STAT3040 Final Project

Alvis Chan u7287079

2023-06-5

Introduction

This project has two core objectives:

- Training a model to predict the price of advertised housing
- Training a model to classify housing close to schools.

There are two datasets available for the project:

- Training dataset: Consisting of 5000 samples of 13 covariates with response variables *price* and *school*
- Testing dataset: Consisting of 3000 samples of 13 covariates without response variables

For each objective, 5 mainstream modelling classes are considered and their respective strengths, limitations and performances are analyzed and discussed. The algorithms and classes included are:

- Prediction
  - Linear Regression
  - Lasso Regularization
  - Polynomial, Splines and General Additive Models
  - Regression Decision Trees
  - Neural Networks
- Classification
  - Logistic Regression
  - Discriminant Analysis
  - Classification Decision Trees
  - Support Vector Machines
  - Neural Networks

Additionally, the analysis is evaluated alongside benchmark metrics obtained through simpler methods, these naive or simple methods will be breifly covered at the start of each major section:

- Additional classes
  - Mean/Median Prediction
  - Naive Bayes Classifier
  - K-Nearest-Neighbours

Explaratory Data Analytics

While examining the data, we observe occurrence of missing data (Table 1)

Table 1: Percentage of Missing Data

	id	price	school	lat	lon	rate	hoa	hvac
Missing (%)	0%	0%	0%	1.9%	1.8%	1.7%	1.4%	1.9%

	garage	view	year	bath	bed	stories	lot	living
Missing (%)	1.4%	1.5%	1.5%	1.5%	1.9%	1.3%	1.3%	1.3%

Conduct a missing completely at random (MCAR) test. From results, data is not missing at random (MNAR). as such we cannot omit the missing values and have to implement data imputation.

	MCAR Results
Statistic	1000
df	195
p.value	0
missing patterns	14

While there are specialized methods to deal with MNAR data, such as selection models and pattern mixture model[1]. The underlying logic and knowledge is beyond the scope of the project. Therefore, we use multiple imputation in attempt to partially eliminate bias of MNAR data with the *mice* package[2].

The parameters used are 7 imputations using a defined pattern-mixture modelling method in *mice* package.

```
# Imputation on missing data
impute <- mice(train, m = 7, method = c("pmm"), seed = 211)
```

Table 2: Summary Statistics for Continuous Variables in Train Dataset

	price	lat	lon	rate	garage	year	bath	bed	stories	lot	living
vars	1.0	2.0	3.0	4.0	5.0	6.0	7.0	8.0	9.0	10.0	11.0
n	5000.0	5000.0	5000.0	5000.0	5000.0	5000.0	5000.0	5000.0	5000.0	5000.0	5000.0
mean	0.0	0.0	0.0	0.0	1.3	4.9	2.7	3.5	1.5	0.0	0.0
sd	1.1	1.0	1.0	1.0	1.4	1.1	1.0	1.0	0.5	0.0	0.8
median	-0.2	0.0	0.1	-0.3	1.0	5.0	3.0	3.0	1.0	0.0	-0.2
trimmed	-0.2	0.0	0.0	-0.3	1.1	5.1	2.6	3.4	1.5	0.0	-0.1
mad	0.4	1.3	1.0	0.1	1.5	1.5	1.5	1.5	0.0	0.0	0.6
min	-1.1	-1.9	-2.9	-0.4	0.0	1.0	0.0	0.0	1.0	0.0	-1.4
max	27.6	2.3	2.4	4.2	11.0	6.0	10.0	10.0	4.0	0.5	11.1
range	28.7	4.3	5.3	4.5	11.0	5.0	10.0	10.0	3.0	0.5	12.5
skew	9.9	0.3	-0.3	3.8	1.4	-0.8	1.1	0.2	0.4	12.5	2.6
kurtosis	193.8	-1.0	-0.2	12.8	4.1	-0.1	3.7	0.9	-1.1	455.4	18.4
se	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

The assumption of normality is crucial in ensuring the validity of statistical inference of tests and models performed on the data later in the report. Kurtosis identifies the shape of the distribution and skew measures asymmetry of the distribution. Acceptable ranges of skew and Kurtosis values are within [-3,3] and [-10,10] respectively[3].

At first glance of the summary table (Table 2), the skew and kurtosis values of *lat*, *lon*, *year*, *bath*, *bed*, *stories*, *garage* are within appropriate range while *price*, *rate*, *lot* and *living* are out of range.

Variables within the range display similar means and medians, suggesting a symmetrical distribution and adherence to the assumption of normality.

However, for other variables such as price and lot, notably high kurtosis values of 193.8 and 455.4, along with skewness values of 9.9 and 12.5, respectively, indicate a pronounced presence of outliers.

Therefore, to meet the normality assumption, transformation of these variables will be necessary. Due to presence of negative values, absolute square root transformations are applied onto the variables.

$$x_{new} = \sqrt{|x|} \times sgn(x)$$

Table 3: Cube Root Transformed Statistics

	price	**rate**	**living**	**lot**
mean	-0.2	-0.3	-0.1	-0.1
median	0.7	0.6	0.1	0.7
min	-1.1	-0.6	-0.2	-1.2
max	5.2	2	0.7	3.3
range	6.3	2.7	0.8	4.5
skew	1.4	3.4	1	0.6
kutosis	2.4	10.1	1.2	-0.8

The summary statistics in Table 3 indicate an improvement in the distribution of variables after applying a cube root transformation. While the kurtosis and skew for *rate* remains outside acceptable range after transformation, implying that the variable is prone to outliers. Considering the mean and median of *rate* are both negative, conventional transformations such as log and roots are limited. As such we will leave *rate* as a caveat in this project.

For categorical variables (Table 3.1), the distribution of *school* and *hoa* are relatively balanced while *hvac* and *view* are significantly unbalanced (17:83 and 78:22 respectively). This may introduce bias towards majority classes when training models. To address this issue, oversampling and undersampling techniques are incooperated into resampling methods to create a more balanced dataset for model training.

Table 3.1: Distriution of Binary Variables

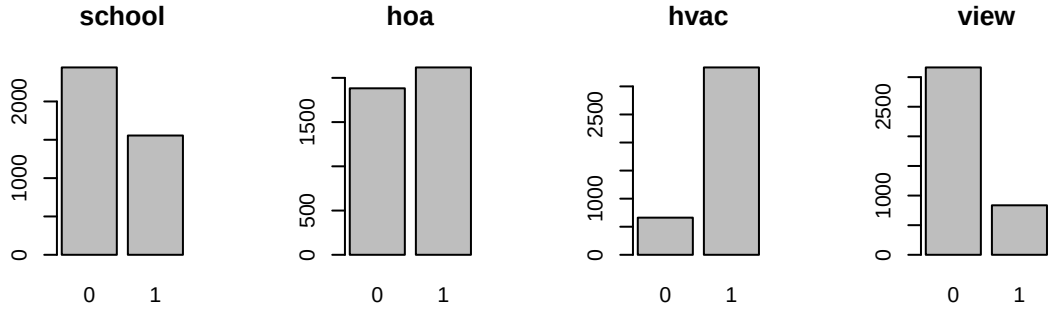


Table 4: Pearson Correlation

	price_sR	school	lat	lon	rate_sR	hoa	hvac	garage	view	year	bath	bed	stories	lot_sR	living_sR
price_sR	1.00	0.06	0.10	-0.28	-0.05	0.01	0.02	0.20	0.16	0.05	0.55	0.34	0.27	0.04	0.64
school	0.06	1.00	0.02	-0.17	0.03	0.32	0.03	0.08	0.08	0.33	0.21	0.12	0.12	0.03	0.27
lat	0.10	0.02	1.00	0.35	0.36	-0.06	0.04	0.02	0.03	-0.12	0.06	0.08	0.07	0.03	0.14
lon	-0.28	-0.17	0.35	1.00	-0.12	-0.29	-0.05	-0.11	-0.15	-0.27	-0.32	-0.23	-0.16	-0.03	-0.46
rate_sR	-0.05	0.03	0.36	-0.12	1.00	0.14	0.04	0.06	0.05	0.15	0.09	0.09	0.04	0.07	0.16
hoa	0.01	0.32	-0.06	-0.29	0.14	1.00	0.05	0.07	0.12	0.61	0.35	0.18	0.32	0.01	0.39
hvac	0.02	0.03	0.04	-0.05	0.04	0.05	1.00	0.02	0.04	0.04	0.07	0.05	0.03	-0.02	0.06
garage	0.20	0.08	0.02	-0.11	0.06	0.07	0.02	1.00	0.18	0.13	0.25	0.15	0.10	0.04	0.23
view	0.16	0.08	0.03	-0.15	0.05	0.12	0.04	0.18	1.00	0.14	0.20	0.12	0.19	0.01	0.20
year	0.05	0.33	-0.12	-0.27	0.15	0.61	0.04	0.13	0.14	1.00	0.45	0.22	0.40	0.03	0.40
bath	0.55	0.21	0.06	-0.32	0.09	0.35	0.07	0.25	0.20	0.45	1.00	0.50	0.58	0.06	0.75
bed	0.34	0.12	0.08	-0.23	0.09	0.18	0.05	0.15	0.12	0.22	0.50	1.00	0.28	0.03	0.55
stories	0.27	0.12	0.07	-0.16	0.04	0.32	0.03	0.10	0.19	0.40	0.58	0.28	1.00	0.02	0.49
lot_sR	0.04	0.03	0.03	-0.03	0.07	0.01	-0.02	0.04	0.01	0.03	0.06	0.03	0.02	1.00	0.04
living_sR	0.64	0.27	0.14	-0.46	0.16	0.39	0.06	0.23	0.20	0.40	0.75	0.55	0.49	0.04	1.00

The response variables, *price* and *school*, do not exhibit strong correlation with majority variables (Table 4). Notably, *price* and *living_sR* are the only pair with moderate correlation (0.61) while the highest correlated variable with *school* is *year* (0.33). Pearson correlation captures linear relationships but does not assume non-linear relationships accurately. Therefore, it is possible the covariates are not linear with responses.

Collinearity between covariates influences the reliability and interpretability of the trained models. In Table 4, some moderate correlations is discovered with *bath* and *living_SR* being most linearly correlated at 0.77. To mitigate impacts of multicollinearity, covariate selection and regularization techniques such as foward-selection and ridge regression are applied when modelling the data.

Naive Predictions

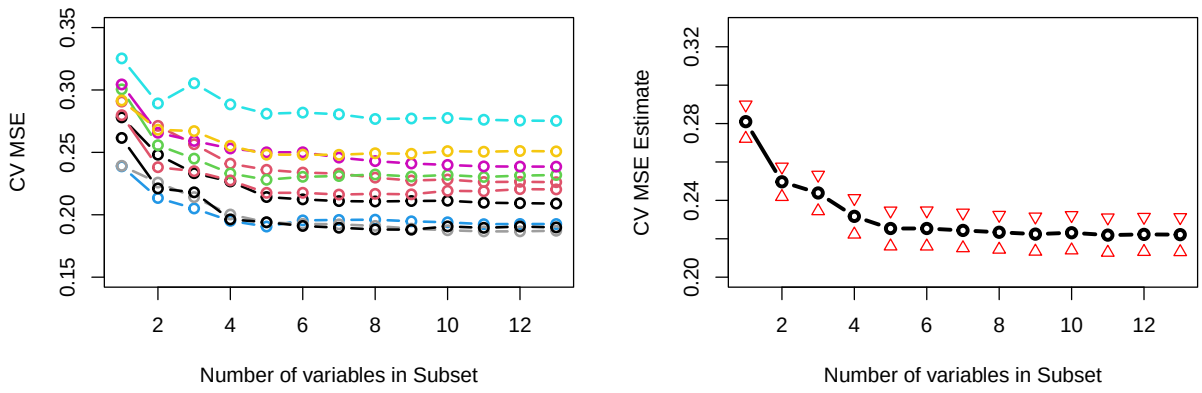
To predict *price_sR* with the provided trainin datam the simplest prediction technique would be to use mean or median as response for all observations. Using mean squared error (MSE) as performance criterion, a 10-fold cross validation is conducted and the average MSE is used as baseline for more complex analysis.

Using R, the MSE for mean approach is 0.472 and the median approach is 0.568.

Prediction 1: Linear Model

A linear model assumes linear relationship between covariates and its response. To optimize the results, a best subset selection is performed and the selected subsets are passed to a custom algorithm for 10-fold cross-validation (10CV) to finalize the best performing subset and model coefficients for the linear model.

Figure 1: Cross−validation MSE and Estimates on Subset Selection



The curve plateaus at around the 5-covariate subset (Figure 1), implying the linear model exhibits best performance at that range.as it plateus. The confidence intervals provides similar observations with 5-covariate subset achieving the lowest intervals.

The final model for multiple linear regression through cross validation and **average test MSE is 0.248.** The best subset has a similar testing MSE of 0.248. A clear interpretation of the model can be made using

$$\hat{y_{price_{SR}}} = \hat{\beta}_0 - \hat{\beta}_1 \cdot rate\_sr - \hat{\beta}_2 \cdot hoa - \hat{\beta}_3 \cdot year + \hat{\beta}_4 \cdot bath + \hat{\beta}_5 \cdot living$$

Table 5: Final GLM Model

Covariate	Estimate	p-value
(Intercept)	0.074	0.132
rate_sR	-0.14	0
hoa	-0.265	0
year	-0.115	0
bath	0.172	0
living_sR	0.57	0

The model confirms some common intuitions, such as a positive correlation between living room size and price, and the negative correlation between tax rates and price. Additionally, the number of baths and novelty of housing correlates with price.

Prediction 2: Lasso regression

Lasso is a regularization technique applied on linear regression. The objective of linear models covered in the prior section is to minimize the sum of residuals. In Lasso regression, an additional term is added on the objective function, which is the sum of the absolute values of the coefficients multiplied by a tuning parameter lambda. Therefore minimizing the quantity[4]

$$\sum_{i=1}^{n}\left(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij}\right)^2 + \lambda\sum_{j=1}^{p}|\beta_j|$$

Similar to LM we will fit a linear model using 10-fold cross validation with lasso regularization with *glmnet* package.

Figure 2: Covariate Coefficients and Lambda

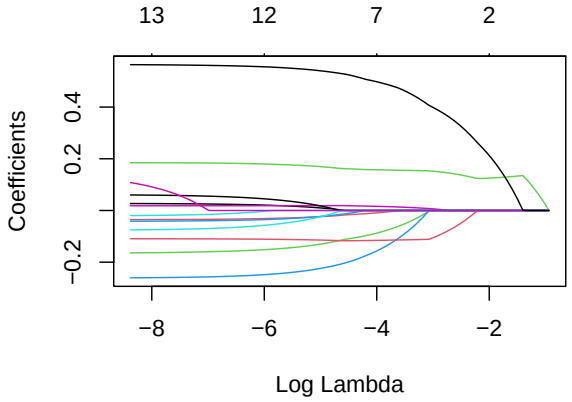
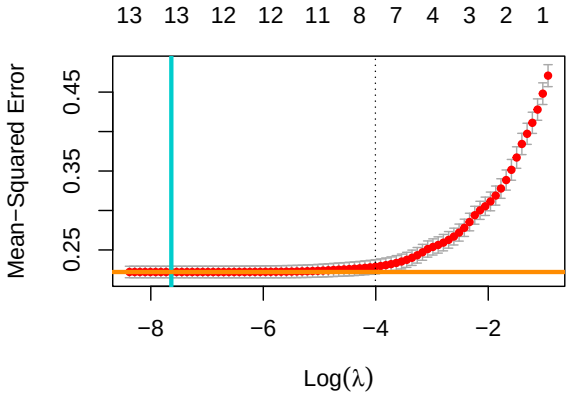


Figure 3: MSE and Lambda



Using a lambda of -7.635 (Figure 3), the final complied model consists of 8 parameters and a testing MSE of 0.2441, which a slight decline to linear regression (Figure 4). However, the interpretation of predictors and response remains similar with linear regression, which reinforces the relationships determined in the prior section.

Figure 4: Actual vs Predicted Values

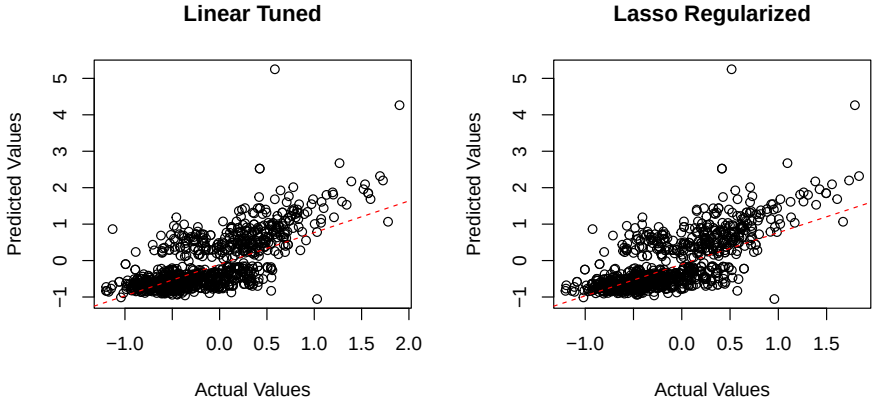


Table 6: Final Lasso Model

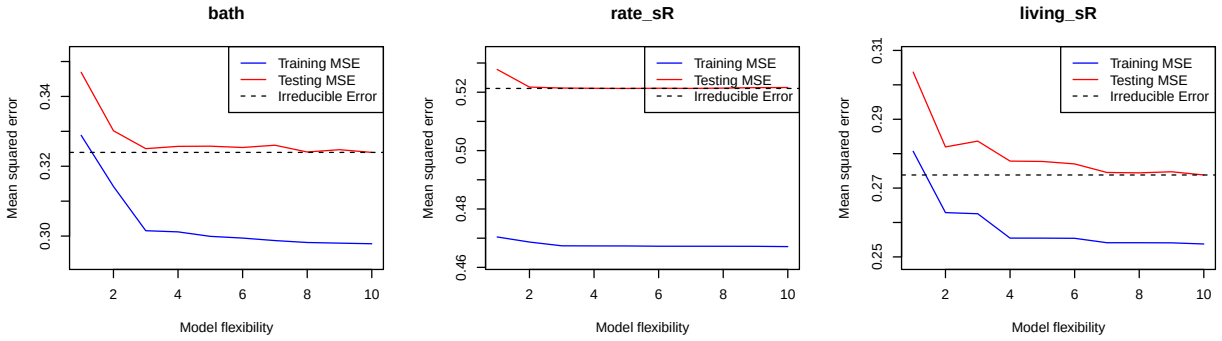
Covariate	Estimate
(Intercept)	0.049
lat	0
lon	-0.009
rate_sR	-0.088
hoa	-0.157
hvac	0
garage	0.016
view	0
year	-0.116
bath	0.157
bed	0
stories	0
lot_sR	0
living_sR	0.498

Lasso regularization excels in covariate selection, flexibility and being robust with multicollinearity. However, Lasso also underestimates coefficients when the sample size is small, hence influencing its model predictive capability.

Prediction 3: General Additive Models

In Predictions 1 and 2 linearity between covariates is assumed. For this section, potential non-linear relationships between the variables are considered. We use polynomial regression to examine potential covariates sharing non-linear relationships with *price_sR* and apply cubic splines to train generalized additive models.

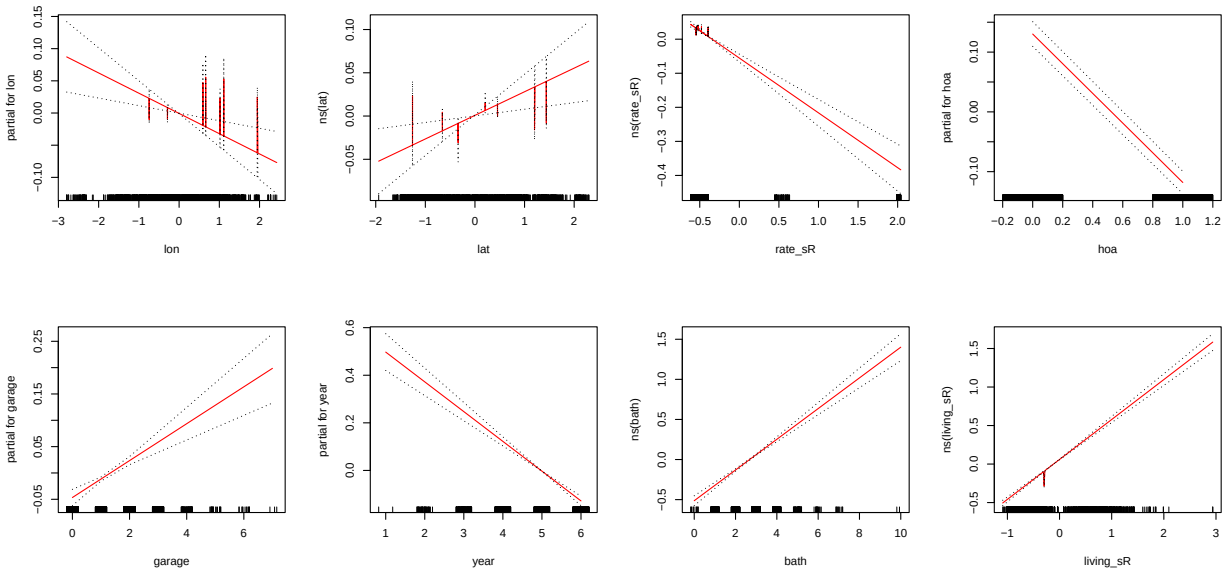
Figure 5: Training and Test MSEs for Polynomial Models of Varing Covariates



A series of polynomial regression are ran onto variables included in lasso regression. Most variables do not show signs of non-linearity through the MSE changes in varing degrees of polynomial regression, with some doing worse at higher flexibilities. However, it is worth noting the observations from the variables *bath*, *rate_sR*, *lat* and *living_sR* display potential non-linearity (Table 5). Judging by trade-off between high bias (left-side) and high variance (right-side), the optimal flexibility for the covariates are:

- *bath*: 3, *rate_sR*: 2, *living_sR*: 4

Figure 6: Splines on Covariates Used



using a bootstrap approach, final testing MSE is 0.2434

Prediction 4: Regression Trees

Decision trees are a different approach to regression. Compared to prior methods, decision trees makes decisions based on a hierarchical structure of if-else conditions. The nature of decision tree implies high interpretability, quick training times, and capable of capturing non-linear relationships.

However, decision trees are prone to overfitting and bias towards features with higher levels. To address these issues, various techniques such as **bagging**, **random forest** and **boosting** are considered to achieve an optimal tree.

Packages used: **tree**, **randomForest**, **gbm**

Start with cross validation to select best size that balances deviance and overfit. this case its 10, prune tree, get results, mse is 0.224.

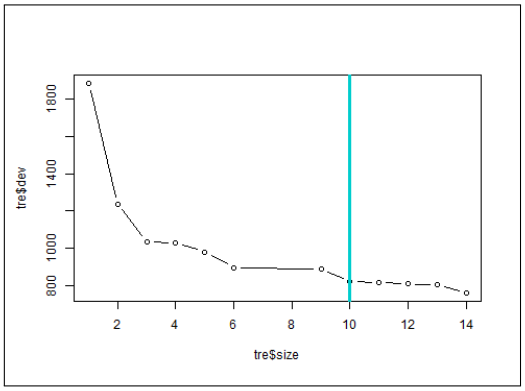
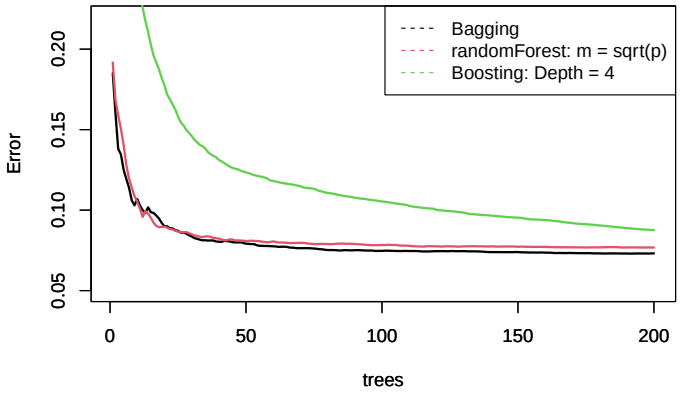


Figure 7: Deviance Against Tree Size: Default Tree

Figure 9: Test Errors vs No. of Trees: Ensemble Methods



Applying the ensemble methods, bagging, random forest, and boosting, **the test MSEs are 0.091, 0.095 and 0.095 respectively.** The performances between the three methods are similar and have greatly improved relative to prior methods (MSE-wise).

The selection of predictors (m) for random forest is set as $\sqrt{p} \approx 4$ and through off-report tests, a depth of 4 presented the best results for boosting. Of the three methods, bagging has achieved the lowest test MSE.

The signifiance of cavoarites determined by the decision trees coincides with majority predictions models explored (Figure 8). In particular, similar to GAM, the trees also regard *lat* and *lon* as crucial predictors.

Trees in general are highly interpretable, simple, capable of handling complex relationships and quick to train at the cost of limited predictive ability. Using ensemble methods such as bagging, boosting and random forest, the weaknesses in individual trees are mitigated to achieve an improved performance.

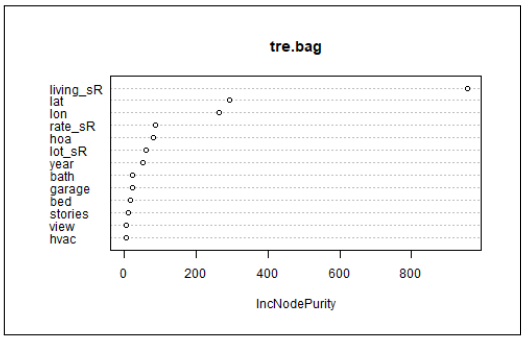


Figure 8: Variable Importance: Bagging

Prediction 5: Neural Network

Neural networks (NN) are capable of capturing complex nonlinear relationships, making them suitable for tasks with intricate data patterns. During training, the network adjusts its internal weights and biases to minimize the difference between predicted and actual values. Neural network regression can automatically learn relevant features from data, but it may be prone to overfitting and requires computational resources.

In this section, NN models are trained using the packages **keras and nnet**. A standard NN model is fitted with **nnet** package as as a benchline for further adjustments using **Keras**. These adjustments include:

- L1 (Lasso) and L2 (Ridge) regularization: Applies penalty term, to encourage smaller models and evenly distributed coefficients. Mitigate the issue of multicollinearity and improve the stability and generalization of the model.
- Number of hidden units and layers (HU/HL): Controls complexity of model. The more units and layers, the more complex the relationship modeled but triaining times increases and may overfit the trainin data.
- Cross-validation splits: The ratio of training data used for model training and validation.
- Learning rates (LR): Decides the step size at which the model adjusts its weights. Essentially handling how fast or slow the model learns. High rate can cause model to coverge to quick while a low learning rate requires more iterations to establish robust relationships between predictors and responses.
- Dropout rates (DR): Regularization to prevent over-fitting data via dropping nodes at random. Thus creates a more robust model that cannot rely on a strong set of nodes.
- Callbacks: Options that can be specified to be executed at various points during the training process. They provide a way to customize and enhance the training steps. In this section, early stopping and learning rate adjustments are considered.

Packages: **nnet, tensorflow, caret, keras**

```
nn.base <- nnet(price_sR ~ ., size=25, data = pr, linout = TRUE)
```

With *nn.base* model, it contains 376 weights and measures a testing MSE of 0.169.

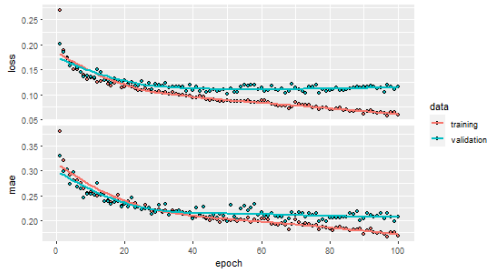
The parameters and options are adjusted in 4 models trained with **keras**. The parameters, *epochs, batch_size, validation_split, first and second layer units*, are fixed at 100, 32, 0.3, 100 and 50 respectively. Furthermore, the rectified linear unit (ReLU) activation function, $g(z)$, is applied during the training of all models.

Table 7: NN Summary Statistics

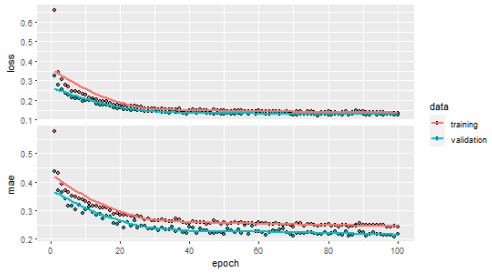
Model	Parameters	Specifications	Test MSE
base	376	1 HL, 25 HU	0.169
nnmod.(a)	6501	2 HL, (100,50) HU, LR = 0.005	0.151
nnmod.(b)	6501	2 HL, (100,50) HU, LR = 0.002, DR = (0.4,0.3), $\lambda_{Ridge} = 0.001$	0.145
nnmod.(c)	7491	3 HL, (100,50,20) HU, LR = 0.002, DR = (0.3,0.2,0.1), $\lambda_{lasso} = \lambda_{Ridge} = 0.001$	0.145
nnmod.(d)	7891	LR = 0.005, DR = (batch normalization,0.2,0.1), $\lambda_{lasso} = \lambda_{Ridge} = 0.001$, callback = Early stopping, Reduce LR on plateau	0.145

The NN models (Table 7) share similar testing MSEs ranging between 0.14 to 0.17. All NN models trained through keras has improved from the base NN model based on the testing MSEs but plateus at 0.145 for the final two models with most tuning.

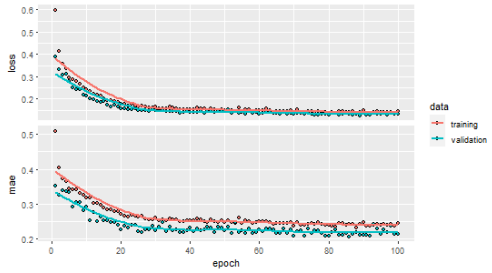
Model (a), with a high learning rate, highly overfits the training data as training loss is lower than validation loss (Fig.9), thus it is not reliable on unseen data. **The best performing NN model is nnmod.(b)** with 2 hidden layers (100 units to 50 units to 1 unit), a learning rate of 0.002, drop rate of 0.4 to 0.3 and $\lambda_{Ridge} = 0.001$. The increase in MSE in models (c) and (d) may be subject to overfitting. With more parameters than available training data (n = 4000), more parameters and layers might have exacerbate the issue of limited data, and cause struggle in generalization of final weights. Additionally, the model may start to fit the noise rather than the underlying patterns, resulting in a higher MSE when applied to unseen data.



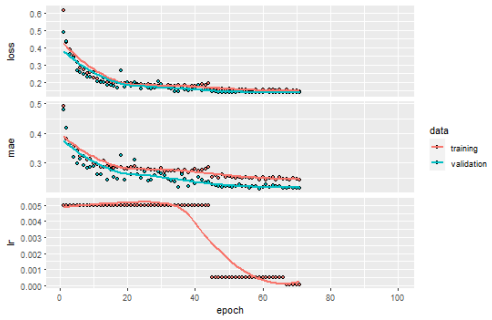
(a) 2 layers: 100->50->1



(b) + Ridge & Dropout



(c) 3 layers:100->50->20->1+Lasso



(d) + Batch Norm & Callbacks

Figure 9: MSE (loss) and MAE of keras Models

Prediction: Discussion and Conclusion

This section we review the prediction models trained to predict the response, *price_sR* (Table 8). For each method and class, the trained models demonstrate progressive improvement in testing MSE with the tree model trained through a Bagging approach achieving the best MSE of 0.091.

Table 8: Summary of Prediction Models

Model	Testing MSE
Naive: Median	0.568
Naive: Mean	0.472
Linear: Best Subset Selection (CV)	0.248
Linear: Lasso Regularization (CV)	0.244
Polynomial and Splines (Bootstrap)	0.243
Trees: Base (CV)	0.224
Trees: Random forest	0.095
Trees: Boosting	0.095
Trees: Bagging	**0.091**
NN: Base (nnet)	0.169
NN: Best (keras)	0.140
NN: Over-paramterized (keras)	0.145

Notably, the respective coefficients, significance, and weights for predictors *lat* and *lon* are more crucial in polynomial and spline models, tree models and NN models. Suggesting a non-linear relationship between *lat*, *lon* and *price_sR*. Further investigation in underlying relationship are needed to achieve better results.

Furthermore, in tree models and NN models, *bath* is not addressed as significant as linear and polynomial models. This raises the concern of multicollinearity between *bath* and *living_sR* (Table 2) in our models.

In the prediction models, *living_sR* is significant in predicting *price_sR*. Which was anticipated as the most correlated covariate (Table 2).

Apart from multicollinearity, two other major limitations restricts model performance. Firstly, missing data was addressed with multiple imputation which only partially recovers the bias and explanatory power of covariates. Secondly, given the ratio of 5:3 between known and unknown data, the models trained may not accurately capture the full complexity and variability of the underlying patterns and may struggle to generalize well to unseen data. These issues needs to be properly addressed in following analysis to train more robust models.

Naive Classificaitons

There are a few simple classification techniques to predict whether a school is nearby the housing (the covariate *school*). Naive Bayes classifier and K-Nearest-Neighbours (KNN) are naive classification techniques that make simple assumptions about the data. Using classification rate (CR) as performance criterion, a 10-fold cross validation is conducted and the average correct classification rate is used as benchmark for further analysis.

Naive Bayes classifier is a probabilistic classifier while KNN is a non-parametric classifier that assigns categories to an observation based on majority vote of its k nearest neighbours.

Through packages *naivebayes* and *class*, the CRs are 0.671 and 0.772 for Naive Bayes and 5-KNN.

Classification: Logsitic Model

Logistic model is a type of generalized linear model (GLM) that models the probability of a binary class. It maps a linear combination of predictors to a value between 0 and 1 using a form of $\frac{exp(p(x))}{1+exp(p(x)}$.

A backward selection method is used to identify the most influential variables. Then a boostrap is conducted to retrieve the best performing coefficients using a default threshold of 0.5 and cauchit link function.

Packages used: *stats*

```
mod <- glm(school ~ ., data = cl, family = binomial(link = 'cauchit'))
mod <- step(mod, direction = "backward")
```

```
## school ~ lat + lon + rate_sR + hoa + year + bed + stories + living_sR
```

The error rate (overall) is minimized with a threshold of 0.6 (33.0%) compared to the default assumption of 0.5 (34.1%) (Figure 10). However, using a threshold of 0.5 results in a higher AUC (0.657) than 0.6 (0.598) (Figure 11). While the overall classification rate provides a general measure of prediction performance, AUC-ROC provides a more comprehensive evaluation on the prediction by also considering trade-off between FP and FN. Therefore, using a threshold of 0.5 is appropriate for this prediction.

The model suggests housing with more bedrooms, higher tax rates are less likely to have a school nearby. A potential reason may be that housing with higher tax rates and more bedrooms are located at a more wealthy neighborhood with no schools nearby.

The final logistic model achieves an accuracy rate of 0.679, with approriate false-positive (FP) rate of 0.276 and false-negative (FN) rate of 0.399. The significance and implications of variables selected are discussed alongside discriminant analysis in the next section.

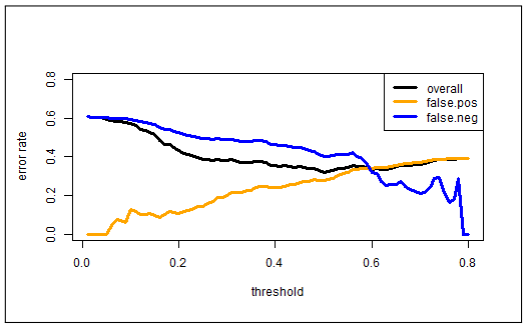


Figure 10: CR in Varing Thresholds

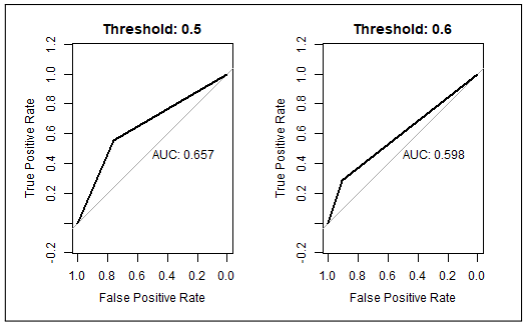


Figure 11: ROC Curves

Classification: Discriminant Analysis

While logistic regression is utlised for classifying binary or categorical variables, discriminant analysis can be used when the response variable is categorical and represents distinct groups. There are also differences in assumptions, discriminant analysis emphasizes on independence, equal variance and normality of covariates.

A k-fold-cross-validation forward selection approach is used to identify the most influential variables. The subset of predictors are passed onto model training with linear and quadratic discriminant analysis.

Linear Discriminant Analysis (LDA)

school~lon+lat

The latitude (*lat*) and longtitude (*lon*) are sole covariates chosen by LDA (5-fold). Apart from the intuitive implication that the geographic coordinates are determinative of whether a school is nearby (*school*) (Figure 12.), it also implies that the structure of housing does not significant change the posterior probabilities on the classification of *school* linearly.

The LDA model achieves an accuracy rate of 0.695 (Figure 13) alongside an AUC-ROC score of 0.638.

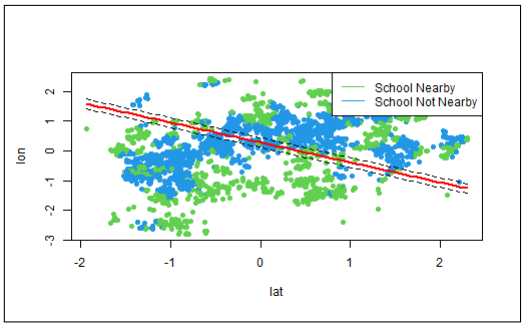


Figure 12: Linear Decision Boundary and Confidence Intervals

Quadratic Discriminant Analysis (QDA)

school~lon+lat+year+rate_sR+garage

Similarly, QDA (10-fold) selects longtitude (*lon*) and latitude (*lat*) as predictors for *school*. Additionally, QDA also chooses presence of heating/cooling/ventilation system (*hvac*), Home Owner Association (*hoa*), garage spaces (*garage*) and lot size (*lot_sR*) as covariates influential for the classification of *school*.

Notably, the group means of *hoa* between the decision boundaries suggest housing without a school nearby tend to not have a HOA.

The QDA model achieves an accuracy rate of 0.733 (Figure 13) alongside an AUC-ROC score of 0.721 (Figure 13).

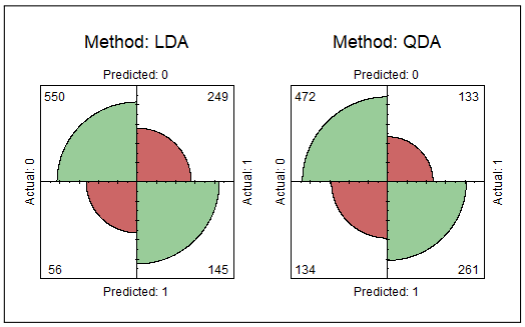


Figure 13: Confusion Matrix on Validation Set

GLM, LDA and QDA

Inspecting the predictors of GLM, LDA, and QDA, there is a similarity of using *lat* and *lon* as covariates for response variable *school*, thus reinforces the importance of the two predictors. Contrast to GLM, through forward selection, LDA and QDA trains a simpler model with classification rates within their respective 95% confidence intervals (Table 10).

Table 9: Confidence Intervals

Model	CI
GLM (Bootstrap)	(0.34,0.341)
LDA (5-fold CV)	(0.284,0.322)
QDA (10-fold CV)	(0.247,0.269)

Overall, **LDA and QDA models performs slightly better on the validation set** relative to GLM (Table 10). For further studies, it should be noted that LDA and QDA generally performs better with larger datasets to estimate covariance matrices more accurately. Whilst not a major issue in this training dataset and predictor subset, discriminant analysis are also more sensitive to multicollinearity relative to GLM.

Classification: Classification Decision trees

Similar to regression trees, classification decision trees makes decisions based on a hierarchical structure of if-else conditions. The difference between the two models include:

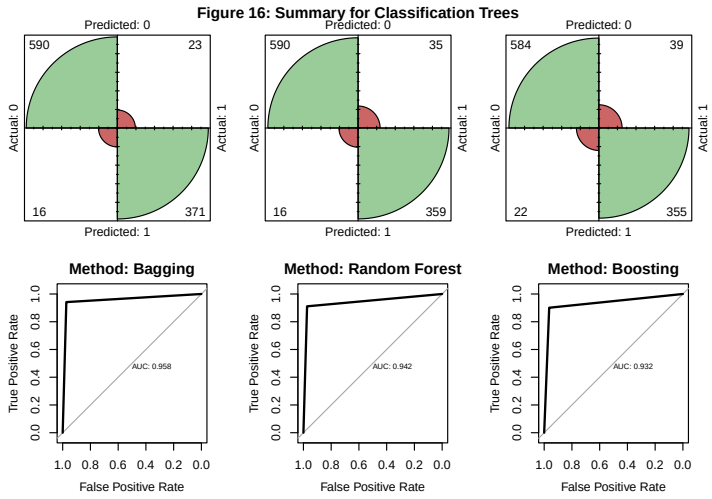
- Different criterion: In a classification setting, variance measures such as RSS cannot be used. Alternative measures are error (mis-classification) rates, Gini index and cross-entropy.
- Data partitioning: Classification trees aim to partition the data into groups as accurate as possible using branches of condition statements. While regression trees capture and creating groups to contain similar responses.

Once again, decision trees are prone to overfitting and bias towards features with higher levels. Therefore, various techniques such as **bagging**, **random forest** and **boosting** are considered to achieve an optimal tree to address the issues.

Packages used: **tree**, **randomForest**, **gbm**

```
tre <- cv.tree(tree(as.factor(school) ~ ., data = cl), K = 10)
```

Start with cross validation to select best size that balances deviance and overfit. this case its 11, prune tree, get results, mse is 0.778.



Applying the ensemble methods, bagging, random forest, and boosting, **the test MSEs are 0.961, 0.949 and 0.939 respectively** (Figure 16). Correspondingly, the methods achieves AUC scores of *0.958*, *0.942*, and *0.932*. The performances between the three methods are similar and have improved significantly relative to prior methods (Classification rate and ROC-AUC).

The selection of predictors (m) for random forest is set as $\sqrt{p} \approx 4$ and through off-report tests, a depth of 4 presented the best results for boosting. Of the three methods, bagging has achieved the lowest test MSE.

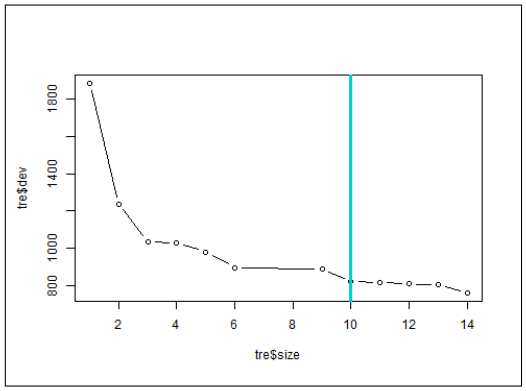


Figure 14: Deviance and Against Clssification Tree Size

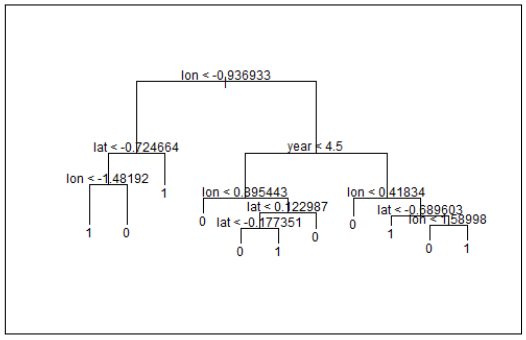


Figure 15: Pruned Classification Tree - 11

Classification: Support Vector Machines (SVM)

In prior sections, decision trees significantly outperform logistics and discriminant regression methods. Decision trees excel in capturing relationships without explicit prior feature selection and are more robust to outliers. Which conjugates that the data may be distinctivly partitioned unique groups. If the conjecture is true, than support vector machines (SVM) should also outperform logstics and discriminant methods since SVM aims to find a plane to separate groups of classifications.

Package used: *e1071*

```
svm.mod <- svm(as.factor(school) ~ ., data = cl, kernel = 'linear', cost = 10, scale = FALSE)
```

As a benchmark model, we fit a linear SVM with a cost argument of 10. This benchmark model achieves an accuracy of 0.658 and AUC value of 0.639. To fine-tune the SVM model, the cost and kernel parameters are optimized using *tune()*. Which utilizes cross-validation to find combinations of parameter values that result in the best performance. The kernal functions examined differs as such:

- Linear: Trains a linear decision boundary. From prior models, this kernal should have the lowest accuracy as data do not share an apparent linear relationship with *school*.
- Radial Basis: Trains a hyperplane decision boundary. By mapping the data to a higher dimensional plane, this method aims to distinguish linear and non-linear relationships between the covariates and response variables.
- Polynomial: Trains a non-linear decision boundary. It is able to capture high complexity relationships but can be prone to overfit if not tuned correctly.

Summary of tuning (Figure 17):

- Linear
  - Accuracy: 0.662
  - AUC: 0.618
  - Cost: 0.001
- Radial
  - Accuracy: 0.843
  - AUC: 0.825
  - Cost: 100
- Polynomial
  - Accuracy: 0.800
  - AUC: 0.779
  - Cost: 100

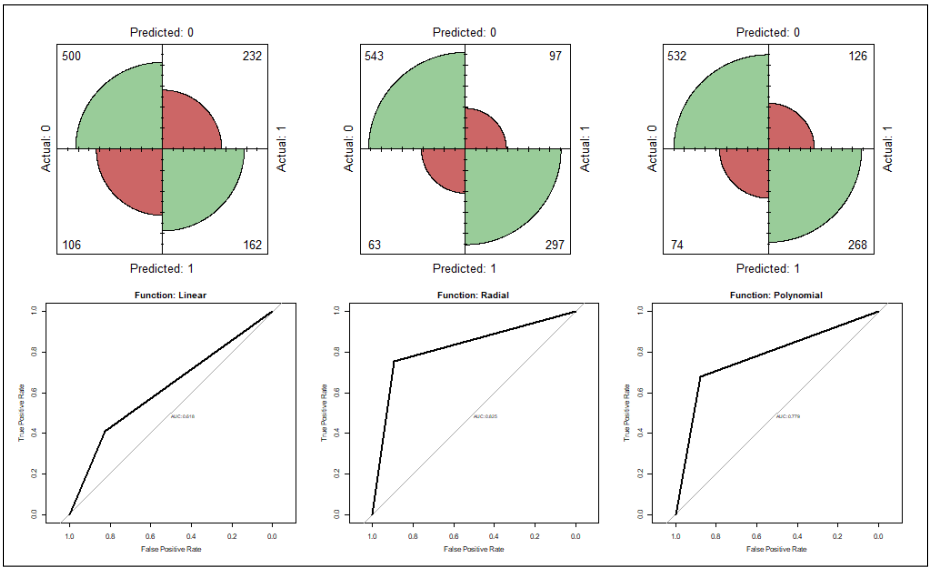


Figure 17: Summary for SVM

Overall, radial kernal presents the most optimal prediction results on the validation data set through CV with polynomial kernals' performance similar. Notably, linear kernal is least effective as conjugeted, reinforcing the hypothesis that covariates share a non-linear relationship with *school* (Figure 12, 17).

Classification: Neural Network

Neural networks (NN) are also capable of performing classification tasks. Similar to using NN for prediction of *price_sR*, NN models are trained using packages **keras and nnet**. The differences underlying classification NN and prediction NN include,

- Difference in evaluation methods: Cost functions used to estimate model performance and describe discrpency between estimated and actual values. For classification models in this section a binary cross-entropy loss function is used.
- Difference in performance metrics: Loss (MSE) and mean absolute error rate (MAE) are common criterions used for regression. Classification utilities accuracy, precision, and F1 score to assess performance of the model. For classification models in this section accuracy is used.
- Activation functions: Classification employs activation functions that provides categorical or binary outputs such as sigmoid (binary) and softmax (multiclass).

The adjustments to parameters are similar to prediction with NN in prior section.

Packages: **nnet, tensorflow, caret, keras**

```
nn.base <- nnet(as.factor(school) ~ ., size=25, data = cl, decay=5e-4, maxit=200)
```

With benchmark NN model, it contains 376 weights and measures a testing accuracy of 0.834.

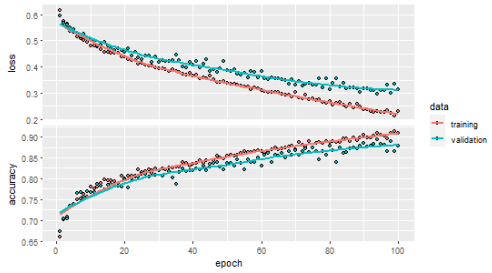
The parameters and options are adjusted in 4 models trained with **keras**. The parameters, *epochs, batch_size, validation_split, first and second layer units*, are fixed at 100, 32, 0.3, 100 and 50 respectively. Furthermore, the rectified linear unit (ReLU) activation function, $g(z)$, is applied during the training of all models.

Table 10: NN Summary Statistics

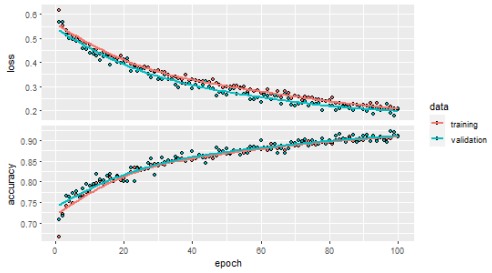
Model	Parameters	Specifications	Test Accuracy
base	376	1 HL, 25 HU	0.834
nnmod.(a)	6501	2 HL, (100,50) HU, LR = 0.002	0.844
nnmod.(b)	6501	2 HL, (100,50) HU, LR = 0.005, DR = (0,0.1)	0.863
nnmod.(c)	7751	3 HL, (100,50,20) HU, LR = 0.003, DR = (0.2,0.1,0.05)	0.86
nnmod.(d)	8151	LR = 0.005, DR = (batch normalization,0.1,0.05), λ_{lasso} = 0.001, callback = Early stopping, Reduce LR on plateau	0.832

The NN models (Table 10) share similar classification rates ranging between 0.83 to 0.86. **The best performing NN model is nnmod.(b)** with 2 hidden layers (100 units to 50 units to 1 unit) and a learning rate of 0.002. Importantly, the utilization of regularization and dropout techniques did not produce improved outcomes. This can be attributed to various factors, such as insufficient data, persistent underfitting of the model, or inadequate parameter tuning. This aspect will be considered a caveat for this project.

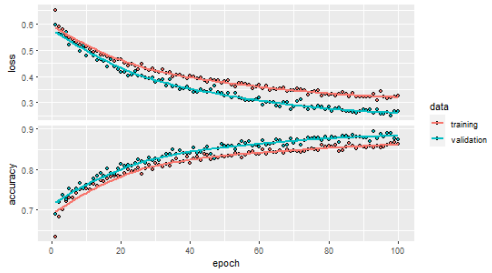
Similarly, the decrease in accuracy of models (c) and (d) is attributable to underfit and limited by data size. With more parameters than available training data (n = 4000), excessive parameters and layers exacerbates the issue of limited data, which results difficulty generalizing final weight values. Additionally, the model may start to fit the noise rather than the underlying patterns, resulting in a decrease in accuracy when applied to unseen data.



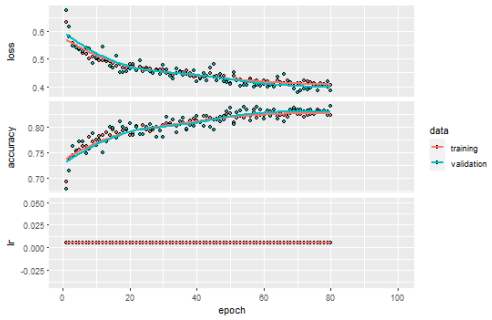
(a) 2 layers: 50->25->1



(b) + Ridge & Dropout



(c) 3 layers:100->50->20->1+Lasso



(d) + Batch Norm & Callbacks

Figure 18: Binary Cross-Entropy (loss) and Accuracy of keras Models

Classification: Discussion and Conclusion:

This section we review the classification models trained to predict the response, *school* (Table 11). The best performing approaches are decisions trees, SVM and nerual networks with a bagging tree model achieving the highest accuracy of 0.961 on validation data.

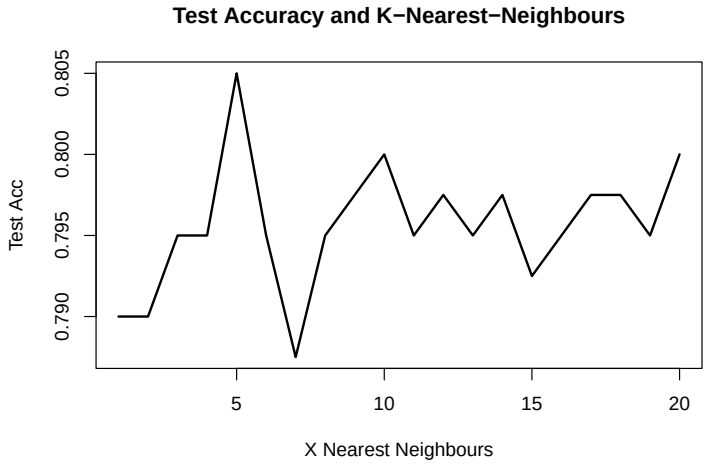
Notably, the covariates *lat* and *lon* are regarded as significant predictors in majority models. The significance becomes more pronounced in models examining non-linear relationships between variables, indicating non-linear relationship between *lat*, *lon* and *school* (Figure 12). Further investigation in underlying relationship is warranted to train better models.

While performing the analysis, two major limitations restricting model performance is discovered. Firstly, missing data, preprocessed with multiple imputation, failed to recover the lost bias and thus the analysis excluded the patterns in missing data. Secondly, given the unbalanced ratio etween known and unknown data, the models fitted have limited effectiveness in capturing the full complexity and variability of the underlying patterns. These issues needs to be properly addressed in following analysis to train more robust models.

Table 11: Summary of Classification Models

Model	**Testing Accuracy**
Naive Bayes	0.671
KNN (5K)	0.772
Logistic: Backward Selection (Bootstrap)	0.679
LDA: Forward Selection (5CV)	0.695
QDA: Forward Selection (10CV)	0.733
Trees: Base (10CV)	0.778
Trees: Random forest	0.949
Trees: Boosting	0.939
Trees: Bagging	**0.961**
SVM: Base (Linear)	0.658
SVM: Linear (Tuned)	0.662
SVM: Radial (Tuned)	0.843
SVM: Polynomial (Tuned)	0.800
NN: Base (nnet)	0.834
NN: Best (keras)	0.863
NN: Over-paramterized (keras)	0.832

Appendix



References

1. Enders, C. K. (2022). *Applied Missing Data Analysis.* Guilford Publications.
2. Woods, A. D., Gerasimova, D., Van Dusen, B., Nissen, J. M., Bainter, S. A., Uzdavines, A., Davis-Kean, P. E., Halvorson, M. A., King, K. M., Logan, J. a. R., Xu, M., Vasilev, M. R., Clay, J. M., Moreau, D., Joyal-Desmarais, K., Cruz, R. A., Brown, D. M. Y., Schmidt, K., & Elsherif, M. (2023). Best practices for addressing missing data through multiple imputation. Infant and Child Development. https://doi.org/10.1002/icd.2407
3. Griffin, M. M., & Steinbrecher, T. D. (2013). Large-Scale Datasets in Special Education Research. In *Elsevier eBooks* (pp. 155–183). https://doi.org/10.1016/b978-0-12-407760-7.00004-9
4. James, G., Witten, D., Trevor, H., Tibshirani, R. (2021). *An Introduction to Statistical Learning* Springer

R-Packages

1. Baptiste Auguie (2017). gridExtra: Miscellaneous Functions for "Grid" Graphics. R package version 2.3. https://CRAN.R-project.org/package=gridExtra
2. JJ Allaire and François Chollet (2023). keras: R Interface to 'Keras'. R package version 2.11.1. https://CRAN.R-project.org/package=keras
3. JJ Allaire and Yuan Tang (2022). tensorflow: R Interface to 'TensorFlow'. R package version 2.11.0. https://CRAN.R-project.org/package=tensorflow
4. Barret Schloerke, Di Cook, Joseph Larmarange, Francois Briatte, Moritz Marbach, Edwin Thoen, Amos Elberg and Jason Crowley (2021). GGally: Extension to 'ggplot2'. R package version 2.1.2. https://CRAN.R-project.org/package=GGally
5. Brandon Greenwell, Bradley Boehmke, Jay Cunningham and GBM Developers (2022). gbm: Generalized Boosted Regression Models. R package version 2.1.8.1. https://CRAN.R-project.org/package=gbm
6. David Meyer, Evgenia Dimitriadou, Kurt Hornik, Andreas Weingessel and Friedrich Leisch (2021). e1071: Misc Functions of the Department of Statistics, Probability Theory Group (Formerly: E1071), TU Wien. R package version 1.7-9. https://CRAN.R-project.org/package=e1071
7. H. Wickham. ggplot2: Elegant Graphics for Data Analysis. Springer-Verlag New York, 2016.
8. Hadley Wickham, Romain François, Lionel Henry and Kirill Müller (2022). dplyr: A Grammar of Data Manipulation. R package version 1.0.8. https://CRAN.R-project.org/package=dplyr

9. Jarek Tuszynski (2021). caTools: Tools: Moving Window Statistics, GIF, Base64, ROC AUC, etc. R package version 1.18.2. https://CRAN.R-project.org/package=caTools
10. John Fox and Sanford Weisberg (2019). An {R} Companion to Applied Regression, Third Edition. Thousand Oaks CA: Sage. URL: https://socialsciences.mcmaster.ca/jfox/Books/Companion/
11. Kuhn, M. (2008). Building Predictive Models in R Using the caret Package. Journal of Statistical Software, 28(5), 1–26. https://doi.org/10.18637/jss.v028.i05
12. Majka M (2019). naivebayes: High Performance Implementation of the Naive Bayes Algorithm in R. R package version 0.9.7, <URL: https://CRAN.R-project.org/package=naivebayes>.
13. Revelle, W. (2021) psych: Procedures for Personality and Psychological Research, Northwestern University, Evanston, Illinois, USA, https://CRAN.R-project.org/package=psych Version = 2.1.9.
14. R Core Team (2021). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.
15. Stef van Buuren, Karin Groothuis-Oudshoorn (2011). mice: Multivariate Imputation by Chained Equations in R. Journal of Statistical Software, 45(3), 1-67. DOI 10.18637/jss.v045.i03.
16. Taiyun Wei and Viliam Simko (2021). R package 'corrplot': Visualization of a Correlation Matrix (Version 0.92). Available from https://github.com/taiyun/corrplot
17. Thomas Lumley based on Fortran code by Alan Miller (2020). leaps: Regression Subset Selection. R package version 3.1. https://CRAN.R-project.org/package=leaps
18. Tierney N, Cook D (2023). "Expanding Tidy Data Principles to Facilitate Missing Data Exploration, Visualization and Assessment of Imputations." Journal of Statistical Software, 105(7), 1-31. doi: 10.18637/jss.v105.i07 (URL: https://doi.org/10.18637/jss.v105.i07).
19. Trevor Hastie (2023). gam: Generalized Additive Models. R package version 1.22-2. https://CRAN.R-project.org/package=gam
20. Venables, W. N. & Ripley, B. D. (2002) Modern Applied Statistics with S. Fourth Edition. Springer, New York. ISBN 0-387-95457-0
21. Venables, W. N. & Ripley, B. D. (2002) Modern Applied Statistics with S. Fourth Edition. Springer, New York. ISBN 0-387-95457-0
22. Xavier Robin, Natacha Turck, Alexandre Hainard, Natalia Tiberti, Frédérique Lisacek, Jean-Charles Sanchez and Markus Müller (2011). pROC: an open-source package for R and S+ to analyze and compare ROC curves. BMC Bioinformatics, 12, p. 77. DOI: 10.1186/1471-2105-12-77 http://www.biomedcentral.com/1471-2105/12/77/
23. Yihui Xie (2021). knitr: A General-Purpose Package for Dynamic Report Generation in R. R package version 1.37.
24. Wood, S.N. (2011) Fast stable restricted maximum likelihood and marginal likelihood estimation of semiparametric generalized linear models. Journal of the Royal Statistical Society (B) 73(1):3-36