LONDON SCHOOL OF ECONOMICS AND POLITICAL SCIENCE

ST443 PROJECT

Car Accident Severity Analysis and Graphical Model Simulation Study

---

Candidate Number:

32456

35757

28722

36805

Everyone contributed 25% to the project.

Date: 9th December 2021

Part I: Car Accident Severity Analysis

1 Introduction

Our project is based on the US countrywide car accident dataset, which contains 1.5 million accident records from February 2016 to December 2020. There are 47 variables describing the details of each accident, for example, Severity, Start Time, Start Latitude, Start Longitude, State, County, Temperature, Humidity, Visibility, etc. Further detail on the data dictionary can be found on Kaggle (`https://www.kaggle.com/sobhanmoosavi/us-accidents`).

The dataset classifies the degree of the severity of all the accidents into 4 categories. The larger the number means the more severe the accident. We are interested in finding out which variables lead to a greater chance of there being a severe accident. We hope to use machine learning methods to find the relationship between the environmental conditions during the incident and the severity of the accidents.

2 Data wrangling

The original dataset contains some variables that have no relationship with forecasting the severity, such as the IDs and street names. It also contains some duplicated variables, such as start latitude and end latitude, which are the same value for most observations. We deleted these variables to make our programme more efficient. In addition, we omit observations with NA values.

3 Visualization

First of all, we plot the distribution of the four severities (Figure 1). The barchart shows that Severity 2 has many more occurrences than the other three categories. In fact, it accounts for 80.65% of all the accidents, and Severity 3 and 4 account for 10.30% and 7.04% respectively (Table 1).

Spatial analysis

We also plot the number of accidents in each state (Figure 2). California has the most accidents, which is almost 3 times as many as the second state Florida. The severity in each state is not too different to the countrywide severity distribution, except Illinois, which has more accidents within the 3^{rd} accident level than the 2^{nd}. Furthermore, we also draw a map with points in different colours (Figure 3). It shows the special pattern of the 4 different levels of severity across America. Although California has the most accidents, we can see that severe accidents mainly occur at the centre of the US, especially the Northeast.

Time analysis

The bar chart shows that there is an apparent day difference in the accidents (Figure 4). Accidents occur more often on weekdays than on weekends. It suggests that part of the accidents may due to commuting to work.
We also separate the data by day and night (Figure 5). There are less accidents that occur during the night.

Special condition

After that, we construct a violin plot to visualize the distribution of humidity at each severity level (Figure 6). We consider that humidity might be one of the serious reasons that trigger the traffic accidents. Figure 5 clearly shows that when the humidity level increases, the density of accidents in the 2^{nd}, 3^{rd} and 4^{th} severity level also increases accordingly. However, there is no apparent relationship between humidity and the accident count in level 1. This result confirms our speculation and provides direction for our further analysis.

4 Modeling

Balancing the data

After cleaning the data, Severity 1 was no longer in the dataset. In addition, Severity 2 had over double the number of observations in 3 and 4 put together. Therefore, for simplicity, we combined Severity 3 and 4 into one group, making it a binary classification. Severity 2 was set to indicate a "light" accident, whilst the combined classes of 3 and 4 represented "serious" incidents. The classes were still highly imbalanced and if not changed, would have been an issue because most machine learning algorithms used for classification have been designed with the assumption of balanced classes [Brownlee, 2020]. Therefore, the Synthetic Minority Over-sampling Technique (SMOTE) was used to balance the dataset. The function `smote()` in R can be used to oversample the minority class or undersample the majority. Both methods were tested on the Logistic regression algorithm and the AUCs were compared. The dataset oversampled gave an AUC of 0.53 whilst the undersampled gave an AUC of 0.56. Therefore, the undersampled dataset was used. The accuracy of the imbalanced dataset was calculated to be very high, at 0.73. However, accuracy is generally not a good metric for unbalanced data, since it will predict the majority class for most samples in the test set, which will, on average, reflect the distribution of the test set. This issue is known as the accuracy paradox [Brownlee, 2021].

Model choice

Four machine learning algorithms were used on the dataset, namely: Logistic Regression, Random Forest, XGBoost and Support Vector Machines (SVM). Logistic regression was chosen because we wanted to evaluate the sample performance on a regression approach. Since our dataset was a binary classification, logistic regression was the most suitable option. In addition, Random forest was chosen because it has the feature importance attribute, which is very helpful for interpretability. Similarly, using XGBoost (a more complex algorithm), we can get a rank of the variable importance. We were interested in comparing the results from these two decision-tree algorithms. Being keen on seeing whether we could obtain improved results, we decided to also use SVM because it is another complex algorithm. Before properly diving into using these approaches however, the dimensions of the dataset needed to first be reduced to help prevent over fitting.

Dimension reduction

Principle Component Analysis (PCA) is used to reduce the dimensions of continuous multivariate data [Kassambara and Mundt., nd]. However, since many of our variables are binary (e.g. the location of accident), PCA was not a suitable choice of method. It was later found that Factor Analysis of Mixed Data (FAMD) is an alternative method to PCA and can be used on datasets containing both continuous and categorical variables. Had there been more time, this dimension reduction method would have also been explored and compared to our current results.

Feature selection

The Group LASSO was also explored. However, there were not many sources online that gave information on the method. Therefore, we decided to use Random Forest for feature selection. The method works by ranking the importance of each feature [Kumar, 2020].Figure 7 shows the results we obtained. The plot on the left shows the Mean Decrease Accuracy, which illustrates how much the accuracy of the model will decrease if a particular variable is removed. The figure on the right presents the Mean Decrease Gini, which measures the likelihood of a specific feature that is misclassified when selected randomly [Tyagi, 2020]. In both plots we can see that Temperature, Humidity and Pressure have the highest impact on model if they are either dropped or misclassified. SHapley Additive exPlanations (SHAP) values is another method used to interpret results from tree-based models. Similar to Random Forest importance features, SHAP plots rank the variables from the most important to the least. Figure 8 below shows a summary plot of the results obtained for the XGBoost model. From the plot we see that high humidity and locations

by a station, amenity or stop area is linked to a more severe accident. The other variables do not show a clear distinction on how their high/low values affect the prediction of accidents. This is not surprising because the AUC scores are not high. Both the Random Forest importance feature and SHAP ranked Temperature, Humidity and Pressure as the top key variables. However, the other features were ranked in different orders. To compare the feature selection methods, the top five variables of each were fitted on the Logistic Regression. Both the results were very similar with SHAP giving an AUC score of 0.56 and Random Forest a value of 0.55. Since the SHAP results were slightly better, we decided to use the corresponding top five features for logistic regression and SVM. The ROC curves for each of the machine learning approaches were generated and final AUC values were computed and are presented in Figure 9 and Table 2 respectively.

Model evaluation

Overall Random Forest had the greatest AUC score with a value of 0.62, followed by XGBoost (0.61), SVM (0.57) and finally Logistic Regression (0.56). The results are not very good because of the dataset, with Logistic Regression and SVM being slightly better than random. Although Random Forests only marginally gave the best results, this method was chosen because it is not as complex as XGBoost. This makes it easier to interpret. Furthermore, the fact that we can extract the most important features is very useful. Random Forest is also an easy algorithm to use and does not require much hyperparameter tuning to get good results. Additionally, the algorithm is not prone to overfitting if there are enough trees in the model. This is a key advantage since more complex models such as XGBoost, have this problem if the parameters are not tuned effectively (e.g. max_depth) [Donges, 2021].

5 Conclusion

In conclusion, Random Forest had the highest AUC score of 0.62, which was very close to that of XGBoost (value of 0.61). The Random Forest algorithm revealed the most important features which was useful in knowing which had the greatest impact on Severity. Temperature, Humidity and Pressure were the top three variables that was shown by both algorithms to have the greatest impact. Had there been more time, other algorithms including K-nearest neighbours, would would have used. In addition, the FAMD dimension reduction method would have been explored, to see whether improved results could be obtained.

Appendix A Tables and Figures

1	2	3	4
0.02010221	0.80654275	0.10298855	0.07036648

Table 1: Percentage of different severity

Model	Logistic Regression	Random Forest	XGBoost	SVM
AUC	0.56	0.62	0.61	0.57

Table 2: AUC Score for each model

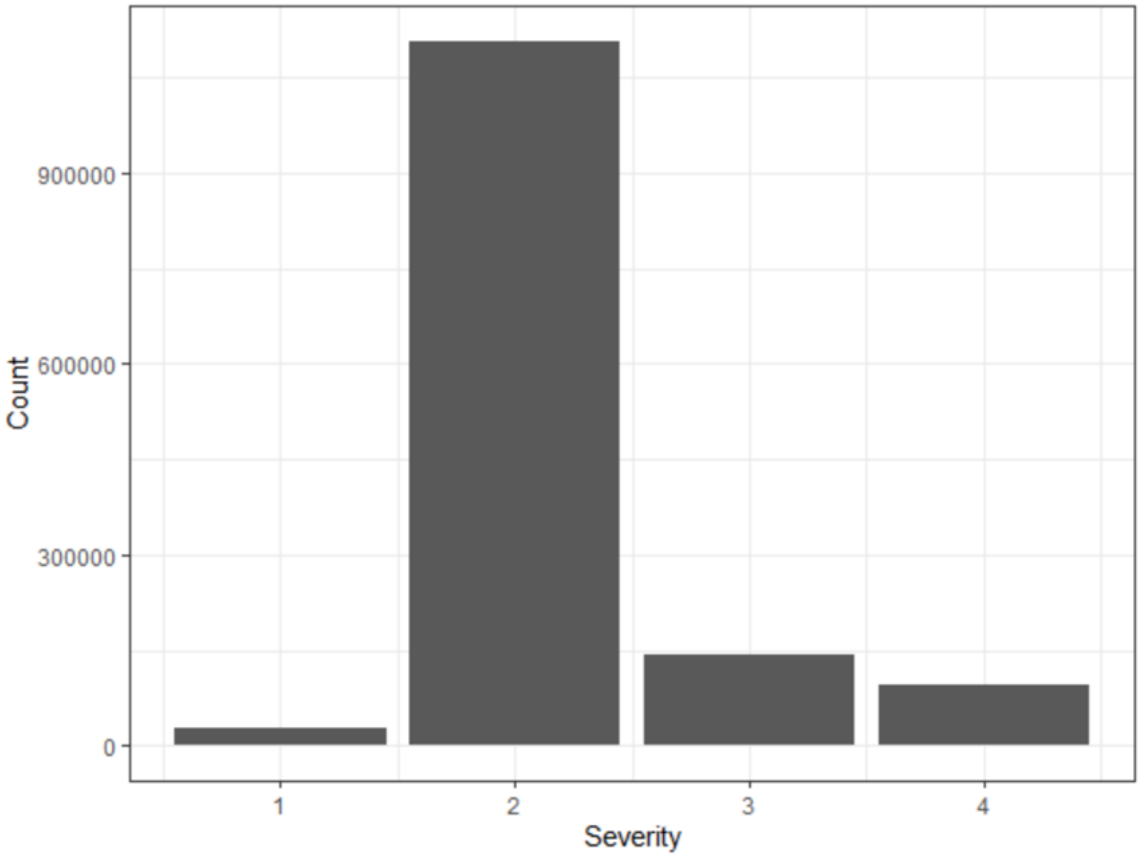


Figure 1: the accident count in different levels of Severity

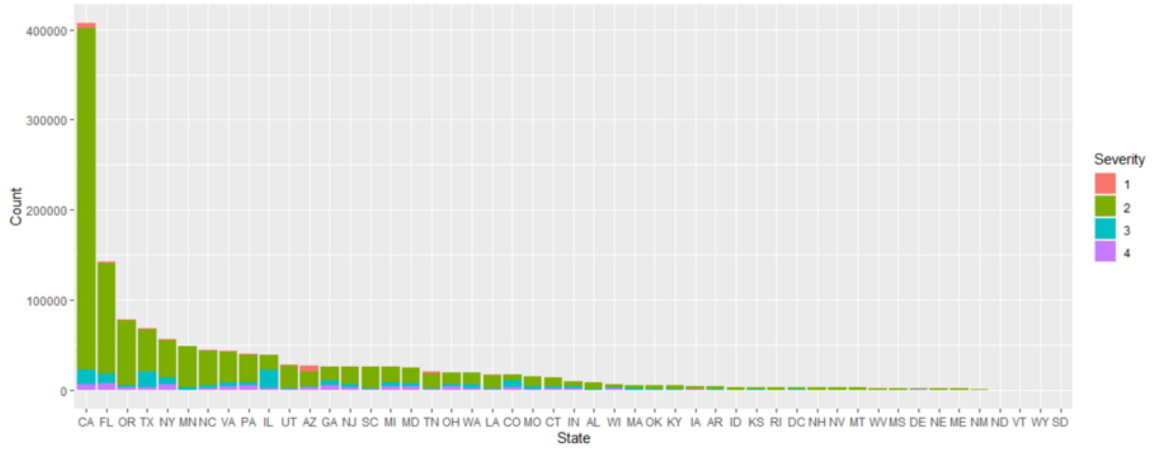


Figure 2: Accident count in different states

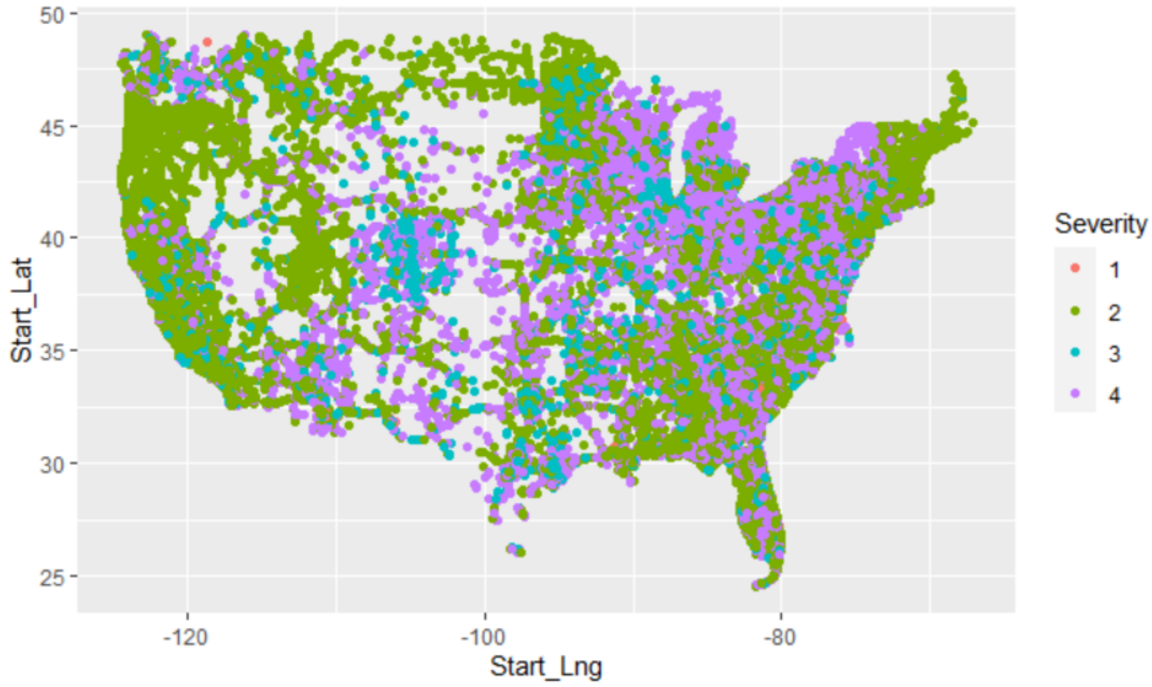


Figure 3: Map with severity levels

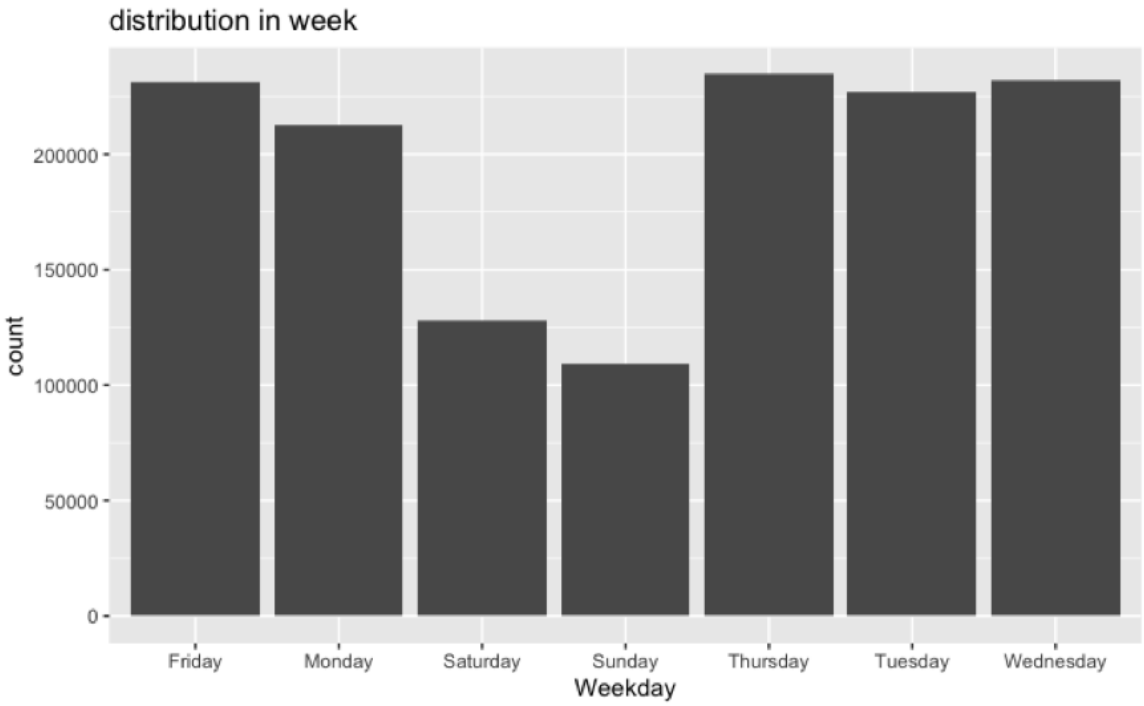


Figure 4: Accidents in weekdays

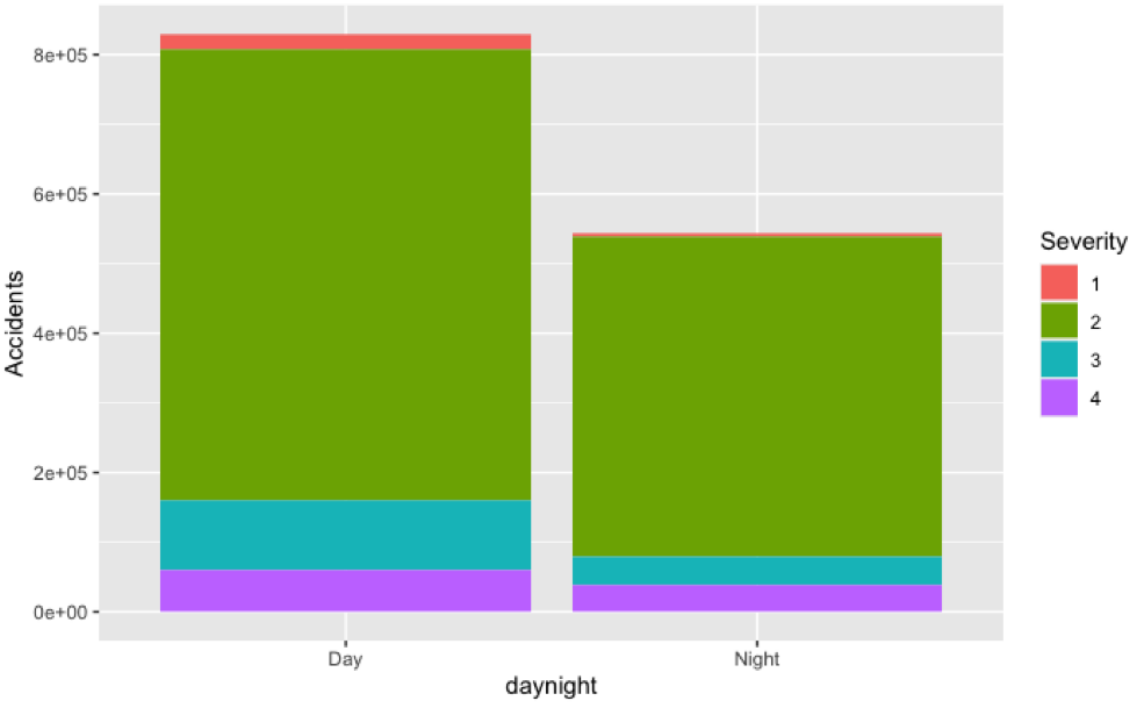


Figure 5: Time difference of the accidents

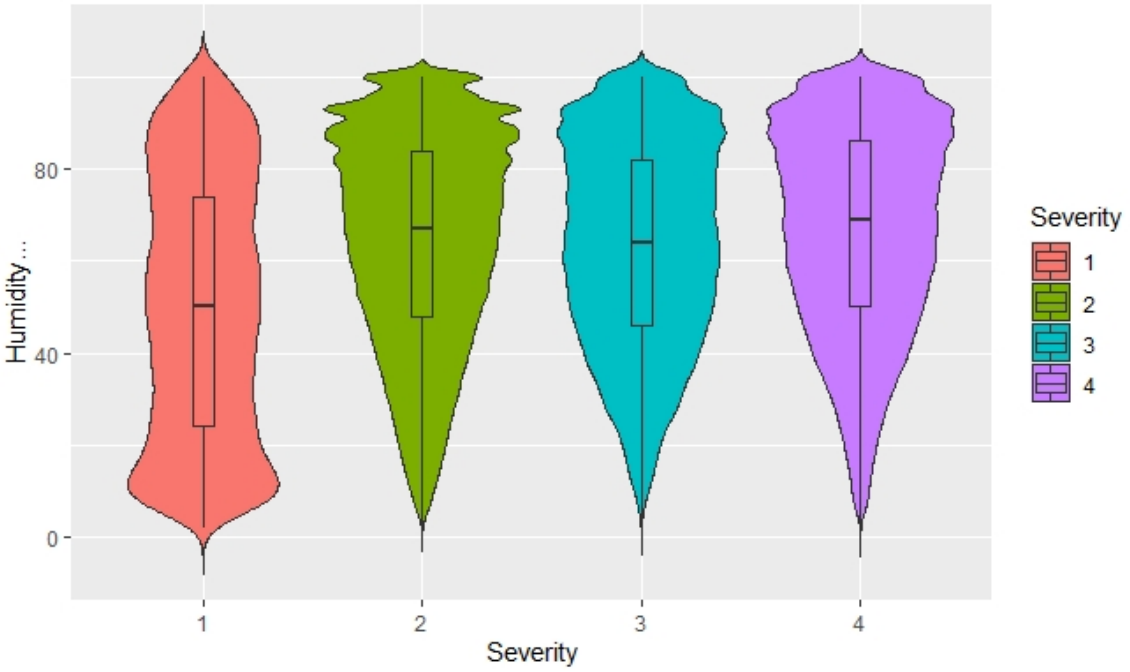


Figure 6: The distribution of Humidity in different levels of Severity

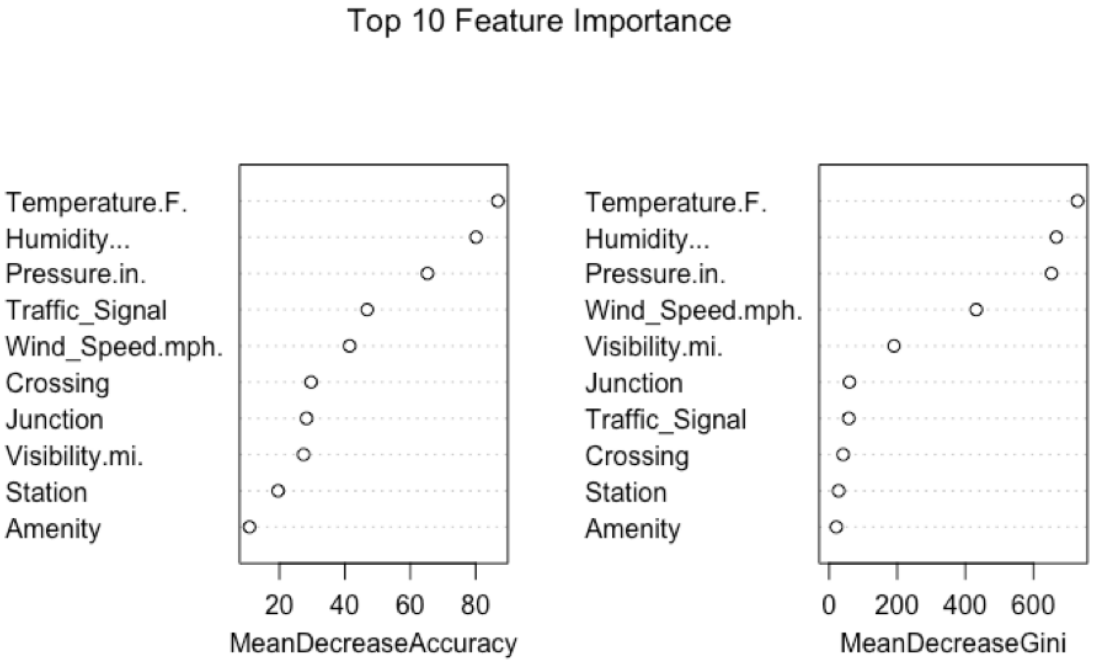


Figure 7: Top 10 importance features from Random Forest

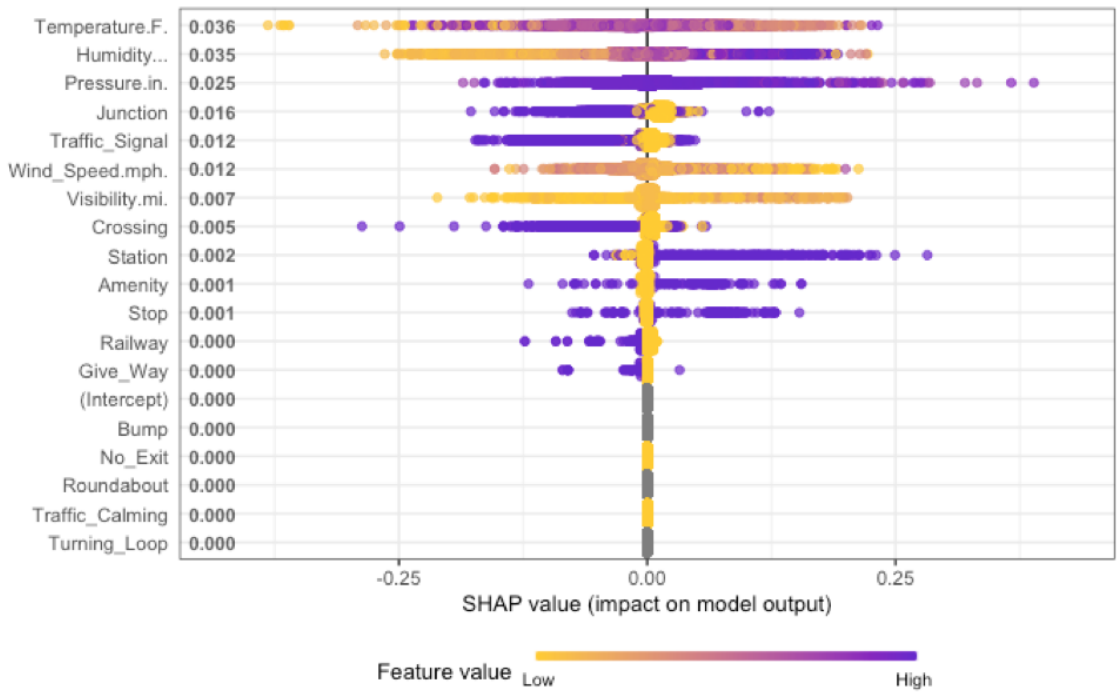


Figure 8: SHAP plot: Each dot in the plot represents an observation in the dataset. The SHAP value is on the x-axis and indicates how much the change is in log-odds. The variable name is on the y-axis. The numbers next to each variable name is the SHAP value. The gradient colour represents the value for that variable (e,g. high temperature is shown as purple whilst a low temperature is yellow).

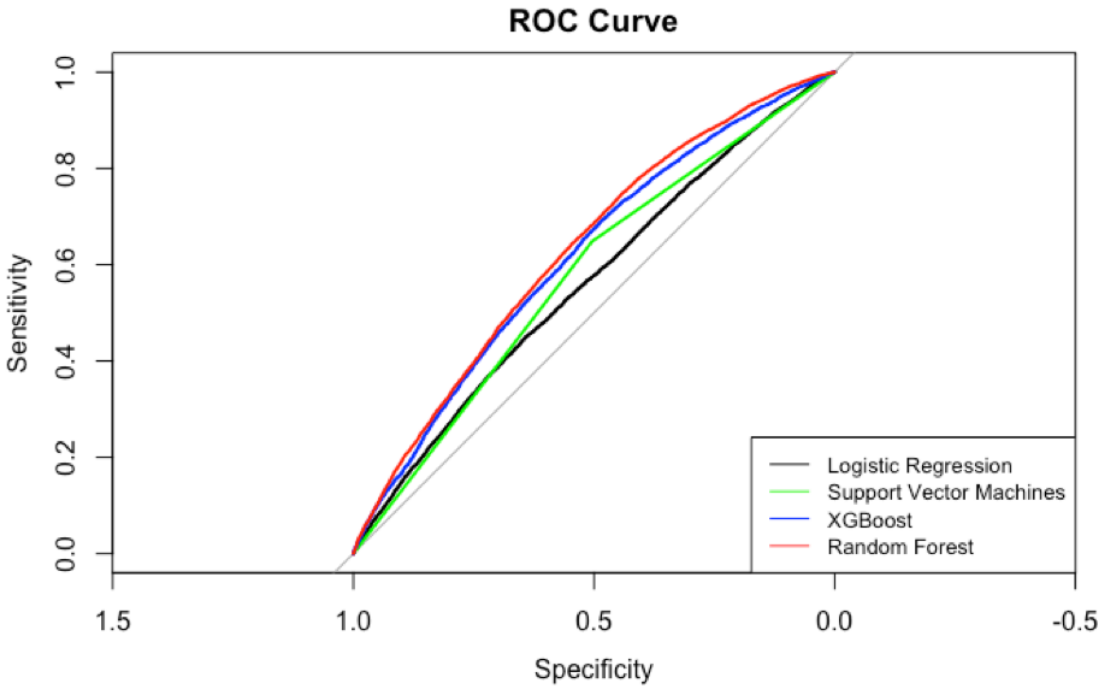


Figure 9: ROC curve for logistic regression, support vector machines, XGBoost and Random Forest

References

Brownlee, J. (2020). A Gentle Introduction to Imbalanced Classification.

Brownlee, J. (2021). Failure of Classification Accuracy for Imbalanced Class Distributions.

Donges, N. (2021). A Complete Guide to the Random Forest Algorithm.

Kassambara, A. and Mundt., F. (n.d.). factoextra : Extract and Visualize the Results of Multivariate Data Analyses.

Kumar, A. (2020). Feature Importance using Random Forest Classifier – Python.

Tyagi, N. (2020). Understanding the Gini Index and Information Gain in Decision Trees.

Part II: Graphical Model Simulation Study

Contents

Chapter 1

Introduction

In this chapter, we are going to introduce how node-wise lasso approach and graphical lasso approach are used to recover edges of graphical models. Let G = (V, E), where V denote the set of nodes, and E denote the edge set between the nodes. We would like to recover E as much as possible using these approaches.

1.1 The lasso

Consider a linear regression with predictors x_{ij} and responses y_i, where $i = 1,2,\ldots,n$ and $j = 1,2,\ldots,p$, the lasso uses a l_1 norm of penalty to solve the regression problem by finding the coefficient $\hat{\beta}_\lambda^L$ that minimises

$$\sum_{i=1}^{n}\left(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij} + \lambda\sum_{j=1}^{p}|\beta_j|\right) = RSS + Penalty, \tag{1.1}$$

where $\lambda \geq 0$ is the regularisation parameter [Xinghao, 2021].

Because of the l_1 penalty, using a large λ in lasso will force some of the coefficients to be zero, which means lasso can do both shrinkage and variable selection [Tibshirani, 2011]. As a result, selecting the tuning parameter λ is an important step in the lasso.

1.2 Node-wise lasso

Consider a graphical model having p variables, and $\mathbf{X} = (X_1,\ldots,X_p)^T$ is multivariate normal. Node-wise lasso approach, as its name suggests, is a penalised regression of each variable X_j on the

remaining variables, for each node $j = \{1, \ldots, p\}$. It can be written as

$$X_j = \sum_{1 \le l \le p, l \ne j} \beta_{jl} X_l + \lambda \sum_{1 \le l \le p, l \ne j} |\beta_{jl}| + \epsilon_{jl}. \tag{1.2}$$

Under a certain choice of tuning parameter λ, one can investigate the lasso estimates, $\hat{\beta}_{jl}$. If $\hat{\beta}_{jl} \neq 0$, node j and node l are estimated to be connected. Based on the 'joint' and 'or' rules, there are two ways to estimate the edge E:

node-wise lasso 1: $\hat{E}_1 = \{(j,l) : \text{both } \beta_{jl} \text{ and } \beta_{lj} \text{ are non zero, for } 1 \le j, l \le p, j \ne l\}$,

node-wise lasso 2: $\hat{E}_1 = \{(j,l) : \text{either } \beta_{jl} \text{ or } \beta_{lj} \text{ is non zero, for } 1 \le j, l \le p, j \ne l\}$.

In the following simulation experiments, we will examine both E_1 and E_2 estimates to see which one performs better in edge recovery.

1.3 Graphical lasso

The graphical lasso is a more popular method to estimate graphical models compared to the node-wise lasso [Qiao et al., 2019]. This method is proposed by Yuan and Lin [2007]. The graphical lasso applies the l_1 penalty on the off-diagonal elements of the precision matrix, which is similar to the idea of the standard lasso. It is a penalised-likelihood method which performs variable selection and shrinkage estimation simultaneously for the Gaussian graphical model.

Consider $\mathbf{X}$ is a multivariate normal with mean 0, covariance matrix $\mathbf{\Sigma} \in \mathbb{R}^{p \times p}$. Let $\mathbf{\Theta} = \mathbf{\Sigma}^{-1}$, according to the *project instructions*, an edge can be proved to be non-existent if and only if Θ_{jl} is 0, where Θ_{jl} is the (j,l)-th entry of $\mathbf{\Theta}$. The sparse estimate $\hat{\mathbf{\Theta}}$, is determined based on minimising the sum between the negative Gaussian log-likelihood and a penalty on the off-diagonal entries of the precision matrix, which can be written as

$$-logdet\mathbf{\Theta} + trace(\mathbf{S\Theta}) + \lambda \sum_{j \ne l} |\theta_{jl}|, \tag{1.3}$$

where $\mathbf{S} = \frac{1}{n-1} \sum_{i=1}^{n} (x_i - \bar{x})(x_i - \bar{x}^T)$ for $\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i$, and λ is again the tuning parameter. Under a certain choice of λ, the estimated edge set is

$$\hat{E}_3 = \{(j,l) : \mathbf{\Theta}_{jl} \neq 0, 1 \le (j,l) \le p, j \ne l\}. \tag{1.4}$$

Chapter 2

Simulation Overview

2.1 Simulation settings

To conduct the simulation experiment, n random samples are generated from a multivariate normal distribution with mean 0 and covariance $\Sigma \in \mathbb{R}^{p\times p}$ using the `mvrnorm` function in R. Values of n and p will be varied in our simulation experiment. The covariance matrix Σ is generated by inverting a matrix Θ, where Θ is a sparse matrix that determines the true edge set of the graphical model.

To generate matrix Θ, we set $\Theta = \mathbf{B} + \delta \mathbf{I}_p$, where $\mathbf{I}_p$ is a $p \times p$ identity matrix, and $\mathbf{B}$ is a symmetric matrix with all the diagonal entries equal to zero, and off-diagonal entries having a probability of 0.1 to be 0.5 and a probability of 0.9 to be 0, making Θ to have a sparsity of 90%. Again, the sparsity will be varied later in the study for comparing the edge recovering performance of different models.

Specifically to generate $\mathbf{B}$, we first generate $(p^2 - p)/2$ random probabilities (this correspond to the number of unique off-diagonal entries in a symmetrical matrix) between 0 and 1 using `runif` function in R, then regard values less than or equal to 0.1 as "0.5", and values greater than 0.1 as "0". After that, the $p \times p$ matrix $\mathbf{B}$ is filled by these updated values. We then consider the value of δ. Since δ should be a positive number making Θ positive definite, and knowing that a positive definite matrix has all the eigenvalues greater than 0, we choose $\delta = -min\{k_1, k_2, \ldots, k_p\} + 0.01$, where $k_1, k_2, \ldots, k_p$ are eigenvalues of $\mathbf{B}$. This approach follows directly from this equation:

$$trace(B) = \sum_{i=1}^{n} k_i \tag{2.1}$$

Finally, Θ is standardised to have unit diagonals using the `cov2cor` function, and the covariance matrix Σ is simply Θ^{-1}.

2.2 Selection of optimal tuning parameters

There are many methods choosing the optimal lambda. The approach we are using involves finding the constant which will be multiplied with the asymptotic convergence rate of LASSO models through grid search and cross validation.
The asymptotic convergence rate is

$$s\sqrt{\frac{\log p}{n}} \tag{2.2}$$

where s is a positive real number, p is the number of covariates, and n is the sample size. [Belloni and Chernozhukov, 2011; Chetverikov et al., 2021]

2.2.1 For node-wise lasso approach

We perform 5-fold cross validation on the simulated data. Each lambda from the predefined grid is used and we calculate the MSE (Mean Squared Error) for each model specified in 1.2. We then take the average across p models and make this our cross validation error. Next, we proceed to choose the lambda which produces the lowest MSE.
After collecting a list of optimal lambdas, we divide all of them by the theoretical convergence rate to get the constants s in 2.2. This produce a distribution of s, and we take the median of s. In later simulations we will have a refined grid which centers around this theoretical lambda, and spans for 95 percent confidence interval.

2.2.2 For graphical lasso approach

Similar to the Node-wise Lasso approach, we perform 5-fold cross validation on the simulated data. Using each lambda from the predefined grid, we fit the training data into GLASSO and calculate the validation negative log-likelihood. We then proceed to choose the lambda which creates the minimum negative log likelihood to be the optimal lambda. [Lee et al., 2016]
Collecting these optimal lambdas 15 times, then dividing them by the theoretical convergence rate to get a list of constants. From this distribution of constants, we take the median value to be the optimal constant, which multiplies by the theoretical convergence rate to get back the optimal lambda. Using this value, we refine our grid of lambda in later iterations to speed up the computation time.

Chapter 3

Simulation Result Evaluation

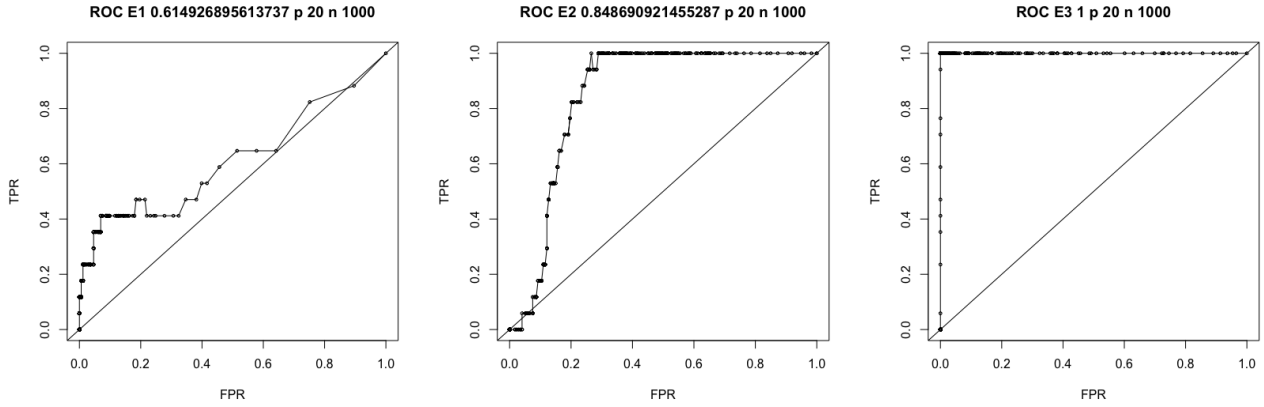
3.1 ROC

We will use a grid of lambdas to directly make prediction using each models for the entire simulated data, then calculate the True Positive Rate and False Positive Rate:

$$TPR_\lambda = \frac{TP_\lambda}{TP_\lambda + FN_\lambda} \tag{3.1}$$

$$FPR_\lambda = \frac{FP_\lambda}{FP_\lambda + TN_\lambda} \tag{3.2}$$

Plotting this for a fine grid of lambdas gives:



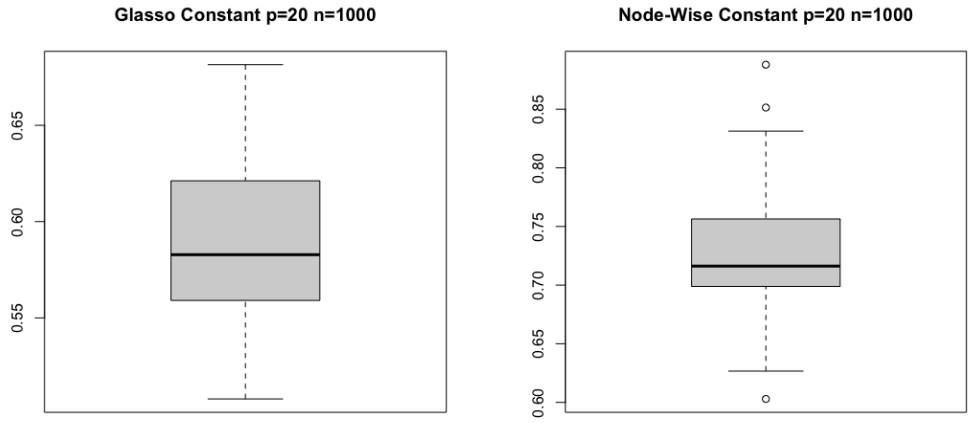
Keep in mind that the use of the true edge set in choosing the best lambda will not be allowed, as not only this leads to overfitting, but also in a realistic scenario this will almost always never be known. Having seen the ROC for each model, we can quickly get a sense of how each model will perform against this specific randomly generated sparse matrix.

From the ROC, we can see that E3 should be outperforming other models with AUC of 1, while Node-Wise 2's AUC being around 0.85, and Node-Wise 1's AUC being around 0.61. Although the

AUC has significant differences, since we cannot directly use this information, it will still be uncertain whether the final model will perform in a similar pattern. Hence, further studies are carried out.

3.2 Constant Analysis

As mentioned in Equation 2.2, we construct a boxplot for the list of constants using a fine grid of lambdas.



The magnitude of both constants for GLASSO and Node-Wise models are similar, with the median of GLASSO's constant being 0.5828264, and Node-Wise's constant being 0.7162004.
We take the median of the constant and multiply the theoretical convergence rate in Equation 2.1 to get back the theoretical lambda, which will be used as the of the refined grid of lambdas in the following 50 iterations. The width of the grid is set to be 1.96 times the standard error of the constants to capture the theoretical optimal lambda with 95 percent confidence.

3.3 Support Recovery

Table 3.1: Median Model Performances

	TPR	**FPR**	**Precision**	**MR**
Node-Wise 1	0.3831522	0.1339404	0.2915028	0.2
Node-Wise 2	1	0.5846209	0.2138021	0.505632
GLASSO	1	0.4236818	0.2731158	0.3578947

TPR: True Positive Rate, FPR: False Positive Rate, MR: Misclassification Rate

In this particular setting of $n = 1000$, $p = 20$, and sparsity ratio of 0.1, this table shows the end

result after 50 iterations. Based on Misclassfication Rate, Node-Wise 1 produced the best result. However, its TPR is the lowest amongst all the models and the FPR is also the lowest. This suggests that Node-Wise 1 predicts the inactive edge more accurately, while incorrectly predicting the active edge most of the time. The reason for this is probably that Node-Wise 1 has a stricter condition on making a positive prediction, and given the sparse setting, most of the predictions will be negative. Node-Wise 2 and GLASSO models both predicted the active edges perfectly, but GLASSO predicts the inactive edges with higher accuracy. The misclassification also tells a similar story, with GLASSO (0.36) outperforming Node-Wise 2 (0.5).

$$Precision_\lambda = \frac{TP_\lambda}{FP_\lambda + TP_\lambda} \tag{3.3}$$

Precision measures how well a model predicts the active edges out of all positive predictions. Compared to TPR, it puts more weighting on making the right positive prediction from the result of replacing FN_λ with FP_λ.

Focusing on precision, Node-Wise 1 (0.29) unsurprisingly has the highest amongst all models, due to its strict condition to make positive prediction. GLASSO (0.27) has lower precision than Node-Wise 2 (0.58), meaning that out of the positive predictions, Node-Wise 2 made more correct predictions. Hence, in some scenario (e.g. Contagious Disease Study) where being precise on positive prediction is more important, Node-Wise 2 would be a better model than GLASSO. However, in the setting of needing to predict the inactive edges with high accuracy due to time or cost (e.g. Crime Analysis) or just overall accuracy, Node-Wise 1 would be a better model.

3.4 Varying n,p and sparsity

Using $p = 20$ and $n = 1000$ as the baseline, we consider changing n, p, and the sparsity structure individually while fixing other variables constant. Through experimenting with $p = \{5, 30, 40\}$, $n = \{30, 300, 1000\}$, and sparsity = {95%,90%,80%}, we found some really interesting results. In Table 3.2 we realise that Node-Wise 1 and GLASSO performs better in higher dimension, predicting negative more accurately, whereas Node-Wise 2's performance does not vary too much. In Table 3.3, it is very clear that LASSO based models performs well in sparser settings, but may have worse performance otherwise. In Table 3.4, lowering n generally leads to worse results. Overall, similar to our previous conclusion, Node-Wise 1 consistently outperforms other two models, and GLASSO models outperforms Node-Wise 2 models is most settings. This is consistent with the discussion found in [Epskamp, 2018].

Table 3.2: Median Model Performances holding $n = 1000$ and sparsity=0.9

	p	**TPR**	**FPR**	**Precision**	**MR**
Node-Wise 1	5	0.9132582	0.4188572	0.5215121	0.3562831
	30	0.2	0.0629471	0.3829517	0.15
	40	0.3041739	0.08494202	0.3038462	0.1429487
Node-Wise 2	5	1	0.7078427	0.3992487	0.505478
	30	1	0.5846209	0.2138021	0.505632
	40	0.9447262	0.46179584	0.1973231	0.4141026
GLASSO	5	1	0.8472391	0.3829471	0.6
	30	1	0.4236818	0.2731158	0.3578947
	40	1	0.27784569	0.2814495	0.2583333

Table 3.3: Median Model Performances holding $n = 1000$, $p = 20$

	Sparsity	**TPR**	**FPR**	**Precision**	**MR**
Node-Wise 1	0.8	0.7562739	0.5243718	0.1521211	0.5
	0.95	0.3448732	0.08251601	0.2970588	0.1394737
Node-Wise 2	0.8	1	0.92	0.1246213	0.8124833
	0.95	1	0.45558292	0.1862812	0.4026316
GLASSO	0.8	1	0.6823518	0.1581192	0.5892951
	0.95	1	0.30397074	0.2642320	0.2736842

Table 3.4: Median Model Performances holding $p = 20$, and sparsity=0.9

	n	**TPR**	**FPR**	**Precision**	**MR**
Node-Wise 1	30	0.2152832	0.0621452	0.3157842	0.1532561
	300	0.3842667	0.1339404	0.3018152	0.191213
Node-Wise 2	30	0.7032451	0.3829184	0.2	0.3853812
	300	0.96549	0.5835318	0.1892989	0.435478
GLASSO	30	0.9548371	0.2511213	0.3182958	0.2387412
	300	1	0.36523	0.2542158	0.3267847

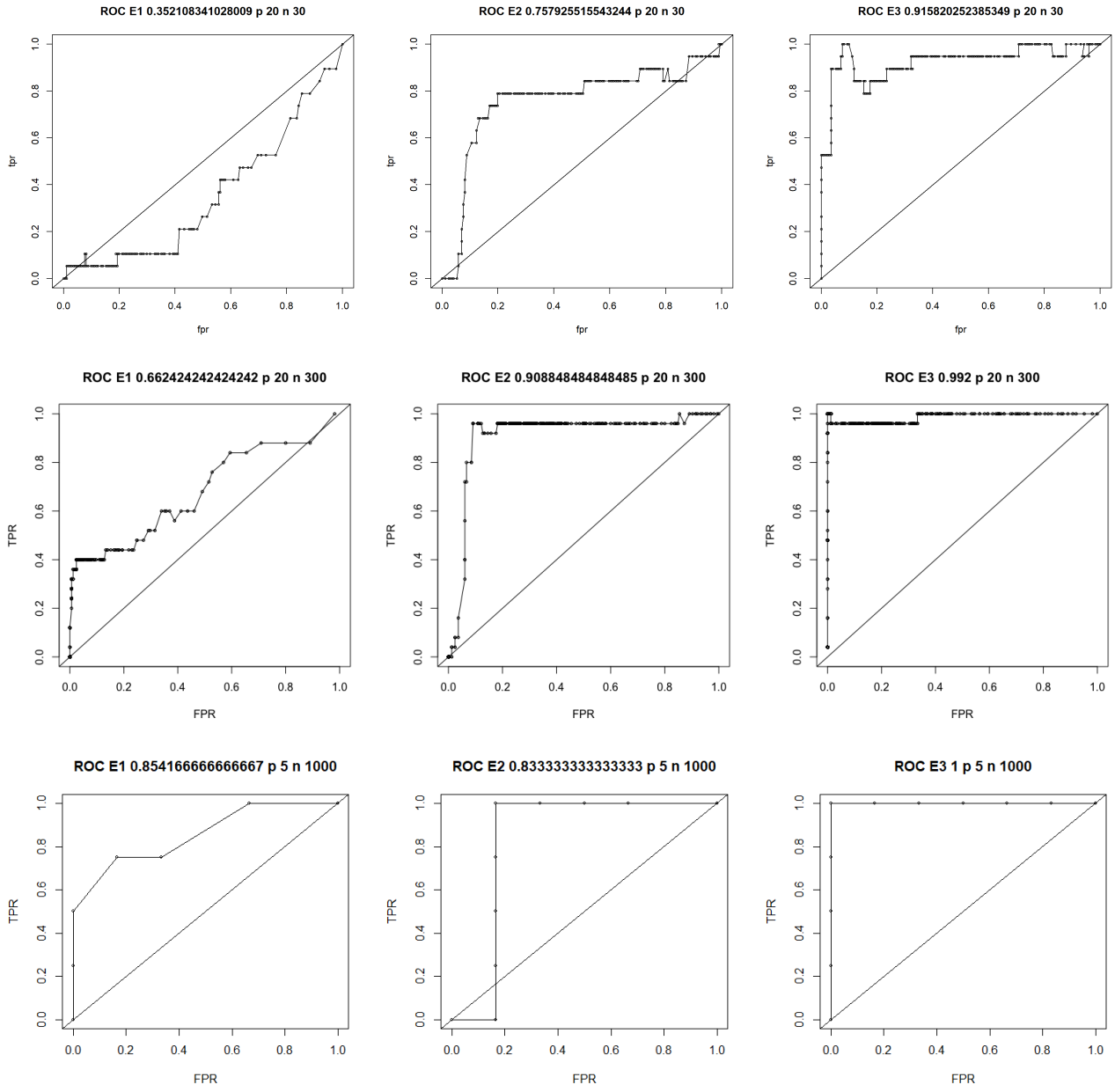
Chapter 4

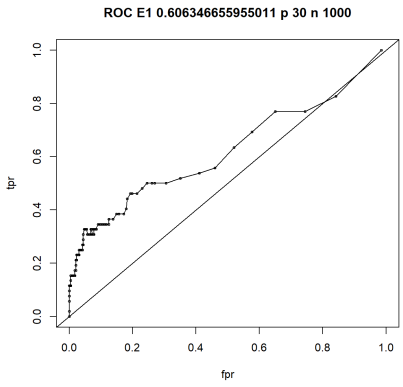
Conclusion

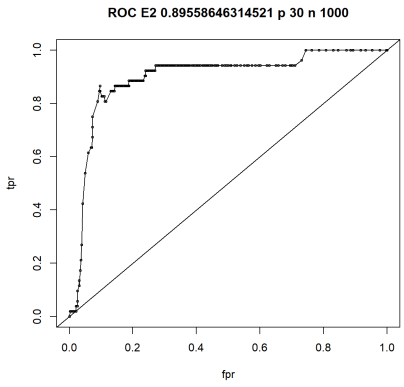
Node-Wise (Both Type 1 and Type 2) and Glasso models utilises LASSO's feature selection to predict whether an edge between two nodes are active. In our simulations, we use the baseline of $n = 1000$, $p = 20$, and sparsity structure of 90%. Then changing one of n, p,sparsity, while keeping the other constant, we conclude that Node-Wise 1 is the best model at predicting negative edges, and as we are working with sparse high-dimensional settings, the stricter "joint" rule for predicting positive edges rather than "or", makes it consistently outperforms other models. If predicting the negative edges are less of a focus, then GLASSO and Node-Wise 2 performs consistently well, with near 1 TPR in all our simulation settings.

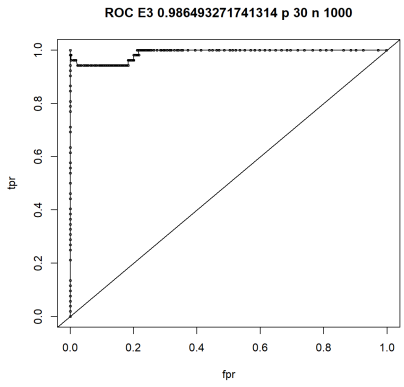
Appendix A

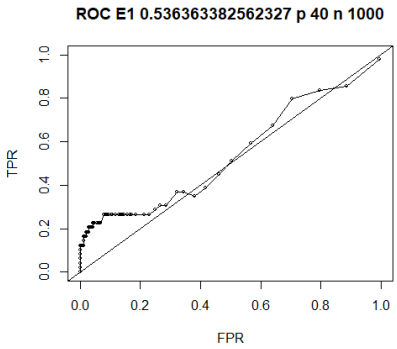
Figures

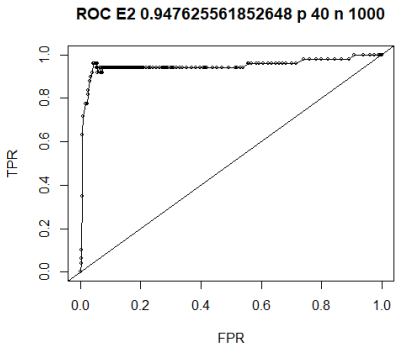


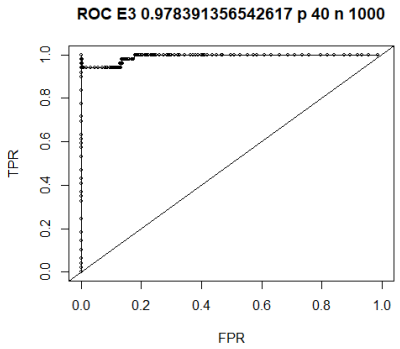

ROC E1 0.606346655955011 p 30 n 1000
tpr
fpr

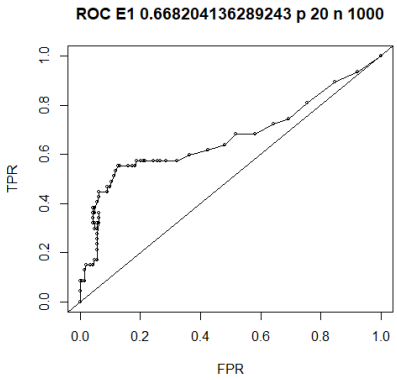

ROC E2 0.89558646314521 p 30 n 1000
tpr
fpr

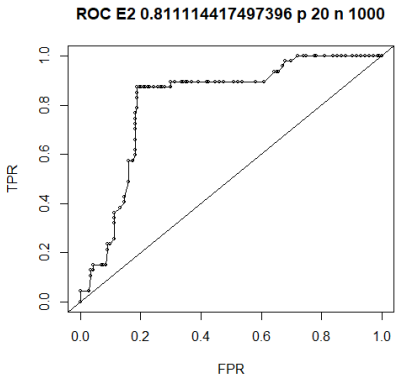

ROC E3 0.986493271741314 p 30 n 1000
tpr
fpr

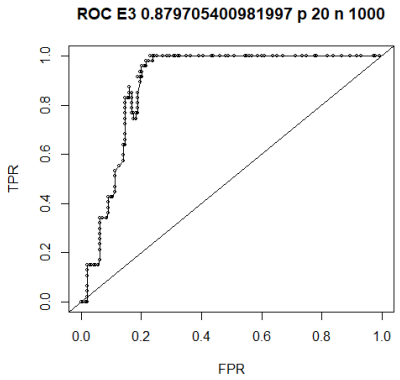

ROC E1 0.536363382562327 p 40 n 1000
TPR
FPR

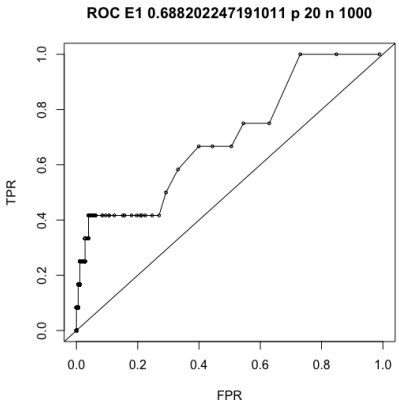

ROC E2 0.947625561852648 p 40 n 1000
TPR
FPR

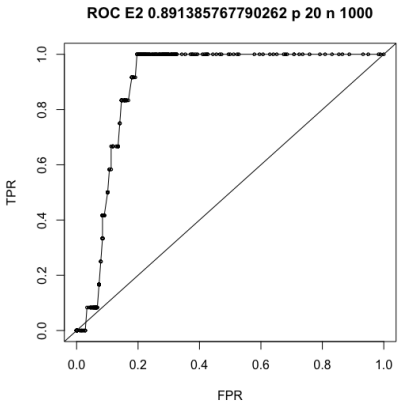

ROC E3 0.978391356542617 p 40 n 1000
TPR
FPR

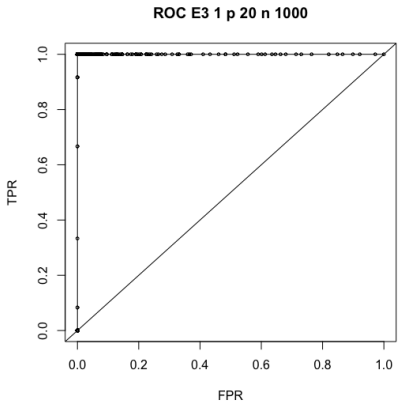

ROC E1 0.668204136289243 p 20 n 1000
TPR
FPR

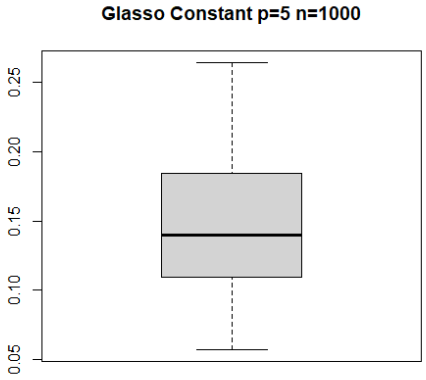

ROC E2 0.811114417497396 p 20 n 1000
TPR
FPR

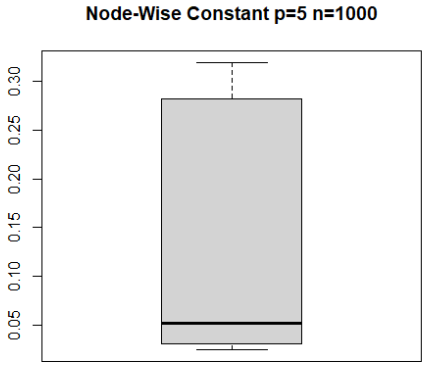

ROC E3 0.879705400981997 p 20 n 1000
TPR
FPR

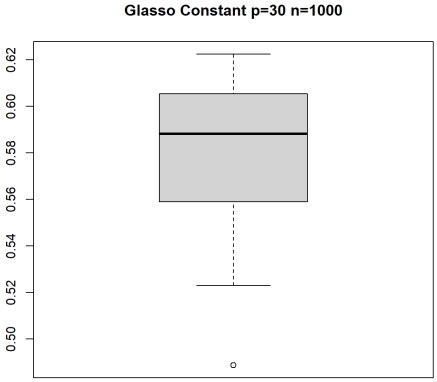

ROC E1 0.688202247191011 p 20 n 1000
TPR
FPR

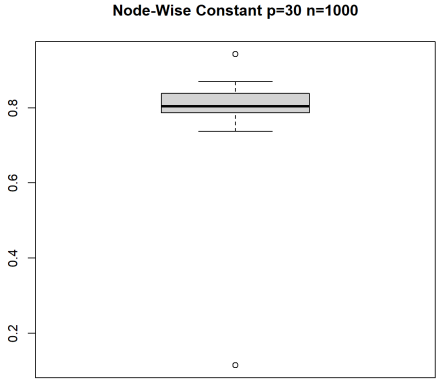

ROC E2 0.891385767790262 p 20 n 1000
TPR
FPR

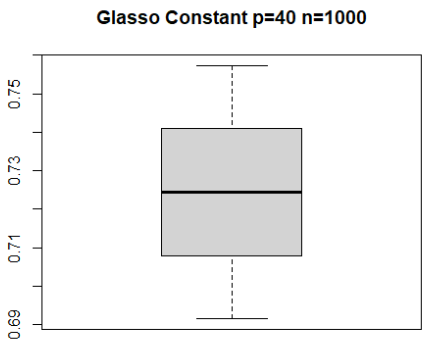

ROC E3 1 p 20 n 1000
TPR
FPR

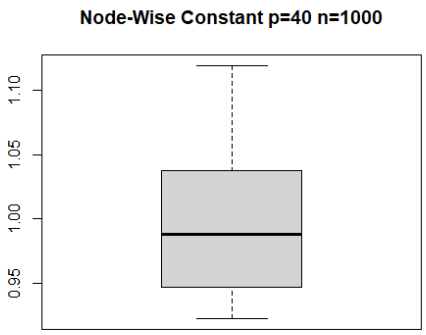

Glasso Constant p=5 n=1000
0.05
0.10
0.15
0.20
0.25

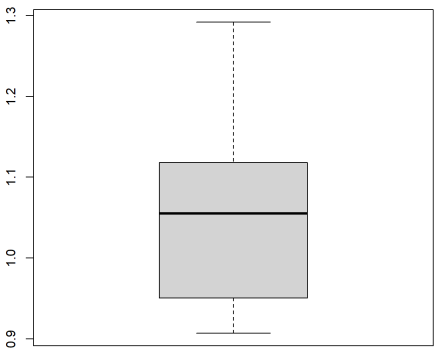

Node-Wise Constant p=5 n=1000
0.05
0.10
0.15
0.20
0.25
0.30

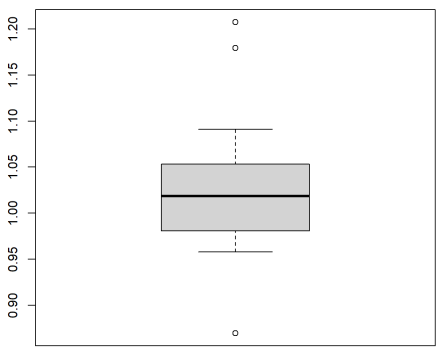

Glasso Constant p=30 n=1000
0.50
0.52
0.54
0.56
0.58
0.60
0.62

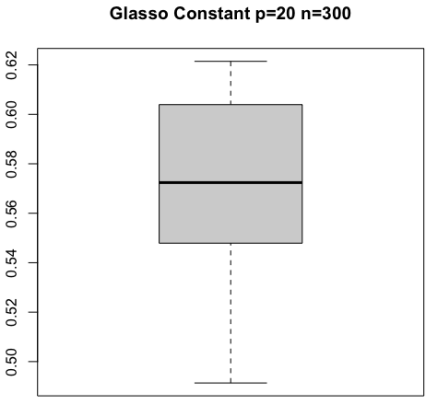

Node-Wise Constant p=30 n=1000
0.2
0.4
0.6
0.8

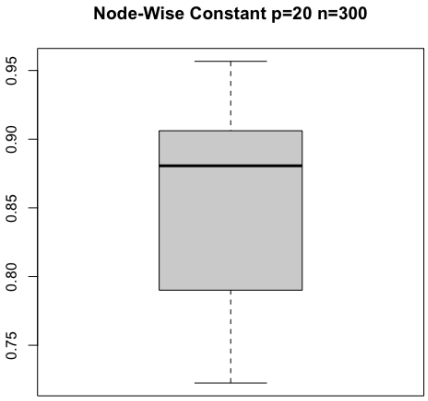

Glasso Constant p=40 n=1000
0.69
0.71
0.73
0.75

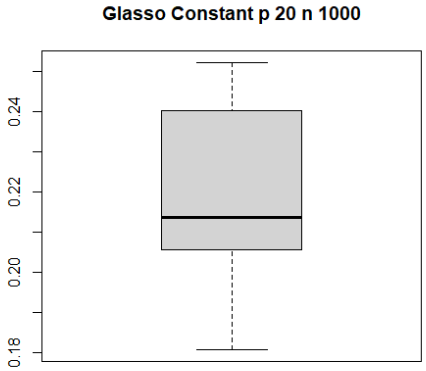

Node-Wise Constant p=40 n=1000
0.95
1.00
1.05
1.10

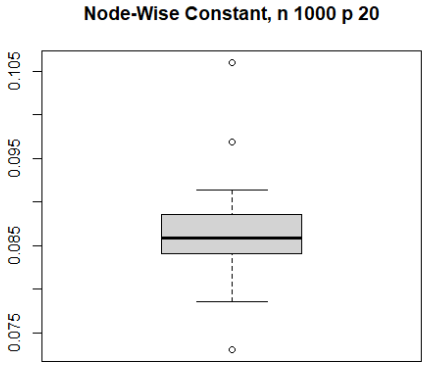

Glasso Constant p=20 n=30
0.9
1.0
1.1
1.2
1.3

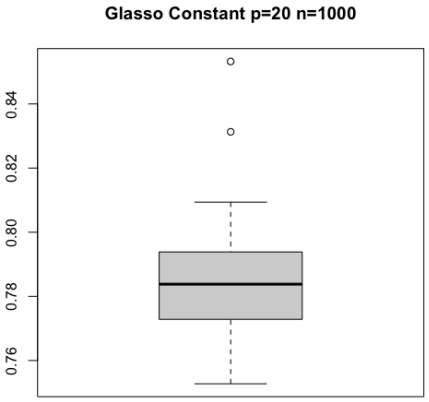

Node-Wise Constant p=20 n=30
0.90
0.95
1.00
1.05
1.10
1.15
1.20

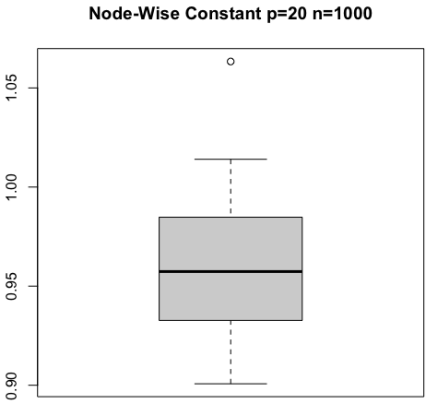

Glasso Constant p=20 n=300
0.50
0.52
0.54
0.56
0.58
0.60
0.62


Node-Wise Constant p=20 n=300
0.75
0.80
0.85
0.90
0.95


Glasso Constant p 20 n 1000
0.18
0.20
0.22
0.24


Node-Wise Constant, n 1000 p 20
0.075
0.085
0.095
0.105


Glasso Constant p=20 n=1000
0.76
0.78
0.80
0.82
0.84


Node-Wise Constant p=20 n=1000
0.90
0.95
1.00
1.05

Bibliography

Belloni, A. and Chernozhukov, V. (2011). Penalized quantile regression in high-dimensional sparse models. *The Annals of statistics*, 39(1):82–130.

Chetverikov, D., Liao, Z., and Chernozhukov, V. (2021). On cross-validated lasso in high dimensions. *The Annals of statistics*, 49(3):1300.

Epskamp, S. (2018). Recent developments on the performance of graphical LASSO networks.

Lee, J. D., Sun, D. L., Sun, Y., and Taylor, J. E. (2016). Exact post-selection inference, with application to the lasso. *The Annals of statistics*, 44(3):907–927.

Qiao, X., Guo, S., and James, G. M. (2019). Functional graphical models.

Tibshirani, R. (2011). Regression shrinkage and selection via the lasso: a retrospective: Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B, Statistical methodology*, 73(3):273–282.

Xinghao, Q. (2021). Lecture notes in st443 machine learning and data mining.

Yuan, M. and Lin, Y. (2007). Model selection and estimation in the gaussian graphical model. *Biometrika*, 94(1):19–35.