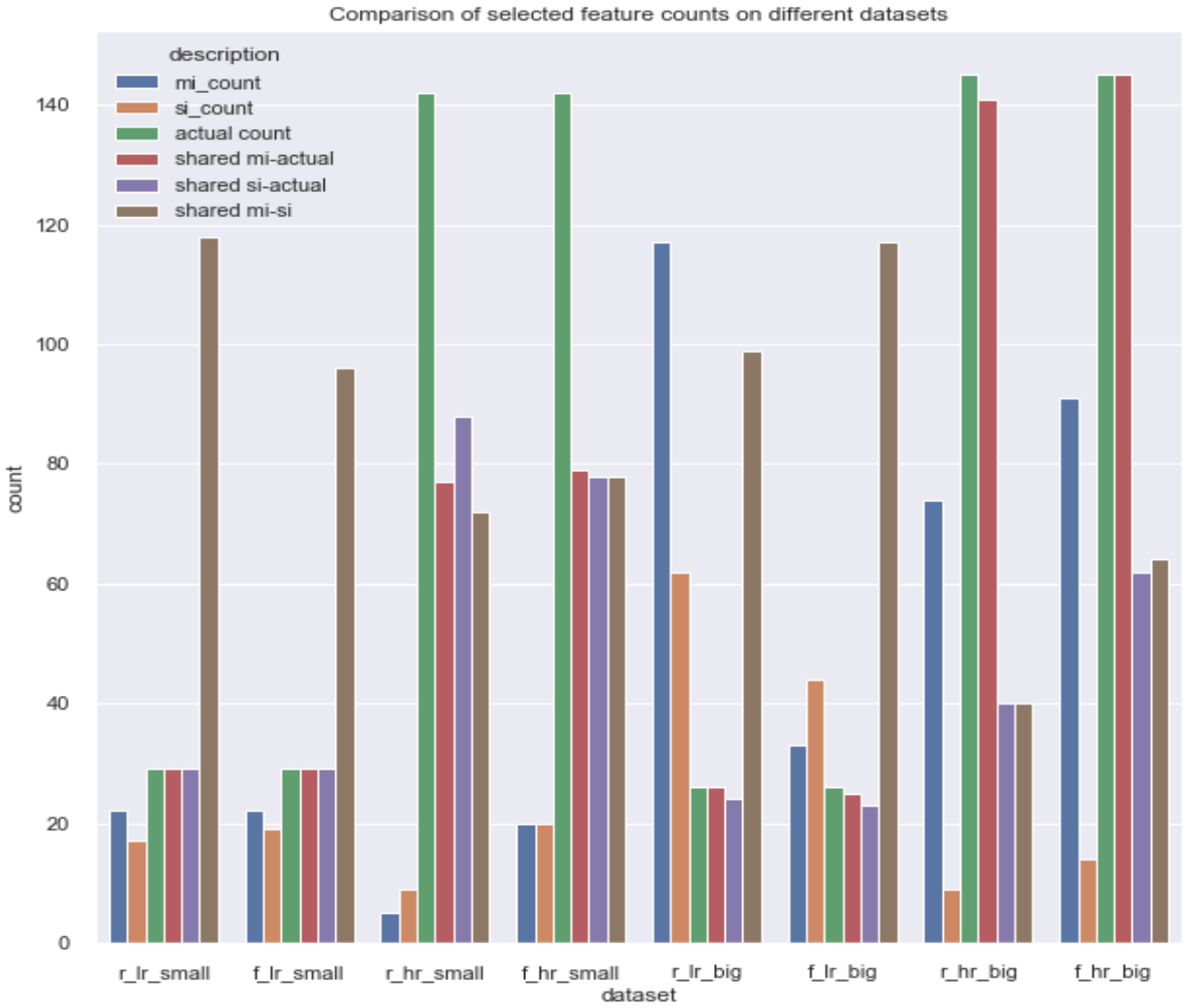


*Figure 12. Here the counts of selected features are presented for each dataset and each imputation algorithm. F in the dataset names means missing by feature and r means missing at random. Mi and si stand for mean impute and soft impute, respectively. To obtain the mean impute and soft impute counts only those features were included that appeared in more than half ( >10) of the simulations. To obtain the shared results between each of the methods, between themselves and the actual data, all of the found features were included, so this is why at some places shared counts are higher than mi- or si- counts.*