Final Exam Exercises

MVE440 Statistical Learning for Big Data

Trilokinath Modi

June 12, 2020

Contents

1 Exercise 1 **1**
1.1 Method 1
1.2 Results 2
1.3 Discussion 6

2 Exercise 2 **8**
2.1 Method 8
2.2 Results 8
2.3 Discussion 11

3 Exercise 3 **17**
3.1 Method 17
3.2 Results 18
3.3 Discussion 18

1 Exercise 1

This section describes about the project that deals with supervised learning for classification. A high-dimensional data-set was given for this task and we need find set of features that best explains the response.

1.1 Method

The complexity of a problem increases with increase in dimensionality, noise, type of problem at hand, multicollinarity and lot more. The dataset at hand is a high-dimensional data with a feature matrix as $X \in R^{323\times800}$. The response is a binary class variable. The objective is to train the chosen model on this feature matrix and a separate validation dataset was provided to evaluate the model. On exploring the data, it was found that the response variable is imbalanced in the ratio of $negative : positive = 3.25 : 1$. Class 1 will be considered as positive class for the rest of the project. First, the feature matrix was rearranged in a way that highly correlated features appear first and lowly correlated features at the end. This is employed just to ease in making the inferences after applying algorithm for feature selection. Although, it is a classification problem and logistic regression doesn't really need the features to be normally distributed but skewness check is done on the dataset and a log transformation for all the features which has high skewness than 0.25 is performed. A positive constant higher than least value is added to entire data set before log transform to avoid Nan's. The feature matrix is then standardised. All the operations are also done in validation dataset for uniformity of results.

The sole purpose of constructing a model is to find features which can explain response. But since this is all done by merely numbers, a model can easily fall into trap of overfitting, dimensionality curse etc. Hence feature selection is important. Wrapping techniques are greedy and computationally heavy. These techniques are highly dependent on training data and can alter the features selected with small changes in training data i.e. high variance. On the other hand, embedding based feature selection disregards features based on their poor performance to explain the variability by adding a penalty and can offer good balance between bias-variance trade-off. Therefore, in this project ℓ_1 lasso and elastic net regulated logistic regression are compared. The ℓ_1 lasso uses hard penalizing technique and sometimes it fails to retain highly correlated features in the model, where as elastic net offers balance between hard penalizing of features and ridge regression i.e. grouping of variables. Since, there is no information about the true model construction, both these methods are equally qualified for feature selection. Lastly, shrunken centroid was disregarded because it assumes that features in each class are independent Gaussian Distributions which is not the case here.

The hyper parameter for elastic net α needs to be tuned for optimal performance. If the true model is constructed using lowly collinear features then it is expected that the best performance will be achieved when α is close to 1. Here, $\alpha \in [0.1, 1]$ with increments of 0.1. It can be noted that ℓ_1 lasso is a special case of elastic net when $\alpha = 1$. Deviance is considered as loss function for both methods because it can be viewed as generalization of SSR in OLS. Prediction on validation dataset is done using the resulting model when hyperparameter $\lambda = \lambda.min$, where $\lambda.min$ is the lambda that gives minimum cross validation error. One can be interested in using $\lambda.1se$ which gives the most regularized model within 1 standard deviation of cross-validaton error but the fact that the sample size of one class is so low that any relaxation in cross validation error can potentially loose a important feature. Number of folds is kept at 17 to achieve a good proportion of training data in cross-validation. A important parameter used here is weights on observations in the ratio of class prior probabilities to ensure that the feature selection don't get complacent by performing good for dominant class. Hence, $w_i = \{1, 1, 1, 3.25, ...\}$ is used as weights for $Y_i = \{0, 0, 0, 1, ...\}$ respectively. However, a possible caveat using this technique is that if any observation of non-dominant class is outlier or incorrectly labelled then it can make the selection worse by a larger effect.

Both of the ℓ_1 lasso and elastic net will are executed 50 times and for each run, the influential features and the confusion matrix are recorded. Most influential features will be extracted based on how large is their frequency of being non-zero coefficients in the model after completing all runs. Running multiple times also balances out the error that could rise due to distribution of observations in different folds.

Generally, there are some prior information which can be used to decide the most interesting evaluation metric for a particular case. However, in this case, one can achieve $\approx 75\%$ accuracy by just random classification[1]. Hence Matthew's Correlation Coefficient, will be the key metric for presenting results. However, other metrics like Accuracy, precision, recall and F1 Score will also be used.

Feature importance ranking is finally done using a wrapper technique with AIC as metric. Backward selection procedure was employed on the model made by set of selected features. The confusion matrix resulting from before and after wrapper method is compared. In making of these confusion matrices, the model is created using entire training feature set and no cross validation and is performed. The reason for no cross validation is to let model explore entire training data at one go and the subset of features generated is expected to explain most of the variability and hence maynot need cross validation for this specific task of finding the ultimate important features.

1.2 Results

Histogram for kurtosis and skewness value is shown in figure 1. It was found that around 193 features had Kurtosis and Skewness larger than 0.25. Hence, a log transform was done on these features and to avoid numerical stability due to negative values, a constant [2] 50 was added to entire feature matrix element wise. The feature matrix is rearranged in a way that high correlated feature are the initial features and lowly correlated features end up at the end. This is the final transformed data on which ℓ_1 and elastic net regularization will be done.

Using the hyperparameters described in section 1.1 for e.g. $alpha \in [0.1, 1]$, the regularization was done on transformed training data, and prediction was done on transformed validation data. The predictions returned probabilities of response class being 1. Since the response set in training data is imbalanced, a threshold probability was decided based on ROC curve for each run. Here, the threshold probability for each run is computed as probability that maximizes the sum of specificity(TPR) and sensitivity(1 - FPR). The mean threshold probability of all the runs and all the α values is found to be 0.372219. A sample ROC curve is portrayed in figure 2.

Regularization effect for different values of α in elastic net model and ℓ_1 lasso i.e. $\alpha = 1$ can be seen from figure 3. As expected, the number of nonzero features decreases with increase in α. The interesting observation is that the model has good stability over all 50 runs and it can be seen by higher frequency in the grey blocks of figure 3. The grey blocks represent frequency of features that

[1]Random classification here means assigning everything to higher class. This term will be used again conveying same meaning.

[2]Any value above absolute value of minimum of feature matrix test and train can be picked.

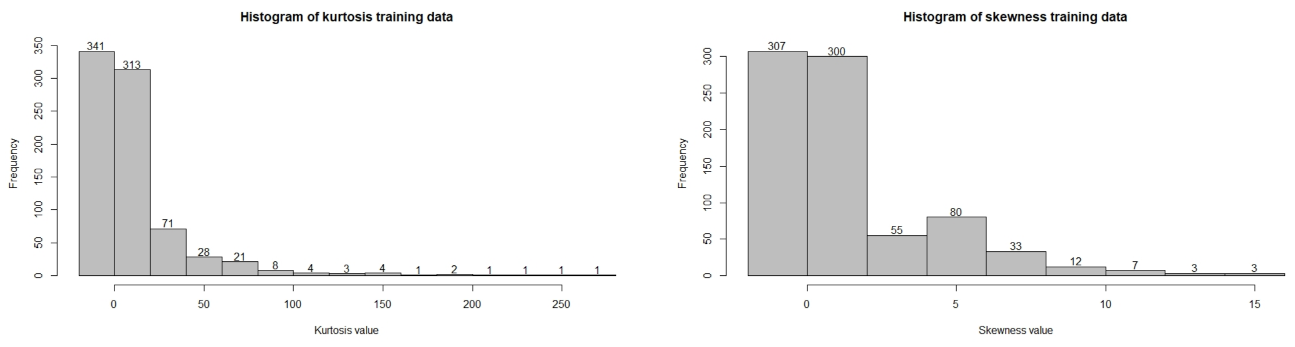


Figure 1: *The figure on the left shows frequency of feature kurtosis value, whereas the figure on the right portrays frequency of feature skewness values.*

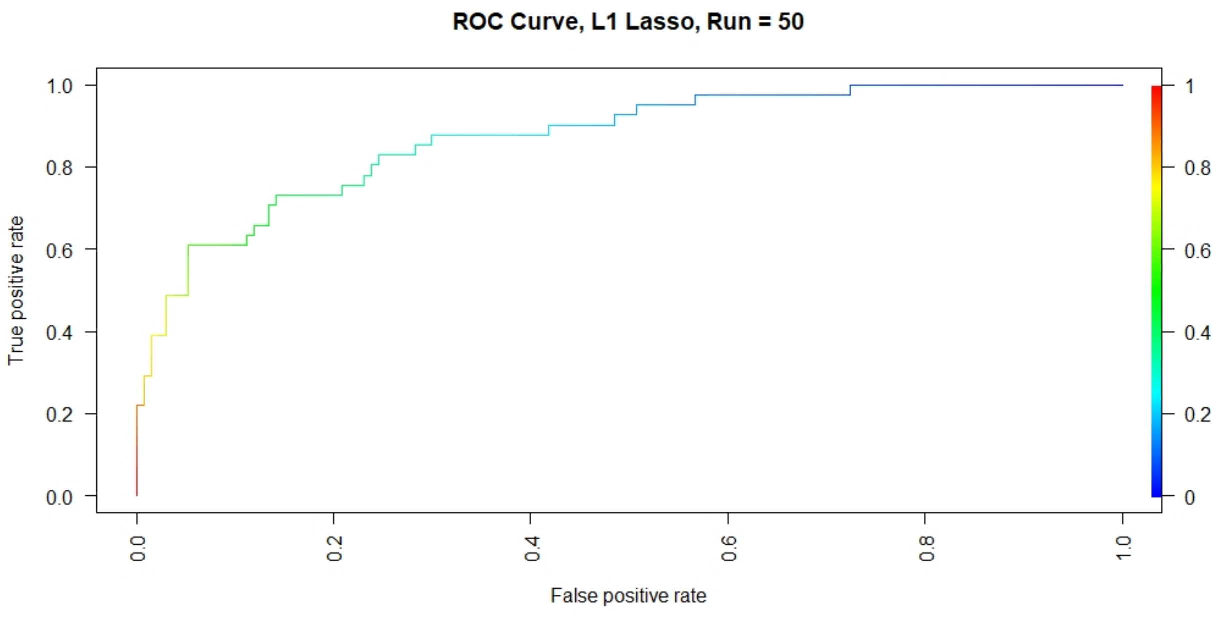


Figure 2: *ROC curve for ℓ_1Lasso, run = 50*

were set to non-zero coefficient for all 50 runs. Significant conflict between correlated features being selected could have resulted in bit larger Blue blocks.

The correlation conflict and feature dominance can be clearly seen through figure 4. Figure 4 indicates that features with average frequency ≥ 8 can be the set of features that explain most of the variability. Average frequency of ≥ 8 can be motivated using figure 3 and figure 4 combined. The basis of this statement is low frequency of highly correlated, refer figure 4 and decreased effect in regularization for $\alpha \geq 0.8$, refer figure 3. The number of features with average frequency of feature value $\geq 8 = 79$.

The performance of elnet and ℓ_1 is compared using evaluation metrics. At $\alpha = 0.5$, the ridge and lasso have near equal impact on deciding coefficients for features. However, for $\alpha > 0.5$, lasso starts to dominate. A very interesting observation from figure 5 that can be made is that as α increases from 0.5, the performance of the model is increasing almost linearly. Table 1 portrays the evaluation metric values. Although, the highest accuracy is achieved at $\alpha = 0.2$, it fails in maintaining a balance

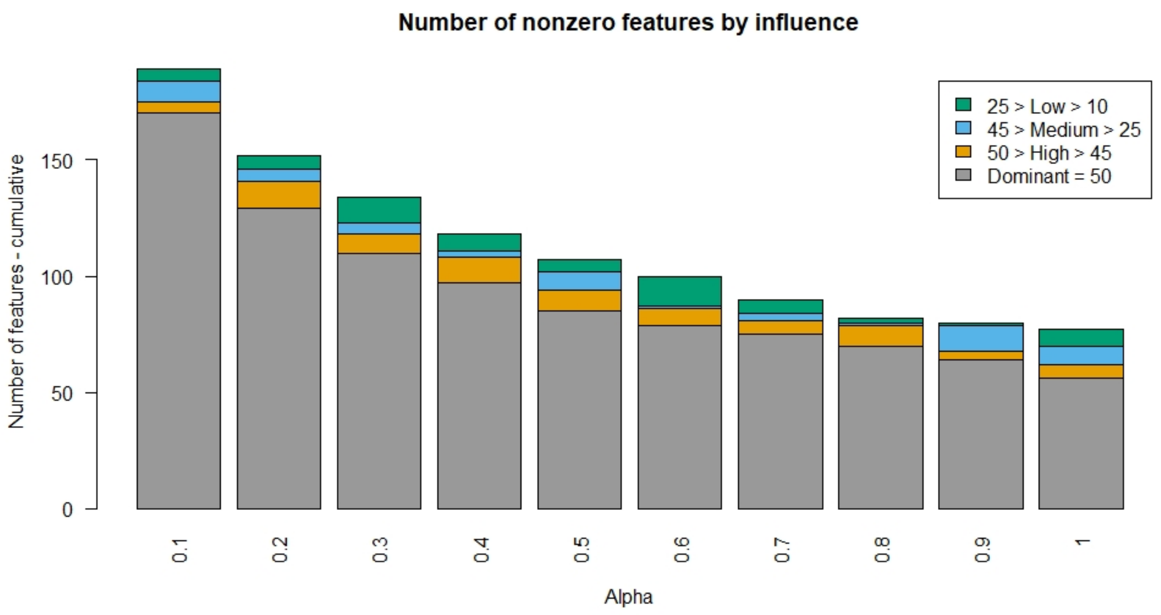


Figure 3: *Stacked barplot representing regularization effect on varying α.Referring to legend, dominant implies features that were non zero for all 50 runs, whereas high implies features that were non-zero for at least 45 runs but were not dominant. Similar inference can be easily drawn for medium and low. Features with negligible presence were discarded for this plot.*

Table 1: *Highest value obtained for each evaluation metric and corresponding alpha values. The highest value is obtained by comparing the average values of all runs for different alpha values*

	Highest	**Alpha**
Accuracy	0.78	0.2
Precision	0.56	0.3
Recall	0.85	0.8
F1 Score	0.63	1.0
MCC	0.52	1.0

between its prediction for both classes which can be supported by a relatively lower MCC value. Moreover, a random classifier can achieve accuracy of ≈ 0.75 but will result in $MCC = 0$. Hence, regularization didn't help in significant improvement of Accuracy but the model performs well in identifying lower dominant class which is very important. As the overall performance is increasing with increase in α, it motivates that the model can be still be improved. Hence, wrapping method is finally employed on subset of feature.

Finally, wrapper method is employed on all the non-zero features for which average frequency of feature ≥ 8, refer Figure 4. Hence, the start point for backward selection is the GLM model that is constructed on set of 79 features. The wrapper method chosen here for this task is Backward selection. Table 2 compares confusion matrix for pre and post wrapper method whereas table 3 portrays the metric values in both of these cases. The final set of features found using wrapper method is portrayed in table 4.

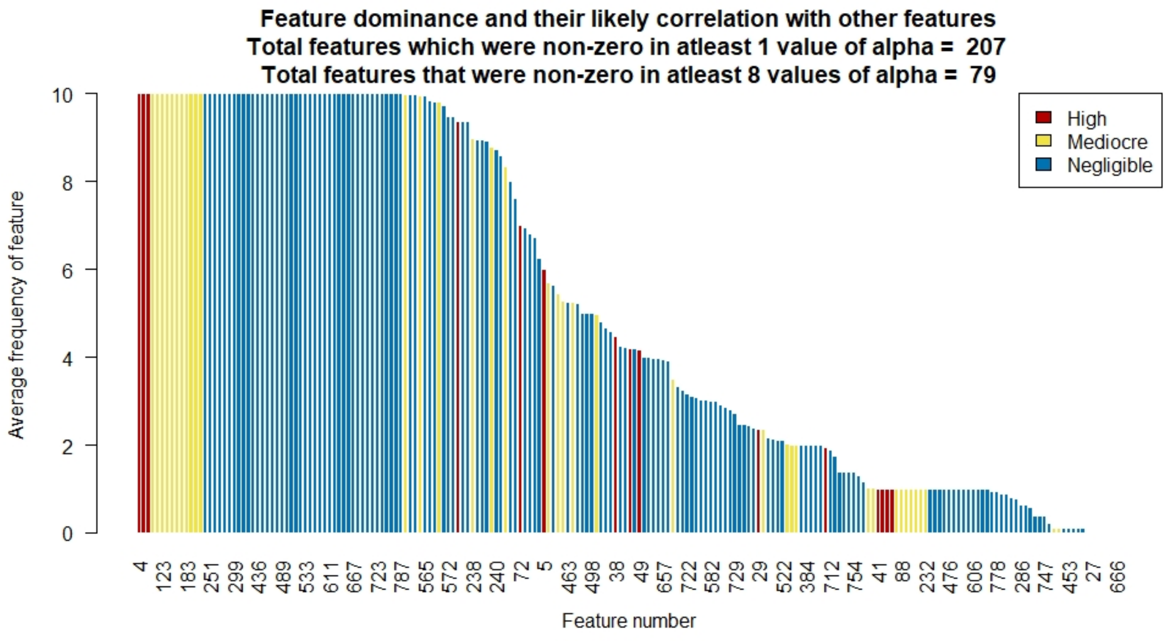
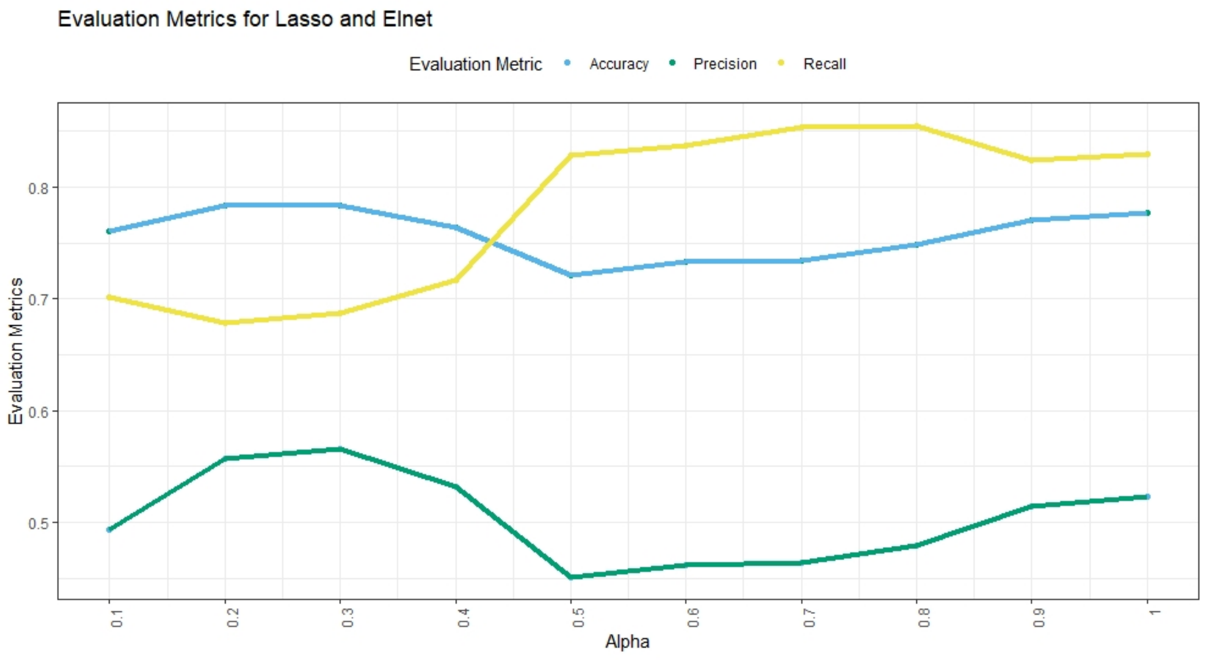


Figure 4: *The figure indicates feature dominance and feature property. Feature dominance implies the average performance of feature across all runs and all values of alpha which is indicated in Y-axis. An average frequency of feature = 10 implies the feature was non-zero in all 50 runs for all alpha values and an average frequency of feature = 0.5 implies the feature was non-zero for around 25 occurrence overall. In reference to the legend in the plot, the color denotes how likely the feature is correlated to other features in entire design matrix. A red implies the feature is highly correlated with at least 80 other features, whereas yellow denotes the feature is mediocrally correlated to at most 80 features and blue denotes the feature is almost independent.*



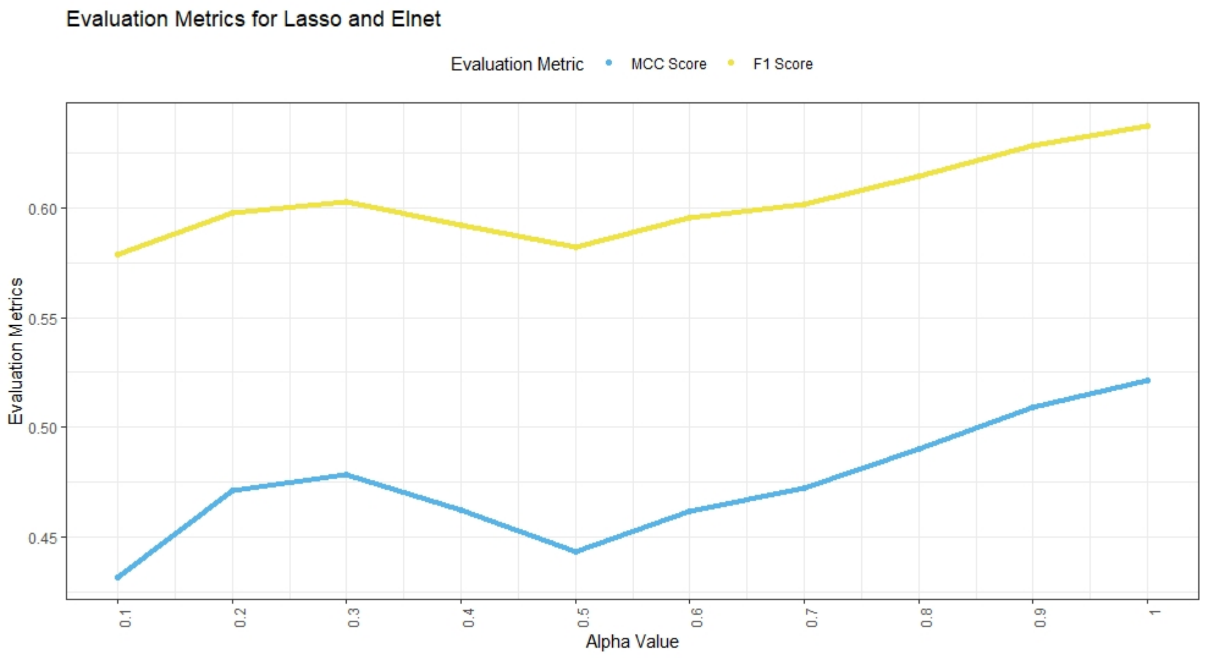


Figure 5: *Evaluation metrics based on confusion matrix. The metrics are computed by taking average over 50 runs for each alpha value. The ℓ_1 lasso is portrayed by $\alpha = 1$ and elastic net for other α values*

1.3 Discussion

Having a good number of columns being highly correlated, it was expected that elastic net will outperform Lasso. But, the figure 5 depicts that lasso slightly outperformed elastic net. The reason can be seen from figure 4 where we can see that the number of red features are low implying the true model was likely constructed by features that are largely uncorrelated. In such a case, lasso does the regularization well. Although it seems that lasso outperforms elnet but both of these methods couldn't penalise the features further. In other words, lasso could manage to bring down

Table 2: *Confusion matrix for GLM on a subset of features before and after wrapper method is employed. The subset of feature for before wrapper method is the set of feature which satisfy average frequency of feature ≥ 8. The subset of feature after wrapper method is the set of feature which wrapper method finds as the optimal set of features.*

P/R	**R0**	**R1**
P0	122	18
P1	12	23

P/R	**R0**	**R1**
P0	119	09
P1	15	32

Table 3: *Evaluation metric for before and after wrapper methods*

Metric	**Before Wrapper Value**	**After Wrapper Value**
Accuracy	0.82	0.86
Precision	0.65	0.68
Recall	0.56	0.78
F1 Score	0.60	0.72
MCC	0.49	0.63

the number of features from 800 to 56, but it still overfits a model to some extent. Hence, it still looses some points due to bias-variance tradeoff. Nevertheless, lasso provides a strong set of features that can be used by any wrapper method and final set of features can be determined. The final set of features described in table 4, out performs all other model. Nevertheless, it is based upon the features selected by lasso.

The confidence in the model can be described by two facts. One,the features selected having low correlation with other features supports the confidence and a MCC score of 0.63,refer table 3, boosts confidence on the performance of final set of features selected and they being true features.

Table 4: *Final selection of features. The green(FOM) implies they are the feature numbers in the matrix given for the task. The orange(FSM) implies they are the features of the shuffled design matrix based on collinearity.*

FSM	31	80	183	243	295	345	434	518	533	539	552	603	611	621	694	702	708	723	750
FOM	34	72	233	260	272	283	324	397	437	515	534	554	588	595	620	643	674	740	790

2 Exercise 2

This section describes about the project that deals with unsupervised learning for clustering. A high-dimensional data-set was given for this task and we need find optimum number of clusters and a set of features that can possibly explain the clustering.

2.1 Method

For this task, a high-dimensional data with a feature matrix as $X \in R^{302\times728}$. There is no response variable. The data is first scaled and then some basic analysis is done. The first basic analysis is PCA because it can sometimes give good insights to data like impact of principal components on explaining variability and projections of observations on key principal components. One of the simple clustering technique, K-medoids is used and corresponding Silhoutte Width for $K \in [3:32]$ is measured. A t-SNE plot with perplexity $Perp \approx \sqrt{N}$ is made as part of exploratory data analysis and to find hidden structures if any.

Model based clustering is then performed on entire data set for number of clusters in $K \in [2,9]$. Although BIC is favorable only when $N \gg p$, but it is still used here to make an initial guess on number of clusters. The initial guess of number of clusters is again used as input argument for model based clustering which is used to label observations as per the resulting clusters.

Variable importance is measure using Random Forest for which the input argument used is labelled observations as response variable and feature set as X. From the variable importance plot, a guess is made for average dimensions that can potentially explain each cluster formation. The guess is made by taking the average over length of elbows formed by Mean decrease in Accuracy and Mean decrease in Gini. The elbow estimates aren't accurate because random forest is susceptible to initialization. Nevertheless, subspace clusetring adapts to average dimesionality by having varying number of explainable features for each cluster and that may lower the imapct of non accurate guess.

The guess made for average dimension is then used as input parameter for projected clustering - Subspace clustering algorithm. Most of the methods discussed above as part of EDA or visualization often get cursed by large P. Subspace clustering algorithm are more robust in such scenario's. The features selected by ProClus on repetitive runs are visualized by their hits for clustering in every run. The algorithm is run for 50 times to achieve stable results. The ProClus algorithm needs number of clusters as input parameter hence, the input parameter of number of clusters will run iteratively for range $K \in [3:12]$. The algorithm doesn't necessarily cluster all observations and hence this can be used to detect outliers if any exists.

The optimal number of clusters will be thus be analysed using silhouette width on K-medoids clusters with varying K and feature set as set of 5 features selected by ProClus algorithm. Hierarchical clustering will be finally used to visualize if the selected 5 features is able to perform clustering. Hierarchical clustering won't be affected by dimensionality as it is greatly reduced. The structure of dendrogram can be used to make final judgements.

2.2 Results

Figure 6 portrays the projections of observations on principal components 1 and 2. looking at the PCA projection, we can say that there is some separation between two clusters and there seem to

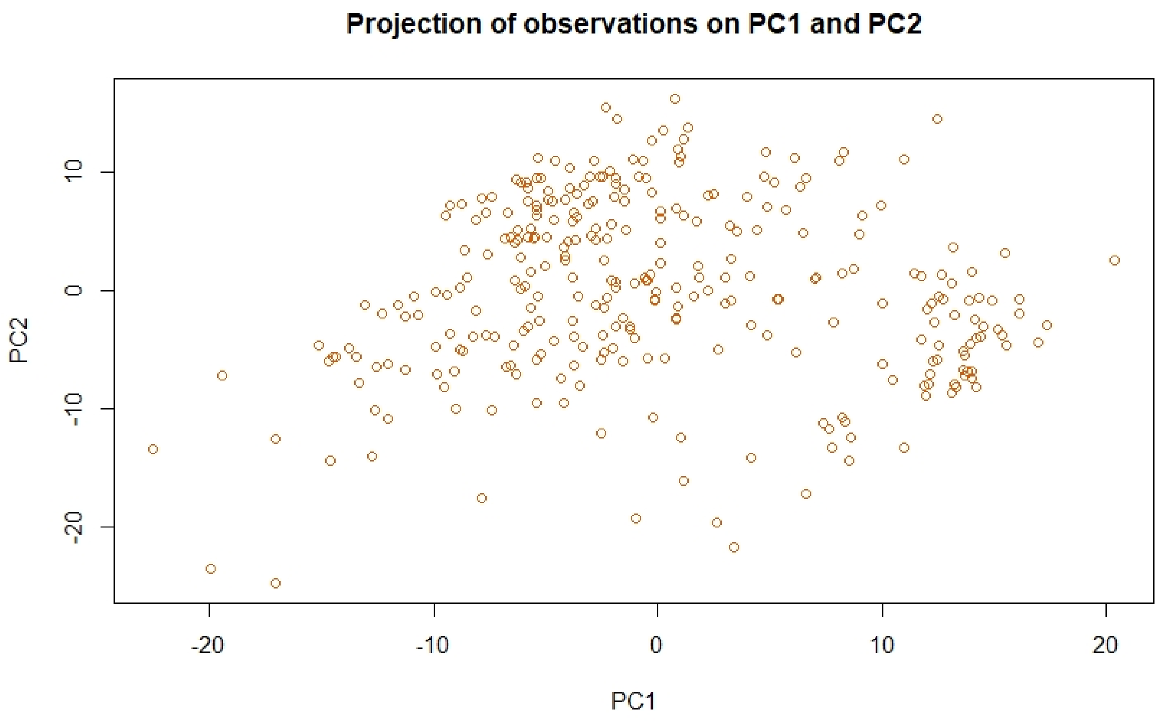


Figure 6: *PCA projection of observations on PC1 and PC2*

be some outliers too. However, it's difficult to say if the large blob is single or made up of small multiple clusters. From figure 7, we can say that merely few number of principal components is sufficient to explain most of the variability and by eyeballing it looks like around 15 is enough. Figure 8 portrays the observation projections on tSNE components. The hyperparameter used is $Perp = \sqrt{N} \approx 17$ and varying iterations. Clearly, there are atleast 3 clusters and it's not very clear if the large blob is just one or many different clusters.

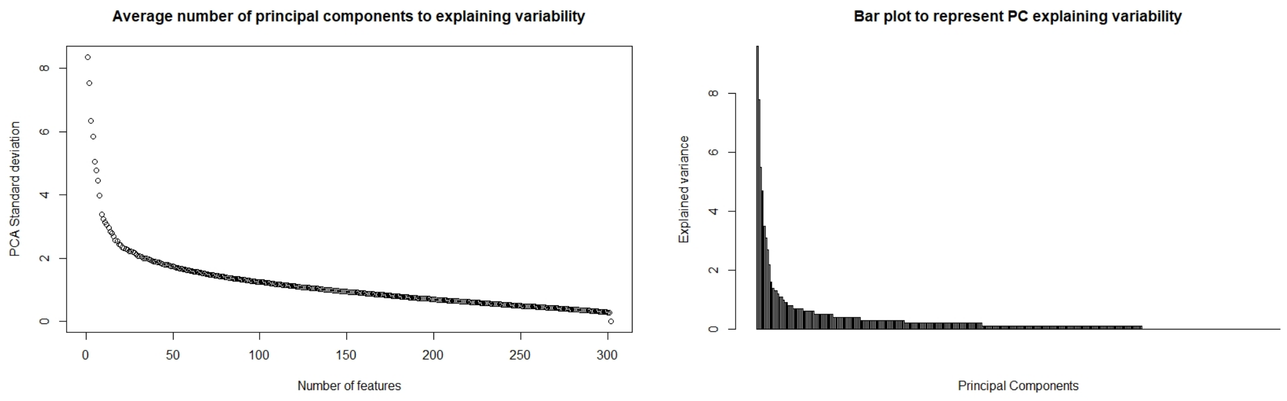


Figure 7: *The figure on the left gives information about average number of principal component that explain variability. The same is depicted in right with bar plot.*

The silhouette width achieves maximum score at $K = 2$ clusters. Any discriminant analysis would have worked greatly for 2 clusters if applied on PCA projections. But the fact that the difference in silhouette width is very small for $K = 2, 3, 4$ undermines that the optimal number of clusters

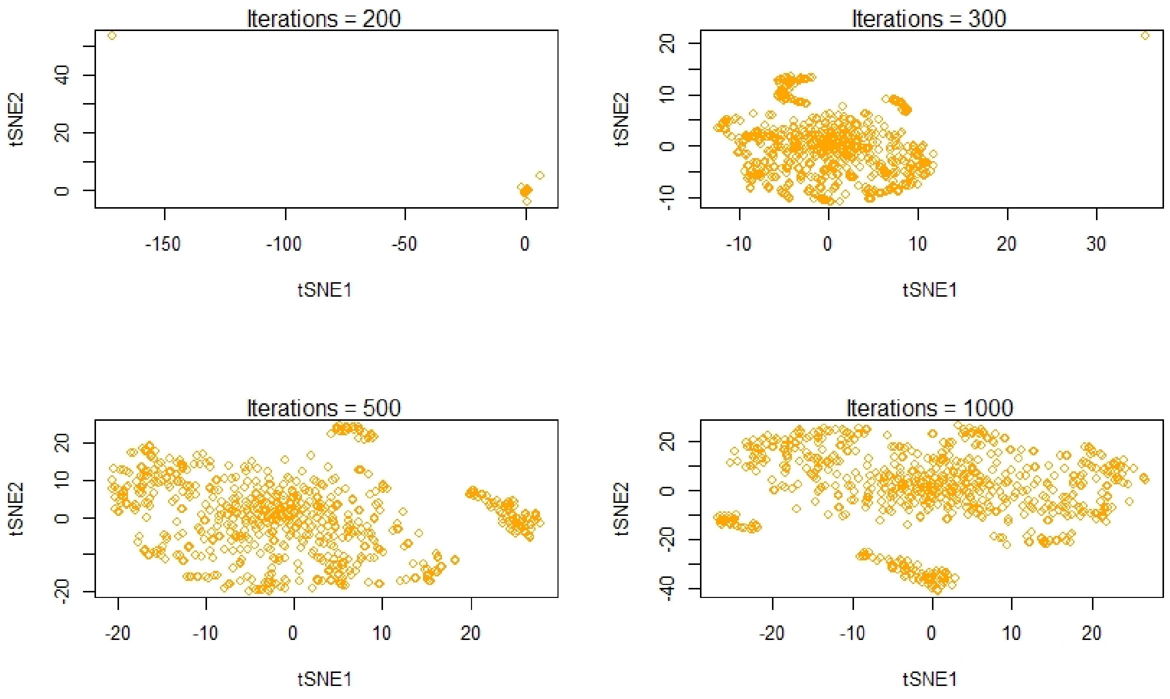


Figure 8: *Projection of observations on t-SNE components, Perplexity for all these cases = 17*

is 2. Moreover, on close look, the difference in silhouette width isn't very large as number of clusters increases. This can be explained with the fact that distances loose it's importance in high dimensional case.

A model based clustering is done on number of features $K \in [2, 9]$ and the resulting BIC graph is presented in top row of figure 10. The resulting clusters for $K = 8$ is is portrayed in bottom row of figure 10. Clusters generated from this model based clusering are then used as class labels for variable importance using random forest. The primary reason for using MB clustering to identify class labels is that, model based clustering tries to identify distributions. The average dimension hyperparameter for ProClus algorithm is guessed based on the number of variables that are identified as important by the random forest algorithm. Figure 11 portrays feature importance. A higher value in mean decrease in accuracy or a higher value in mean decrease in gini index implies the feature is important. Average of $freq(Mean\ decrease\ in\ Accuracy > 4)$, $freq(Mean\ decrease\ in\ Gini > 2)$ is used as the average dimensions of cluster for ProClus algorithm and the average was found to be 14.

Each run of the ProClus algorithm involves clustering with input number of cluster in $K \in [3, 12]$. And for each of these case, the observations clustered and features responsible for clustering was recorded. The figure 12 represents frequency of features that were impactful for clustering the data. Note that, the x-axis represent the feature hit value and y-represent frequency of features that has achieved this hit value. Unsurprisingly, there is negligible contribution from 643 features with a hit rate close to 0. The top 5 features with decreasing order of hit rate is considered as final selection. Since ProClus doesn't necessarily cluster every observation and hence, a histogram for observation

[3]The construction of this particular plot is motivated from stackoverflow, however, the idea behind concepts used is original work

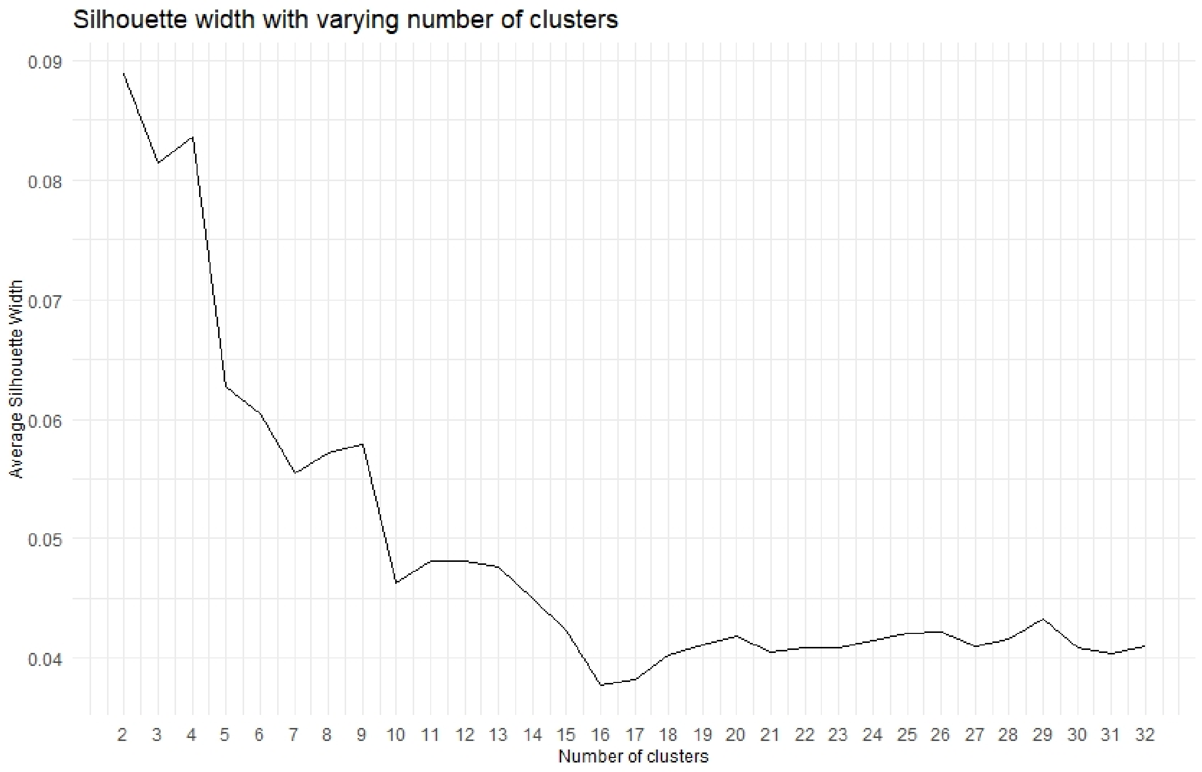


Figure 9: *Silhouette width for clusters ranging K = 2 to K = 32 using K-Medoids.*

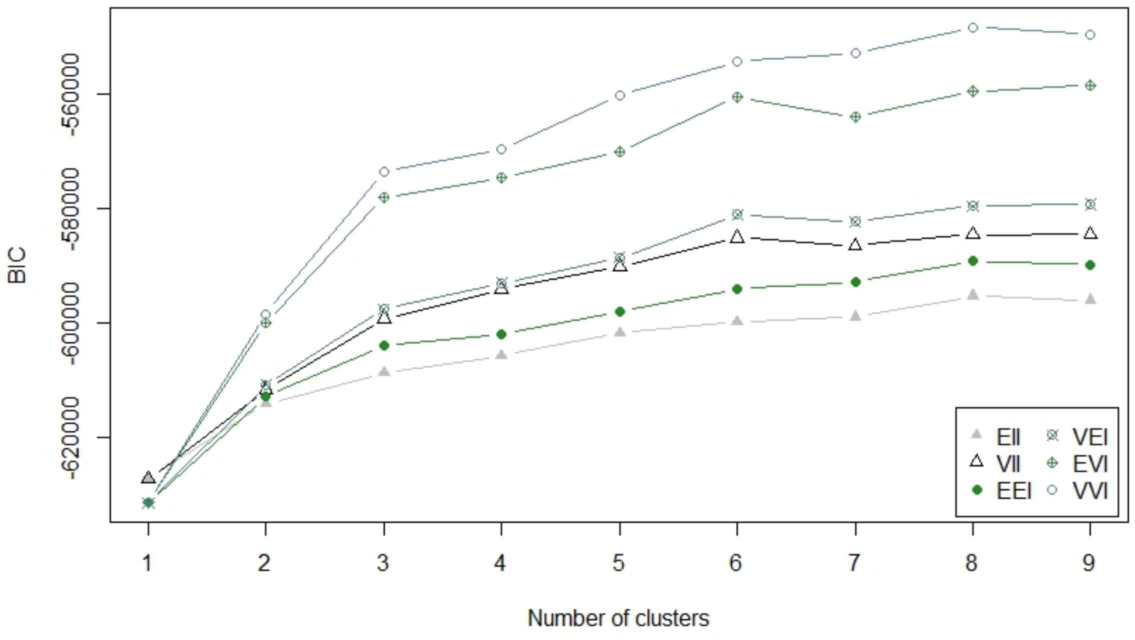
hits is portrayed in figure 13. The observation clustering index(OCI) implies the average number of times the observation will be clustered if number of cluster changes in $K \in [3, 12]$. Hence, an OCI of 5 implies the observation is clustered in at least 5 times when number of clusters is changed from $K = 3$ to $K = 12$. The histogram in figure 13 produces important inferences which are mentioned in caption of the figure 13.

Clustering is performed using K-medoids with varying K on dataset with the 5 selected features. 7 possible outliers were also removed from the same as per figure 13. The sihouette width is displayed in figure 14. To see if features explain clustering, a heirarchical clustering is performed and it's dendrogram is portrayed in 15. Goodness of feature selection is discussed in section 2.3.

2.3 Discussion

The dendrogram can be interpreted in a way that a huge blob of observations is clustered as one class and four observations far apart are being clustered as different classes. Hence, the feature selection can be concluded as poor. This can be explained with the fact that the feature which could possibly separate these blobs absent from the set of final selected features.

One thing that could possibly went wrong is collinearity in features. If the deciding feature is correlated with other then it may decrease the total hit score for all these features. In such a case, the program can be made in a way that highly correlated features can be grouped and their total hit score can decide if all of these features should be further analysed or if they can be disregarded altogether. Other possible way to tackle this problem is to combine highly correlated features by taking average and analyse separately if the combined feature achieves a good hit score.



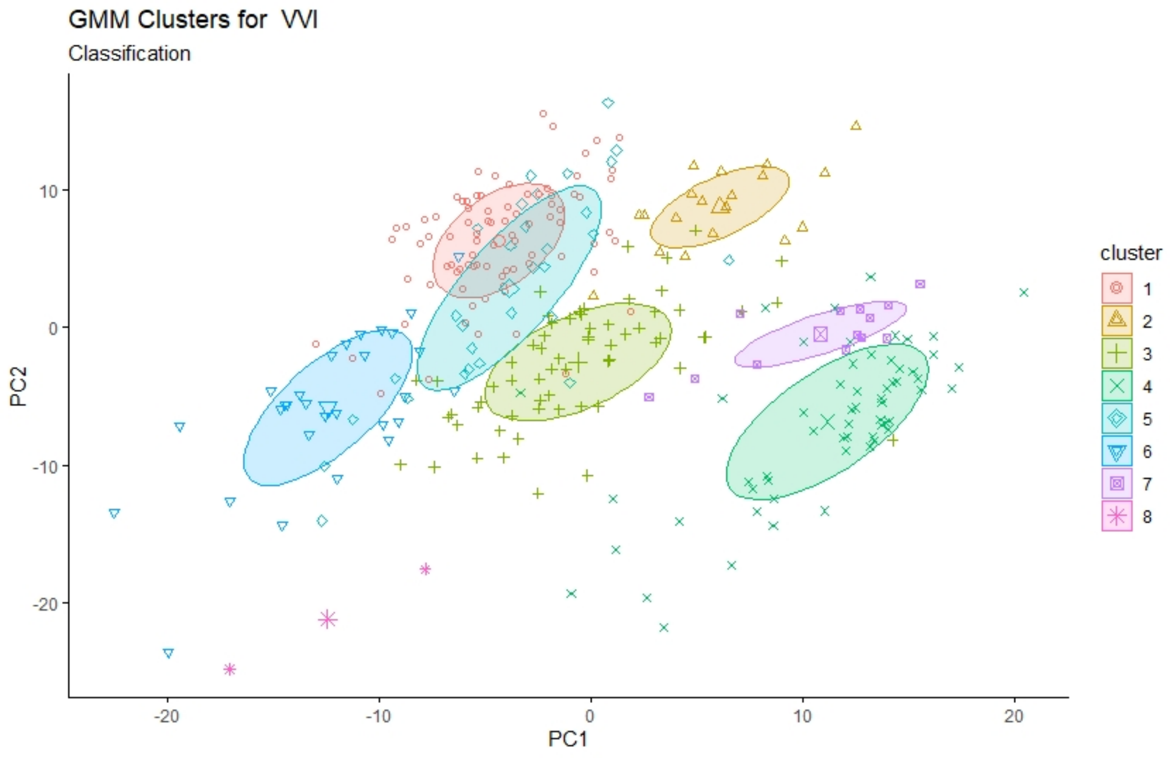


Figure 10: *The figure at top represents BIC value for different model type and varying number of clusters. The highest BIC is achieved at 8 clusters and hence the corresponding model based clustering is portrayed on bottom figure.Footnote*[3]*regarding figure.*

Other, possible issue can be ProClus being repeatedly trapped in a cluster formed by possibly few outliers forcing other blobs to combine and missing out the feature that can actually separate the

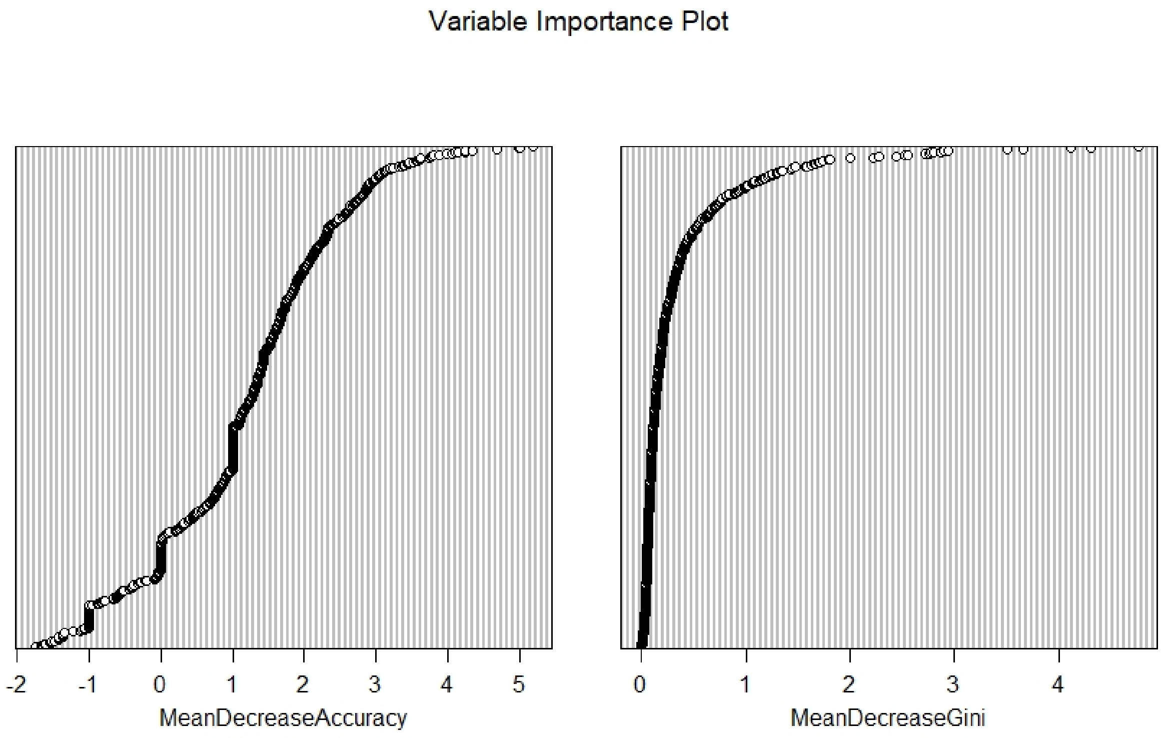


Figure 11: *variable importance using random forest*

two combined blobs.

To summarise, the hyperparameter selection for ProClus needs additional care. A constant watch on collinear features could improve the feature selection.

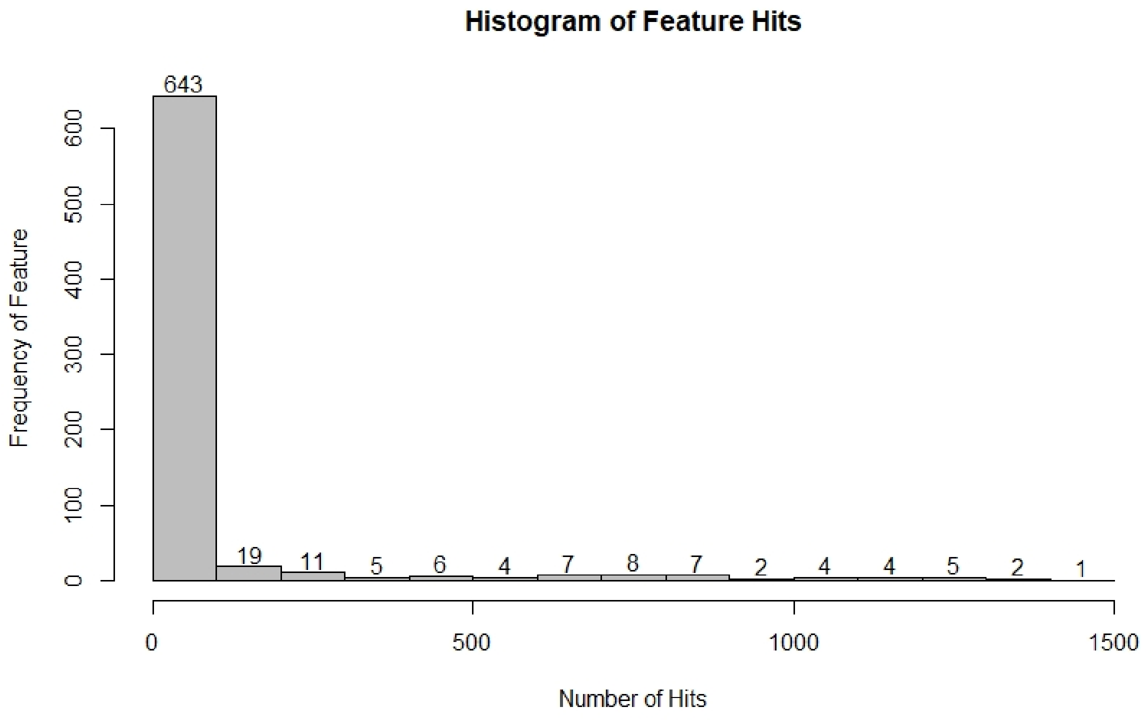


Figure 12: *Histogram of feature hits represents frequency of features that hit a particular score. The score is presented in x-axis. The left bar indicates that 643 features have near negligible contribution to explain variability and the right bars indicate features that are most likely to decide clusters.*

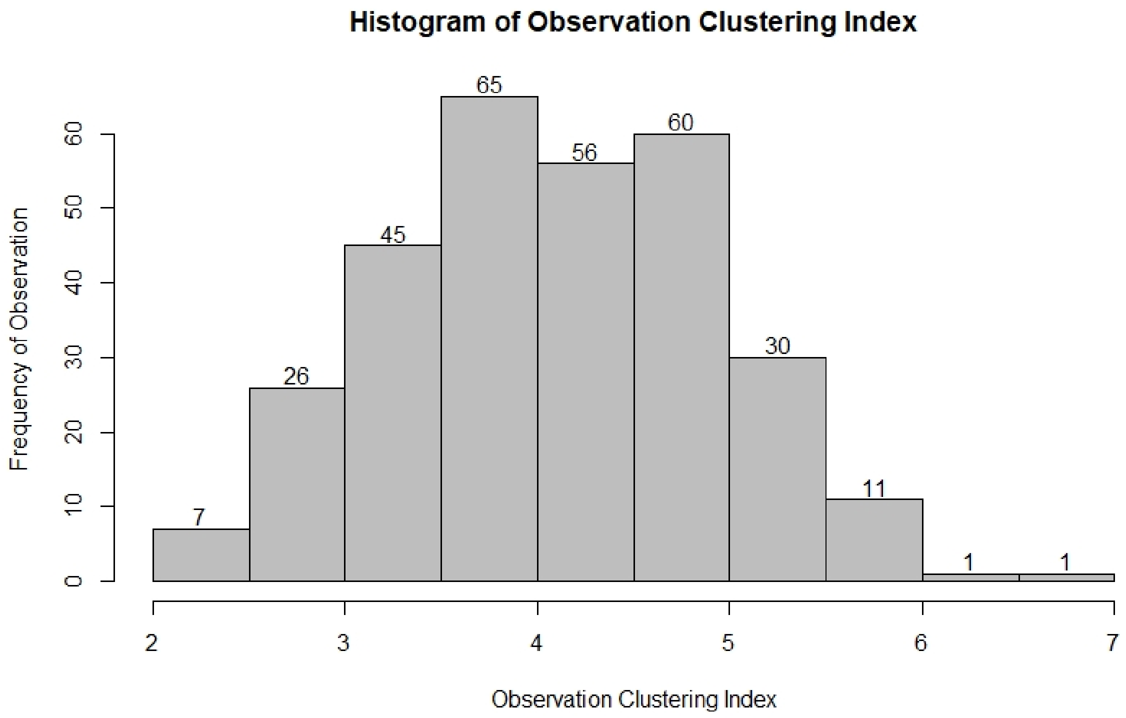


Figure 13: *The histogram represents how frequency of observations that are clustered only OCI times when number of clusters in ProClus algorithm is changed from $K = 3$ to $K = 12$. The x-axis represents observation clustering index. The left most bar represents 7 observations are clustered very few times implying they are possible outlier or in other words, neither density connected, nor density reachable. The right end bar's represent observations that are prone to being clustered incorrectly for large number of clusters or they can be good core points for small number of clusters.*

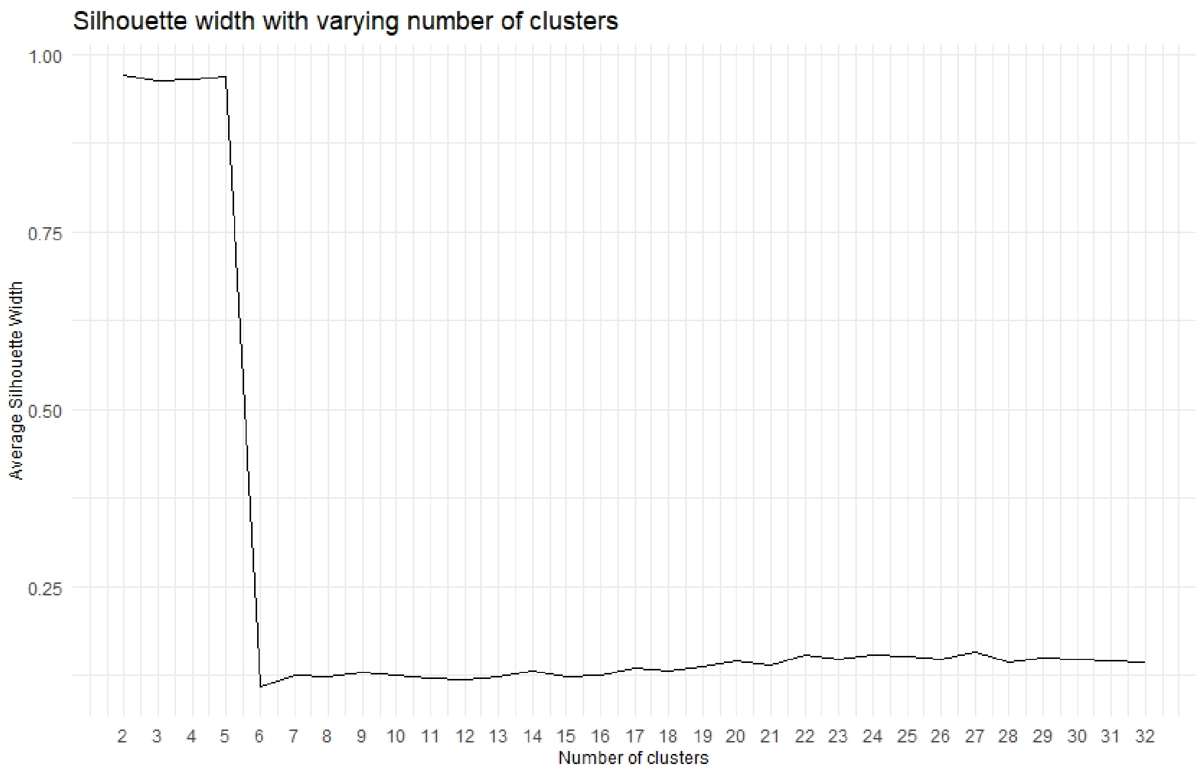


Figure 14: *Silhouette width using selected features and excluding outlier detected observations*

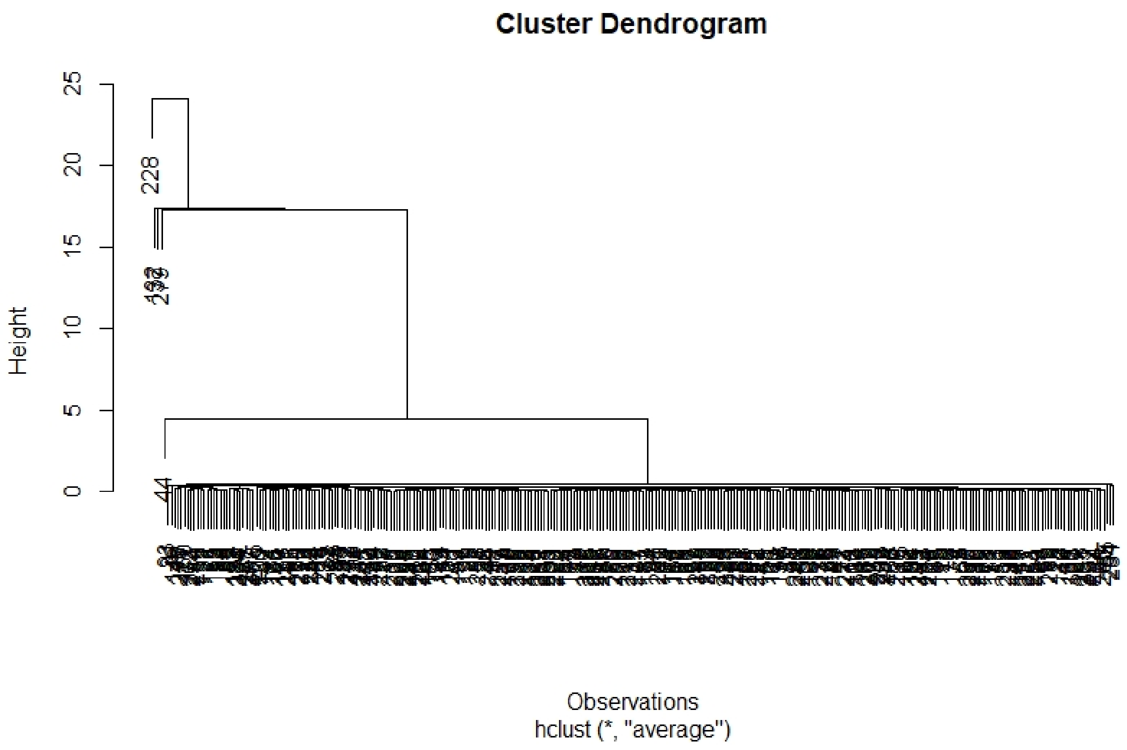


Figure 15: *Dendrogram using average linkage*

3 Exercise 3

This section describes about the project that deals with robustness of adaptive lasso. The problem type is regression and typically focuses on how false features can be penalized and set to become a zero coefficient feature adaptively. The central idea behind adaptive lasso is that the standard lasso fails to be consistent with Oracle properties where as the adaptive lasso enjoys oracle properties if weights selected can be data-dependent.

3.1 Method

Lasso is predominantly used technique for feature selection with residual sum of squares as loss function for regression problem or deviance for classification problem. However, Zou in 2006 showed that lasso doesn't accurately recover the true subset of features in some setups. And Zou suggested Adaptive Lasso which penalises features based on weights. The robustness in penalisation comes with the fact that the weights are data-dependent. Here, performance of adaptive lasso is tested using simulated data is compared to standard lasso.

A total of 50 sets of data is generated. Each set contains 7 sets of dataframe. These 7 dataframe is mainly because of varying number of observations. Each dataframe is formed using a constant number of features i.e. $p = 50$. The true non-zero beta coefficients for each data frame will be for the first 5 features. The number of observations will follow a geometric progression sequence $N = \{100, 200, 400, 800, 1600, 3200, 6400\}$ and for each N, the feature matrix $X \in R^{N\times 50}$ is simulated such that each variable is Gaussian Distributed $X_i \sim \mathcal{N}(0, 1/\sqrt{N})$. The true beta coefficients are generated from Gaussian distribution $\beta_j \sim \mathcal{N}(0, 1)$. The resulting model is $Y = X \cdot \beta + \epsilon$, where $\epsilon \sim \mathcal{N}(0, \sigma^2)$, $\sigma = \sqrt{\sum Y^2/(N-1)}/SNR$ and signal to noise ratio $SNR = 1$. All in all, there are 350 different data frames and each data-frame is executed 50 times(runs) for stable results. Each of these run involves cross validated feature selection by lasso and adaptive lasso. The hyperparameter λ is tuned by cross validation for both lasso and adaptive lasso and is done by the package. The hyperparameter γ can take value $\gamma = \{0.25, 0.5, 1, 2, 4, 8\}$ and performance on each of these γ value is measured.

Weights for adaptive lasso can be assigned based on OLS estimates and γ_j value as $W_i = |\hat{\beta}^i_{OLS}|^{-\gamma_j}$. A transformed design matrix $X' = X \cdot diag(W)^{-1}$ is used with response variable Y as input parameters for standard lasso regression. The resulting estimated coefficients are $\hat{\beta}'_{Lasso}$. Finally, the estimated adaptive lasso coefficients can be computed using $\hat{\beta}_{AdaLasso} = diag(W)^{-1}\hat{\beta}'_{Lasso}$.

The estimated adaptive beta coefficients and the original beta coefficients are transformed to factors such that a non-zero value implies positive class 1, whereas a zero coefficient implies negative class 0. Confusion matrices are drawn considering the estimated beta coefficients $\hat{\beta}_{AdaLasso}$, $\hat{\beta}_{Lasso}$ as prediction and true beta coefficient β as reference.

The area of interest lies in both true positive and true negatives and hence evaluation metrics like specificity, sensitivity and Matthew's correlation coefficient are used to compare the performance of lasso and adaptive lasso. Confusion Matrices for each of these 350×50 data frames is stored. The evaluation metric values are averaged over 50 runs for each N in a single set of data. since, there are 50 sets of data for each N, mean of averages is computed and it represents the corresponding evaluation metric for both adaptive lasso and standard lasso respectively. The average is taken in a way that the performance of γ can be evaluated.

3.2 Results

Figure 16 and figure 17 represents performance of differnt γ on dataset with varying N. Table 5 and table 6 are tables to compare the performance of Adaptive Lasso and standard Lasso. The gamma values used to compare adaptive lasso with standard lasso are $\gamma = 0.5$ and $\gamma = 4$. The same thing is also represented in Figure 18 but the table comparison is much more readable.

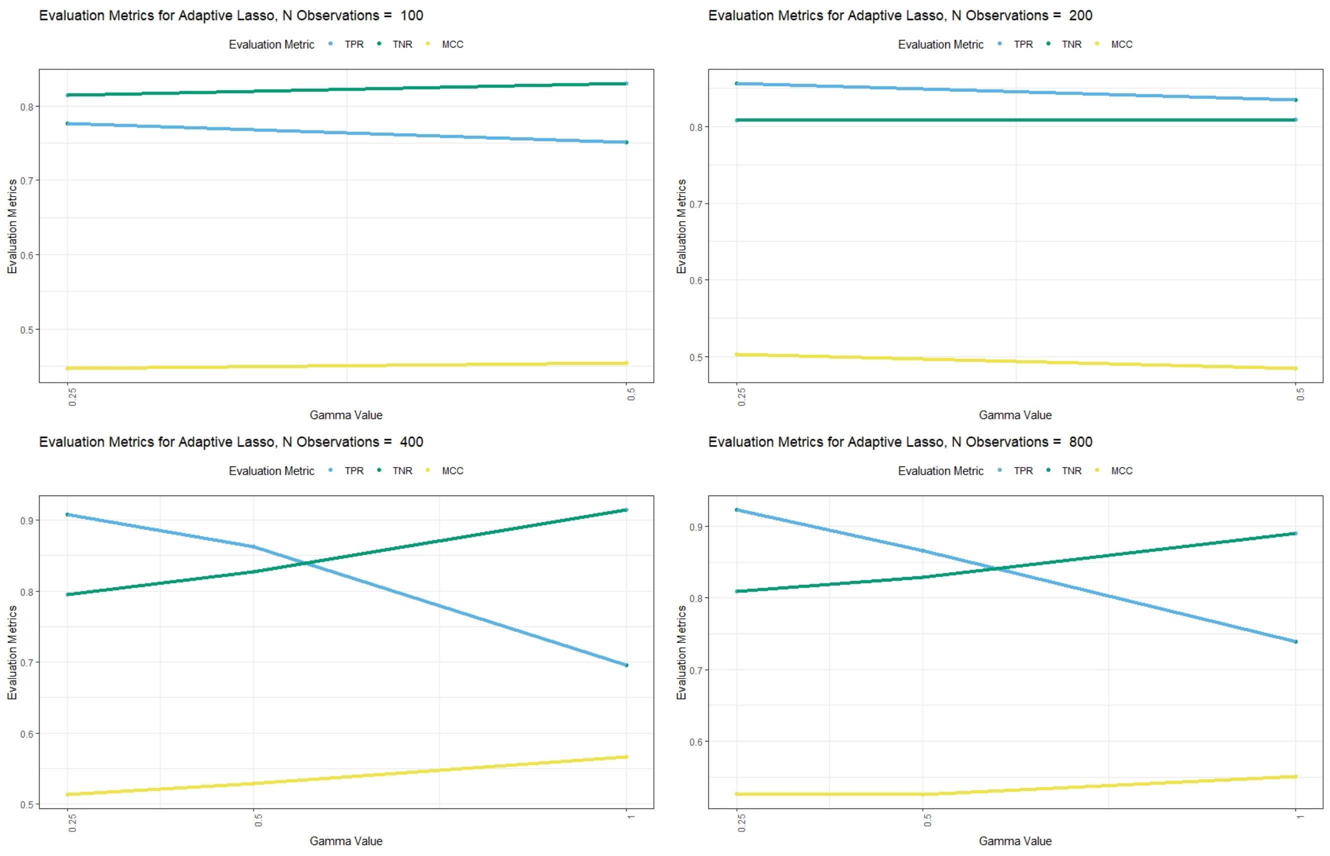


Figure 16: *Performance of adaptive lasso with varying γ and number of observations. The plot on the top left represents evaluation metrics taken by average over all runs and all sets of data with 100 observations. Similarly other plots are for N = 200,400,800 respectively arranged in row-major matrix fashion.*

3.3 Discussion

Few underlying observations are, as number of observations increases, the standard lasso ability to eliminate zero coefficients(true negatives) is extremely high and works better than adaptive Lasso. This can be seen from table 6. However, the adaptive lasso ability to identify true positives is dominant as compared to standard lasso. This can be seen in table 5. Few points in regards to the results obtained-

- For any value of N, to achieve a high TPR, prefer adaptive lasso with smaller γ i.e. $\gamma < 1$
- For small value of N, here $N = 100$, to achieve a higher TNR, prefer adaptive lasso with larger γ i.e. $\gamma > 1$.
- For larger values of N, here $N > 200$, prefer standard lasso to achieve higher TNR. Also, in

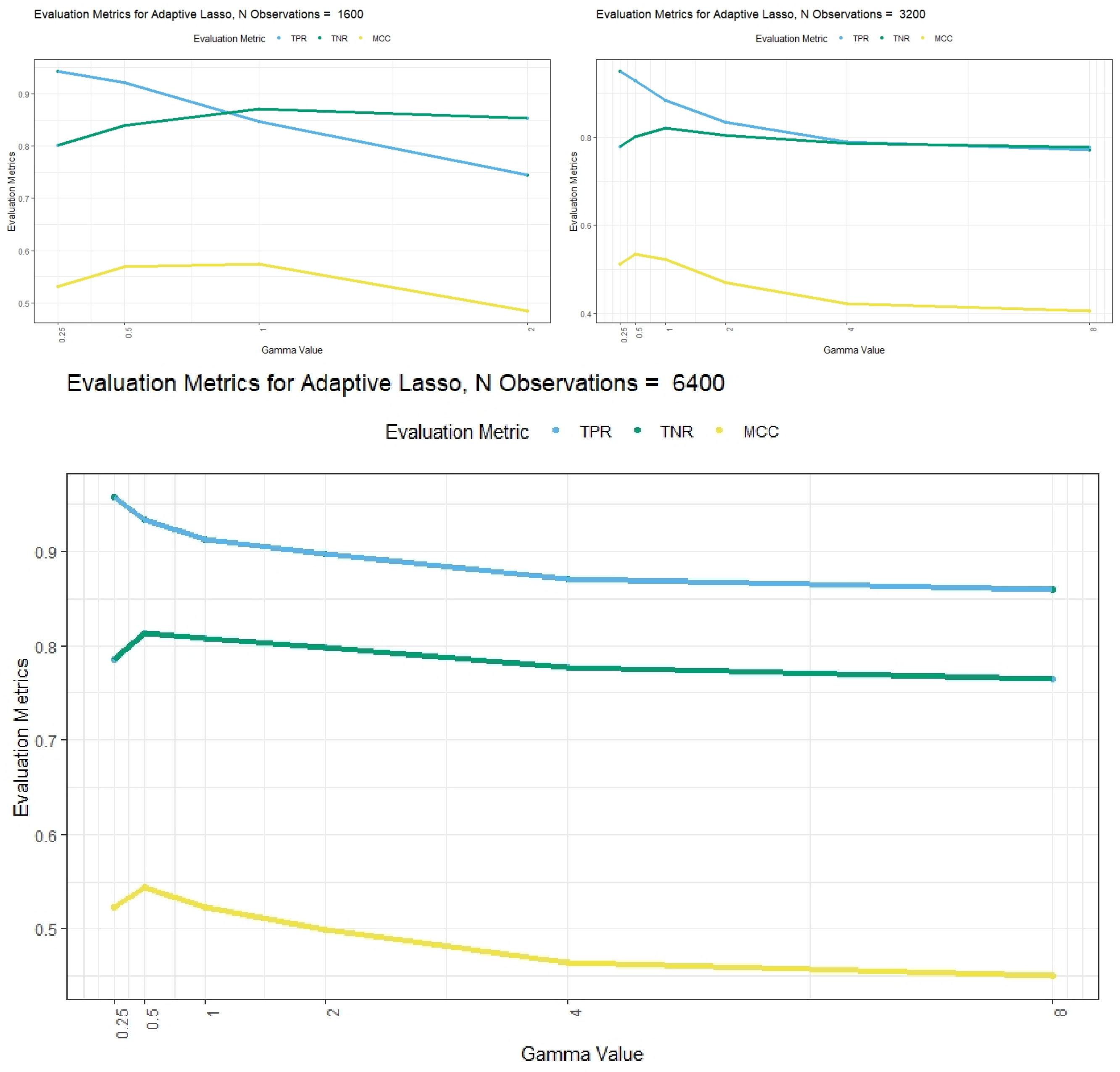


Figure 17: *Performance of adaptive lasso with varying γ and number of observations. The plot on the top left represents evaluation metrics taken by average over all runs and all sets of data with 1600 observations. Similarly other plots are for N = 3200,6400 respectively arranged in row-major matrix fashion.*

such a case adaptive lasso with larger γ i.e. $\gamma > 1$ works better than smaller γ i.e. $\gamma < 1$

- For very high N, here, $N > 1600$, prefer standard lasso over adaptive lasso for higher TNR. Also, in such a case adaptive lasso with smaller γ i.e. $\gamma < 1$ works better than larger γ i.e. $\gamma > 1$

Hence, it can be concluded that adaptive lasso works better than standard lasso for smaller values of N, whereas, for larger values of N, both works exceptionally good. However, adaptive lasso needs extra care with it's hyperparameter and is very sensitive to its setting. A smaller γ i.e. for $\gamma < 1$

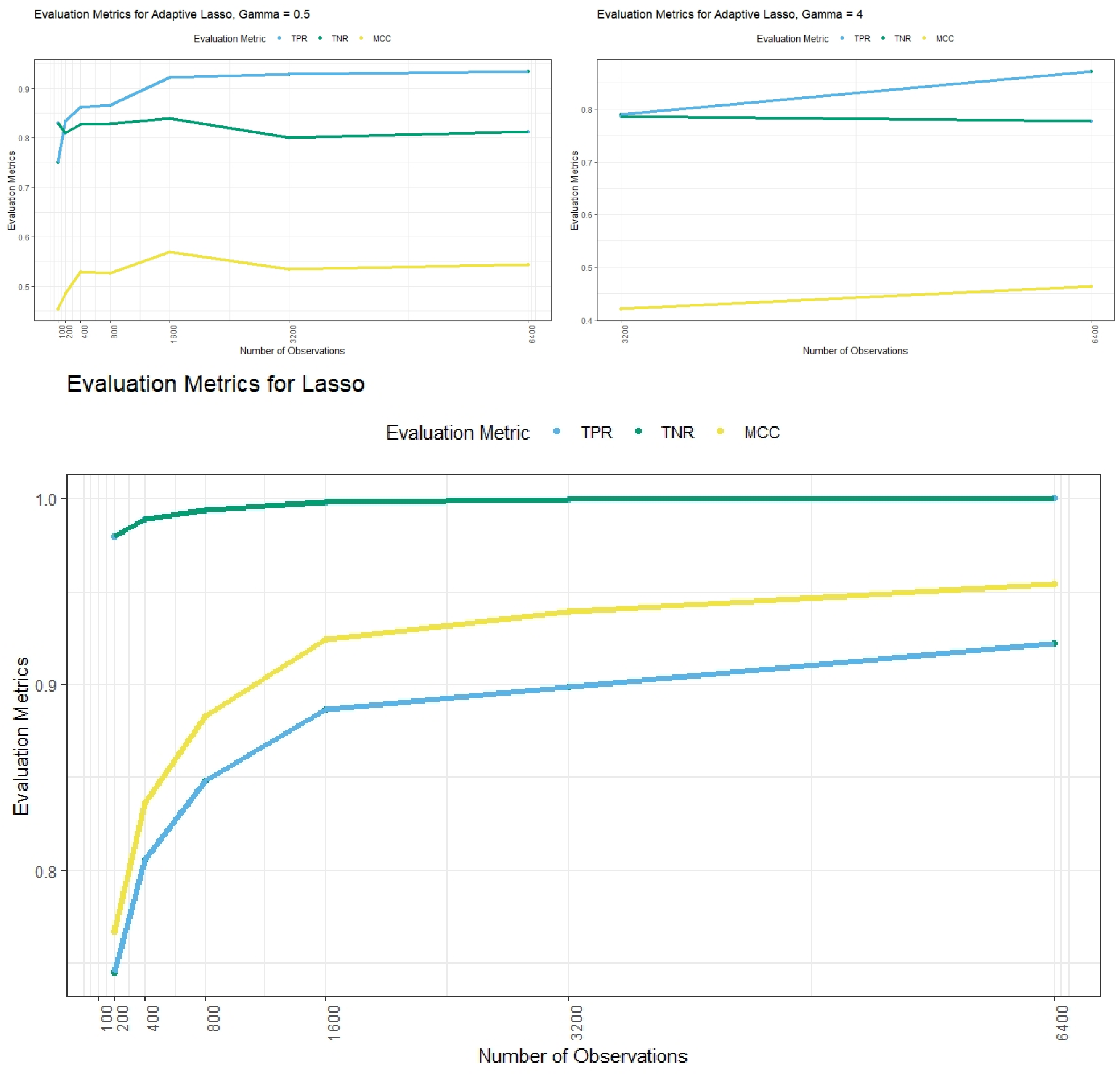


Figure 18: *Performance of adaptive lasso against standard lasso. The top row contains two plots of adaptive lasso for $\gamma = 0.5$ and $\gamma = 4$ respectively for all N. The bottom figure portrays performance of Lasso for all N.*

should be generally preferred if sample size is either too small or too large. The hyperparameter λ was always left for auto-tune by cross-validation and is generally observed to be in the range of $\lambda \in [0.001, 0.09]$ for both lasso and adaptive lasso. This range can be more precise by taking average overall runs.

Table 5: *True positive rate comparison between Ada Lasso and Lasso. Negative implies nonzero coefficients as nonzero. Green implies best performance among the three followed by orange then black.*

N	100	200	400	800	1600	3200	6400
AdaLasso $\gamma = 0.5$	0.75	0.83	0.86	0.86	0.92	0.92	0.93
Ada Lasso $\gamma = 4$	0.06	0.12	0.30	0.52	0.66	0.78	0.87
Lasso	0.61	0.74	0.80	0.84	0.88	0.89	0.92

Table 6: *True negative rate comparison between Ada Lasso and Lasso. Negative implies zero coefficients as zero. Green implies best performance among the three followed by orange then black.*

N	100	200	400	800	1600	3200	6400
AdaLasso $\gamma = 0.5$	0.82	0.80	0.82	0.82	0.83	0.80	0.81
Ada Lasso $\gamma = 4$	0.98	0.96	0.92	0.88	0.84	0.78	0.77
Lasso	0.96	0.97	0.98	0.99	0.99	0.99	0.99