Predicting a Building's Energy Efficiency

Jarred Priester

2/5/2022

1. Introduction

- 1.1 Overview of the problem
- 1.2 description of the dataset
- 1.3 goal of the project
- 1.4 plan of action

2. Data Cleaning

- 2.1 downloading the data
- 2.2 cleaning NAs
- 2.3 cleaning black observations
- 2.4 scaling the data set

3. Data Visualization

4. Models

- 4.1 train and test sets
- 4.2 linear regression
- 4.3 ridge regression
- 4.4 random forest
- 4.5 ensemble

5. Results

- 5.1 table of results
- 5.2 plot of results
- 5.3 brief thoughts on results

6. Conclusion

- 6.1 summary
- 6.2 limitations
- 6.3 next steps

1. Introduction

1.1 overview of the problem

With record high and low temperatures across the globe, it is becoming increasing important to be efficient when it comes to heating and cooling our buildings. Whether you are trying to reduce the cost of your energy bill or you're trying to reduce your carbon footprint, improving the energy efficacy of your building can both save you some money and even help the environment. We will be looking at a data set that can help us with both!

1.2 description of the data set

This data set we will be using is from the University of California, Irvine Machine Learning Repository.

The following is UCI's information on the data set:

Source:

The dataset was created by Angeliki Xifara (Civil/Structural Engineer) and was processed by Athanasios Tsanas (Oxford Centre for Industrial and Applied Mathematics, University of Oxford, UK).

Data Set Information:

We perform energy analysis using 12 different building shapes simulated in Ecotect. The buildings differ with respect to the glazing area, the glazing area distribution, and the orientation, amongst other parameters. We simulate various settings as functions of the afore-mentioned characteristics to obtain 768 building shapes. The dataset comprises 768 samples and 8 features, aiming to predict two real valued responses. It can also be used as a multi-class classification problem if the response is rounded to the nearest integer.

Attribute Information:

The dataset contains eight attributes (or features, denoted by X1...X8) and two responses (or outcomes, denoted by y1 and y2). The aim is to use the eight features to predict each of the two responses.

Specifically:

X1 Relative Compactness

X2 Surface Area

X3 Wall Area

X4 Roof Area

X5 Overall Height

X6 Orientation

X7 Glazing Area

X8 Glazing Area Distribution

y1 Heating Load

y2 Cooling Load

https://archive.ics.uci.edu/ml/datasets/Energy+efficiency

1.3 goal of the project

The goal of this project is to create multiple regression models that come up with predictions for both the heating and cooling load. We will take the best performing model.

1.4 plan of action

We will download and cleaning the data. We will use visualization tools to get a better understanding of the data that we are working with. THen we will be create the following regression models: *linear regression*, *ridge regression*, *random forest*, *ensemble*. The models will be evaluated by using the Root Mean Squared Error (RMSE)

RMSE =

$$\sqrt{\frac{1}{n}\sum_{u,i} = (\hat{y}_{u,i} - y_{u,i})^2}$$

Finally, we will create a results table of the models and evaluate the results

2. Data Cleaning

2.1 downloading the data

```
#loading libraries
library(tidyverse)
library(dplyr)
library(ggplot2)
library(ggthemes)
library(caret)
library(elasticnet)
library(knitr)
library(matrixStats)

#loading the data
energy <- read.csv("../Building_Energy_Efficiency/ENB2012_data.csv")
```

First, let's that a quick look at the data

```
head(energy)
```

```
##     X1    X2    X3     X4 X5 X6 X7 X8    Y1    Y2
## 1 0.98 514.5 294.0 110.25  7  2  0  0 15.55 21.33
## 2 0.98 514.5 294.0 110.25  7  3  0  0 15.55 21.33
## 3 0.98 514.5 294.0 110.25  7  4  0  0 15.55 21.33
## 4 0.98 514.5 294.0 110.25  7  5  0  0 15.55 21.33
## 5 0.90 563.5 318.5 122.50  7  2  0  0 20.84 28.28
## 6 0.90 563.5 318.5 122.50  7  3  0  0 21.46 25.38
```

```
summary(energy)
```

```
##        X1              X2              X3              X4
##  Min.   :0.6200   Min.   :514.5   Min.   :245.0   Min.   :110.2
##  1st Qu.:0.6825   1st Qu.:606.4   1st Qu.:294.0   1st Qu.:140.9
##  Median :0.7500   Median :673.8   Median :318.5   Median :183.8
##  Mean   :0.7642   Mean   :671.7   Mean   :318.5   Mean   :176.6
##  3rd Qu.:0.8300   3rd Qu.:741.1   3rd Qu.:343.0   3rd Qu.:220.5
```

```
##  Max.   :0.9800   Max.   :808.5   Max.   :416.5   Max.   :220.5
##        X5             X6             X7               X8              Y1
##  Min.   :3.50   Min.   :2.00   Min.   :0.0000   Min.   :0.000   Min.   : 6.01
##  1st Qu.:3.50   1st Qu.:2.75   1st Qu.:0.1000   1st Qu.:1.750   1st Qu.:12.99
##  Median :5.25   Median :3.50   Median :0.2500   Median :3.000   Median :18.95
##  Mean   :5.25   Mean   :3.50   Mean   :0.2344   Mean   :2.812   Mean   :22.31
##  3rd Qu.:7.00   3rd Qu.:4.25   3rd Qu.:0.4000   3rd Qu.:4.000   3rd Qu.:31.67
##  Max.   :7.00   Max.   :5.00   Max.   :0.4000   Max.   :5.000   Max.   :43.10
##        Y2
##  Min.   :10.90
##  1st Qu.:15.62
##  Median :22.08
##  Mean   :24.59
##  3rd Qu.:33.13
##  Max.   :48.03
```

```
class(energy)
```

```
## [1] "data.frame"
```

```
str(energy)
```

```
## 'data.frame':    768 obs. of  10 variables:
##  $ X1: num  0.98 0.98 0.98 0.98 0.9 0.9 0.9 0.9 0.86 0.86 ...
##  $ X2: num  514 514 514 514 564 ...
##  $ X3: num  294 294 294 294 318 ...
##  $ X4: num  110 110 110 110 122 ...
##  $ X5: num  7 7 7 7 7 7 7 7 7 7 ...
##  $ X6: int  2 3 4 5 2 3 4 5 2 3 ...
##  $ X7: num  0 0 0 0 0 0 0 0 0 0 ...
##  $ X8: int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Y1: num  15.6 15.6 15.6 15.6 20.8 ...
##  $ Y2: num  21.3 21.3 21.3 21.3 28.3 ...
```

```
names(energy)
```

```
##  [1] "X1" "X2" "X3" "X4" "X5" "X6" "X7" "X8" "Y1" "Y2"
```

It is a bit confusing without the feature names, so we will rename the column names to match the data set description

```
colnames(energy) <- c('Relative_Compactness',
                      'Surface_Area',
                      'Wall_Area',
                      'Roof_Area',
                      'Overall_Height',
                      'Orientation',
                      'Glazing_Area',
                      'Glazing_Area_Distribution',
                      'Heating_Load',
                      'Cooling_Load')
```

2.2 cleaning NAs

finding NAs in the data

```
colSums(is.na(energy))
```

```
##      Relative_Compactness              Surface_Area                 Wall_Area
##                         0                         0                         0
##                Roof_Area            Overall_Height               Orientation
##                         0                         0                         0
##              Glazing_Area Glazing_Area_Distribution              Heating_Load
##                         0                         0                         0
##              Cooling_Load
##                         0
```

We do not have any missing data in this data set which is not normal but I'll take it.

2.3 cleaning blank observations

Now we will be checking to see if there are any blank observations

```
colSums(energy == "")
```

```
##      Relative_Compactness              Surface_Area                 Wall_Area
##                         0                         0                         0
##                Roof_Area            Overall_Height               Orientation
##                         0                         0                         0
##              Glazing_Area Glazing_Area_Distribution              Heating_Load
##                         0                         0                         0
##              Cooling_Load
##                         0
```

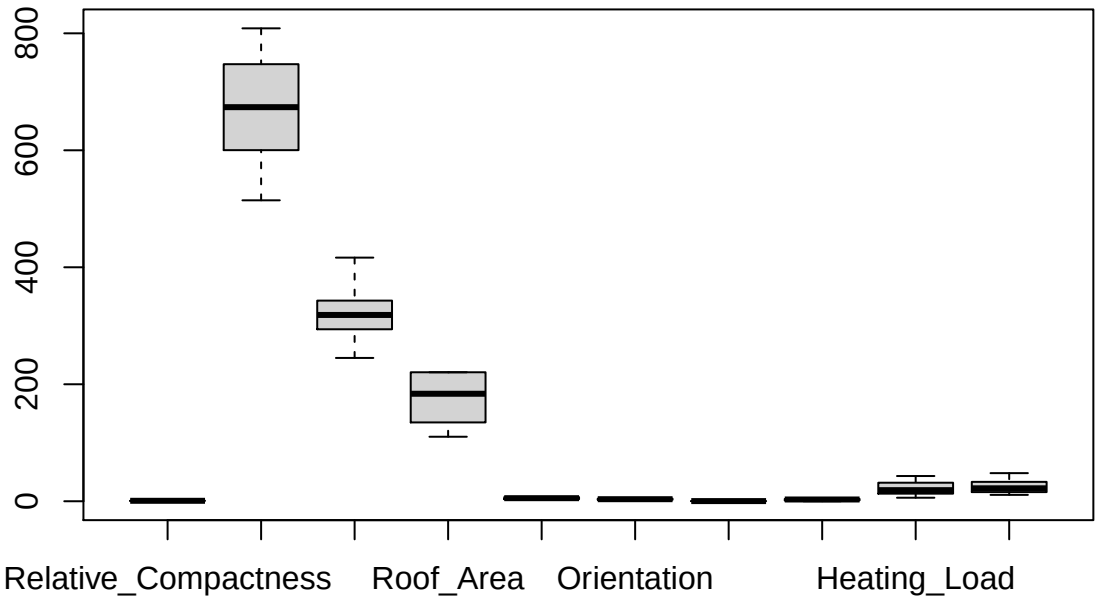
No blank observations so we will move on to visualizing the data.

2.4 scaling the data set

You can see from the summary of the data set that the features have a wide range of observations. This large of a difference could potentially skew our predictions because the models may overvalue features with larger values. In order to reduce that we will scale the data set.

Here is a boxplot of the data set before we scale. You can see the large differences in ranges between a few of the features.

```
boxplot(energy)
```

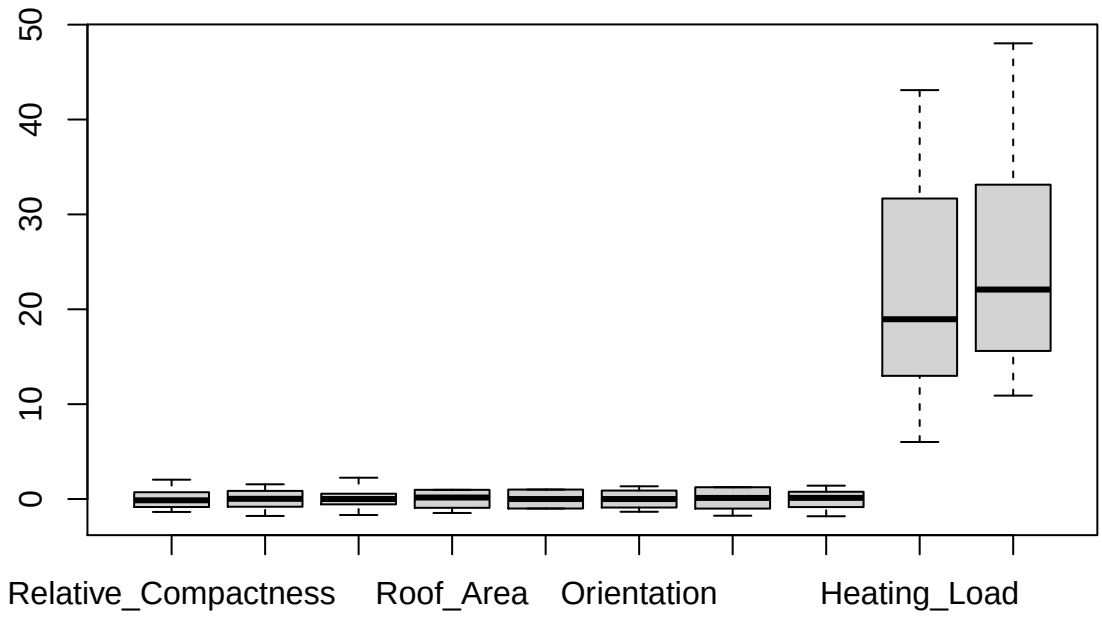


Now let us scale the data set

```
energy[,1:8] <- scale(energy[,1:8])
```

Now let us look at a boxplot of the scaled dataset. We can now see that the features are scaled.

```
boxplot(energy)
```



Let us check the mean of each feature to make sure that the data set is scaled. Means should be 0

```
options(digits = 3)
format(colMeans(energy[,1:8]), scientific = FALSE)
```

```
##      Relative_Compactness              Surface_Area                Wall_Area
##  "-0.000000000000000081"  "-0.000000000000000004429"  " 0.000000000000000000"
##                Roof_Area            Overall_Height               Orientation
##  " 0.000000000000000002035"  " 0.000000000000000000"  " 0.000000000000000000"
##              Glazing_Area Glazing_Area_Distribution
##  " 0.000000000000000001214"  " 0.000000000000000000121"
```

Now let us check the standard deviation. Should be 1

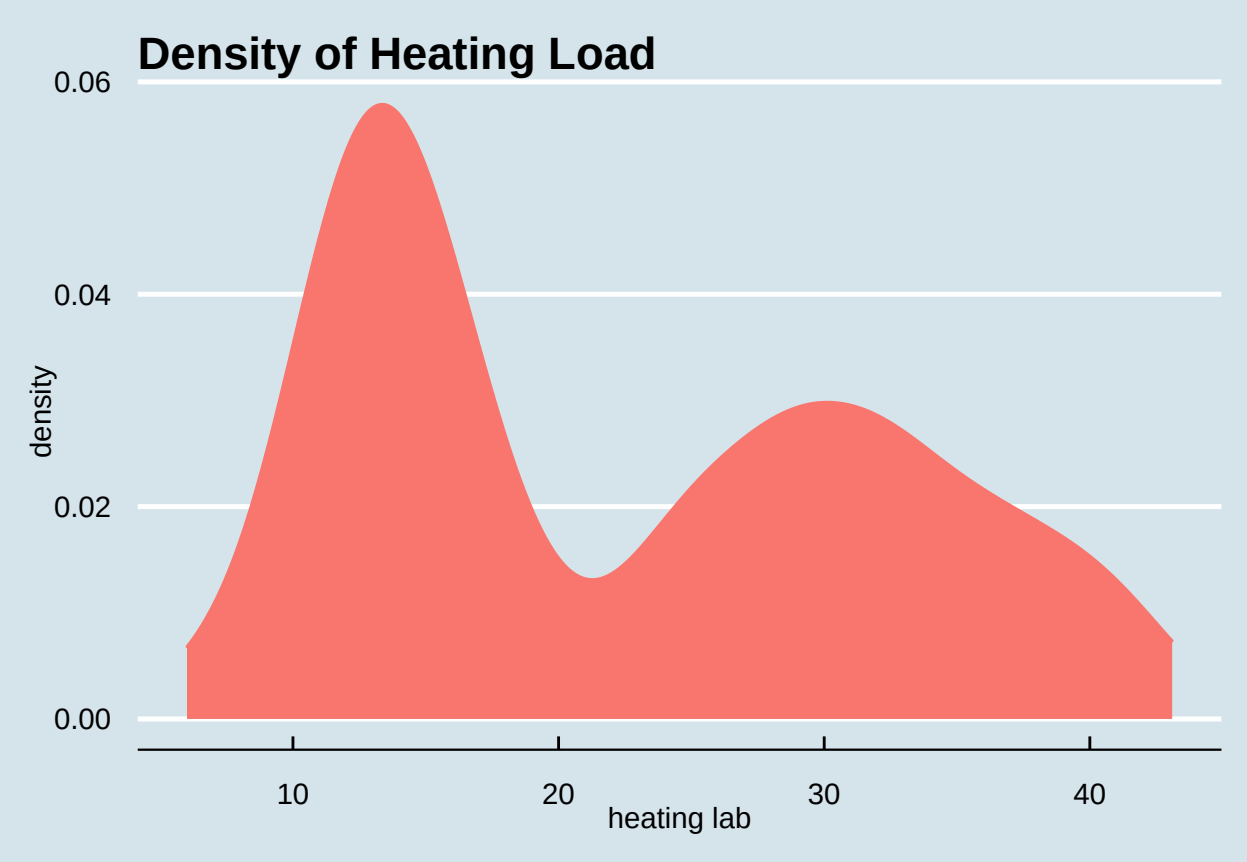
```
energy %>% select(-Heating_Load,-Cooling_Load) %>% summarise_if(is.numeric,sd)
```

```
##   Relative_Compactness Surface_Area Wall_Area Roof_Area Overall_Height
## 1                    1            1         1         1              1
##   Orientation Glazing_Area Glazing_Area_Distribution
## 1           1            1                         1
```

3. Data Visualization

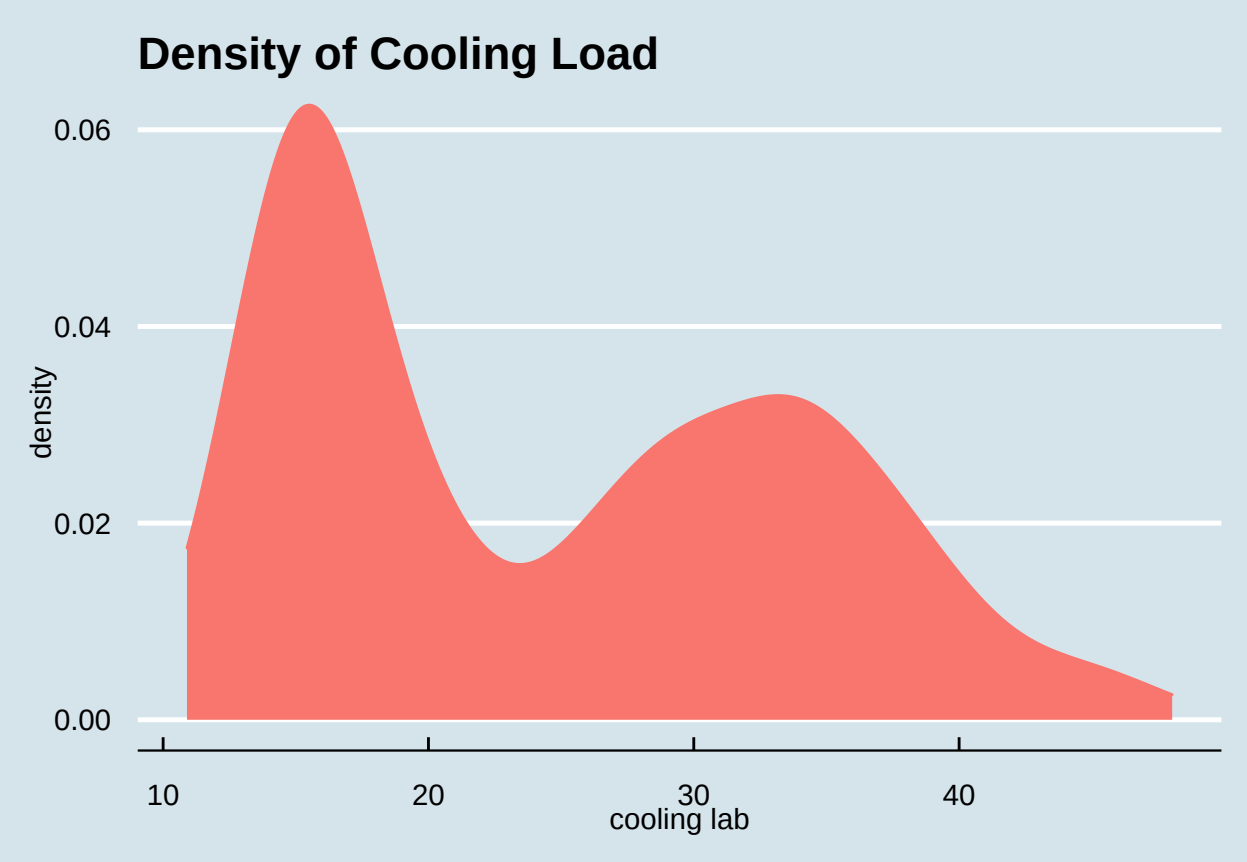
First, let us look at the density of heating load

```
energy %>% ggplot(aes(Heating_Load)) +
  geom_density(aes(fill = "red", color = "red")) +
  xlab("heating lab") +
  ggtitle("Density of Heating Load") +
  theme_economist() +
  theme(legend.position = "none")
```



Second, the density of Cooling load

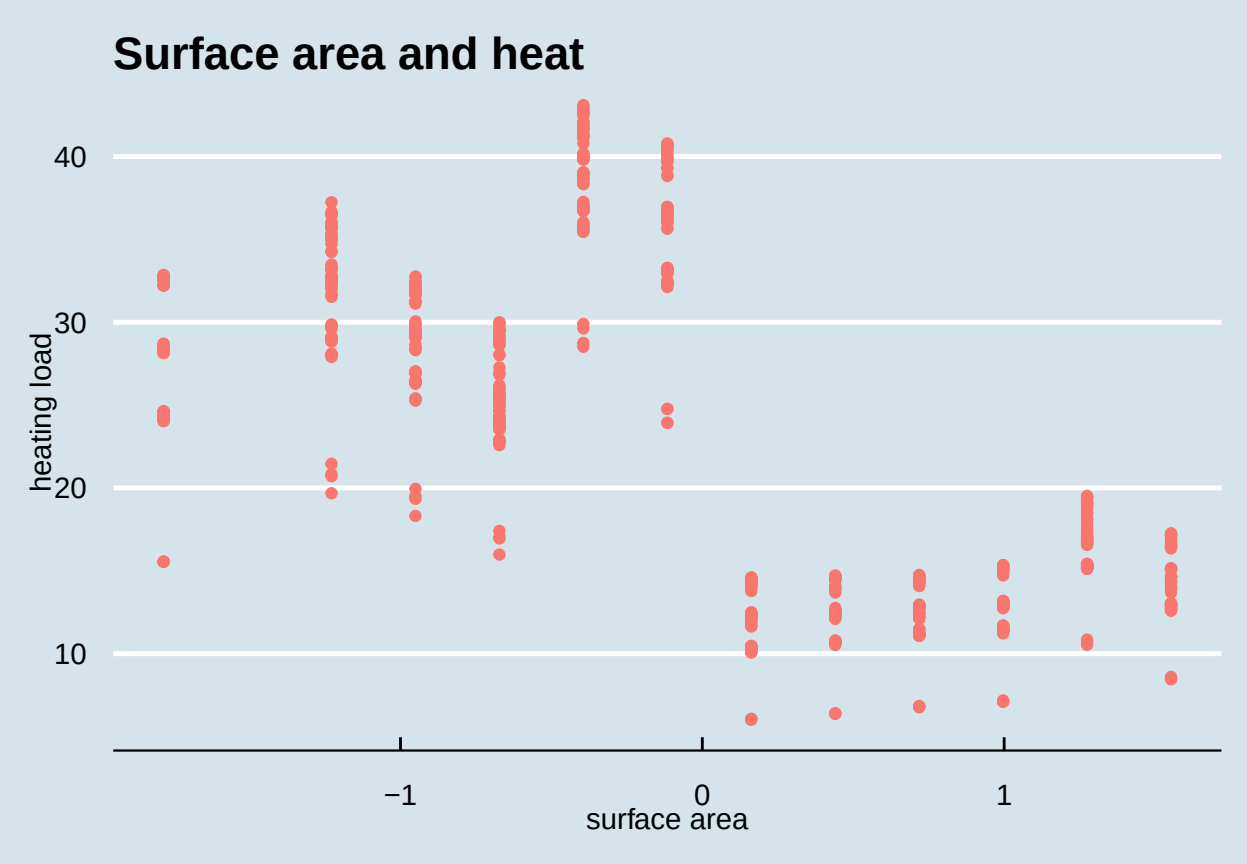
```
energy %>% ggplot(aes(Cooling_Load)) +
  geom_density(aes(fill = "blue", color = "blue")) +
  xlab("cooling lab") +
  ggtitle("Density of Cooling Load") +
  theme_economist() +
  theme(legend.position = "none")
```



Both heating and cooling density look similar

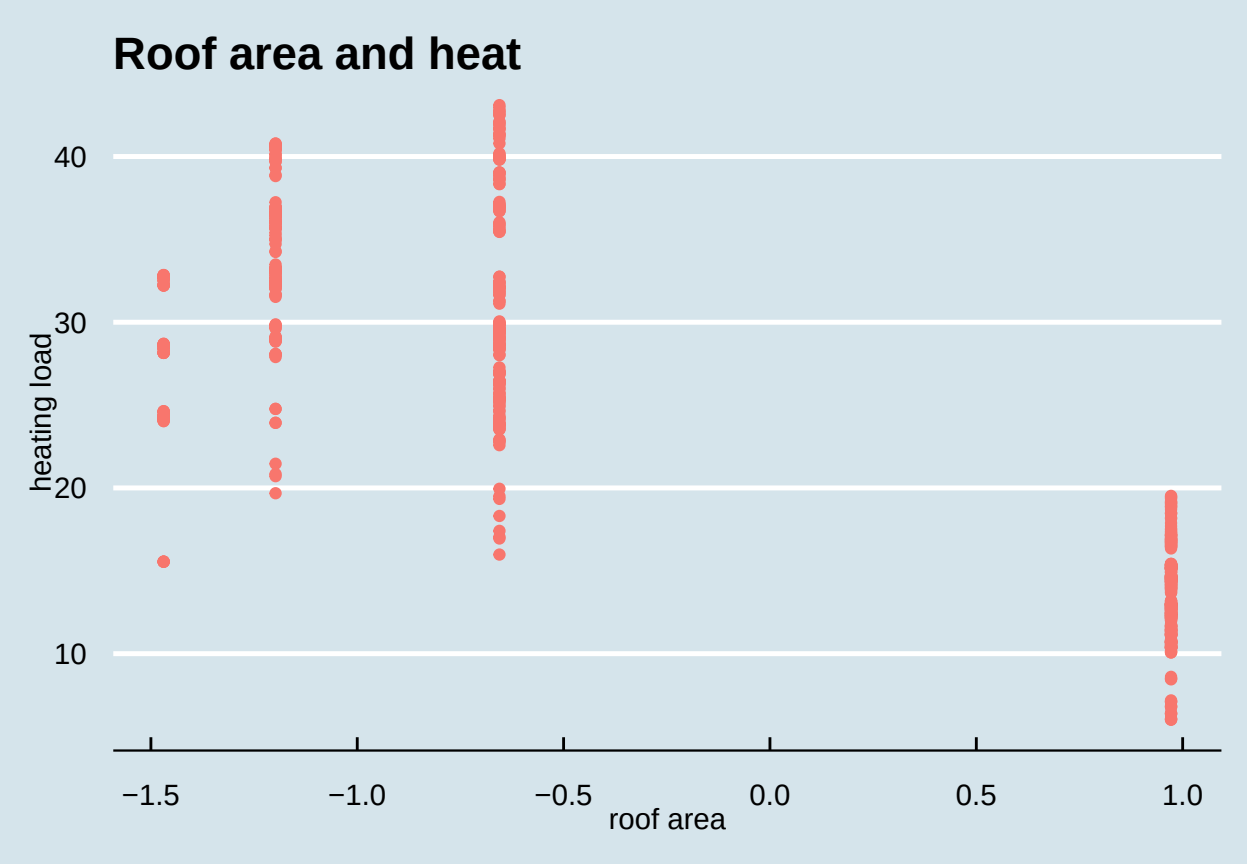
scatter plot of surface area and heating load

```
energy %>% ggplot(aes(Surface_Area,Heating_Load)) +
                    geom_point(aes(color = "red")) +
                    xlab("surface area") +
                    ylab("heating load")+
                    ggtitle("Surface area and heat") +
                    theme_economist() +
                    theme(legend.position = "none")
```



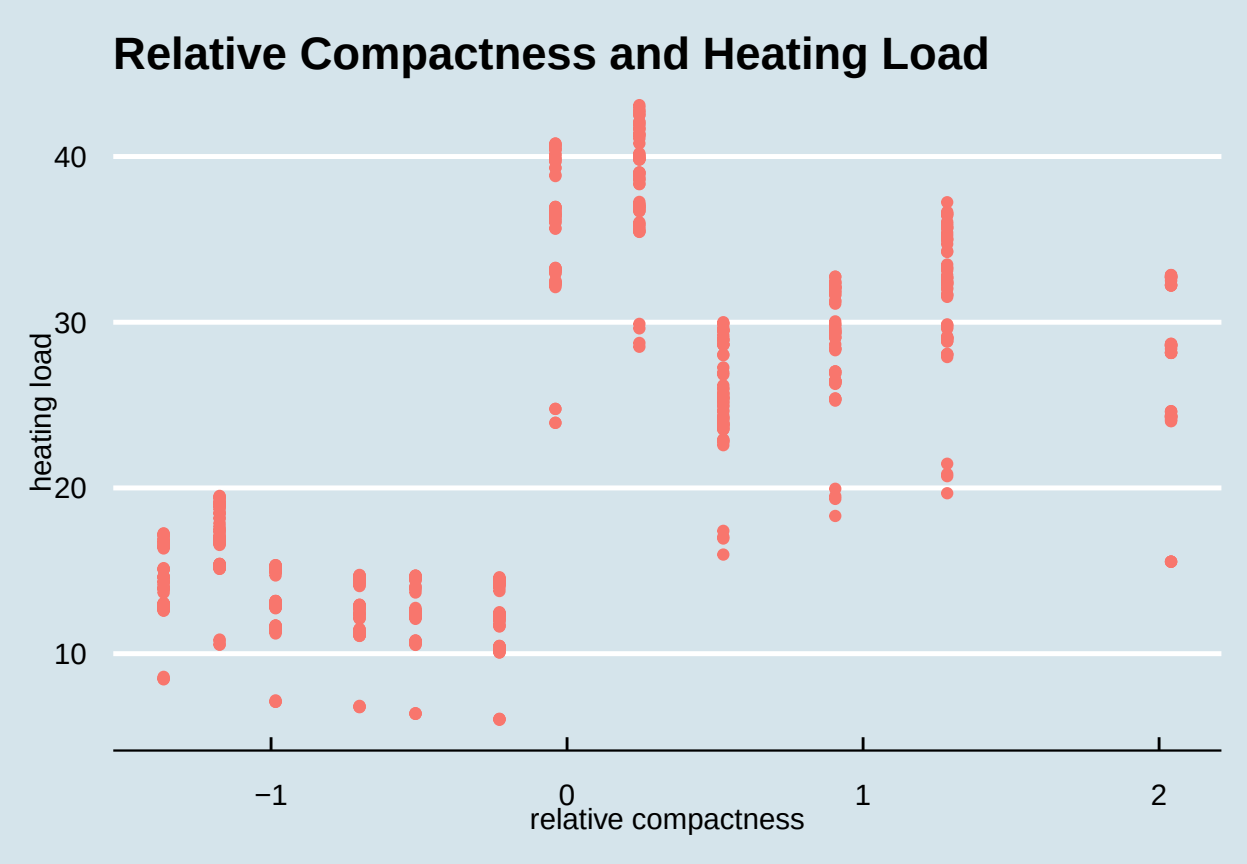
scatter plot of roof area and heating load

```
energy %>% ggplot(aes(Roof_Area,Heating_Load)) +
  geom_point(aes(color = "red")) +
  xlab("roof area") +
  ylab("heating load")+
  ggtitle("Roof area and heat") +
  theme_economist() +
  theme(legend.position = "none")
```



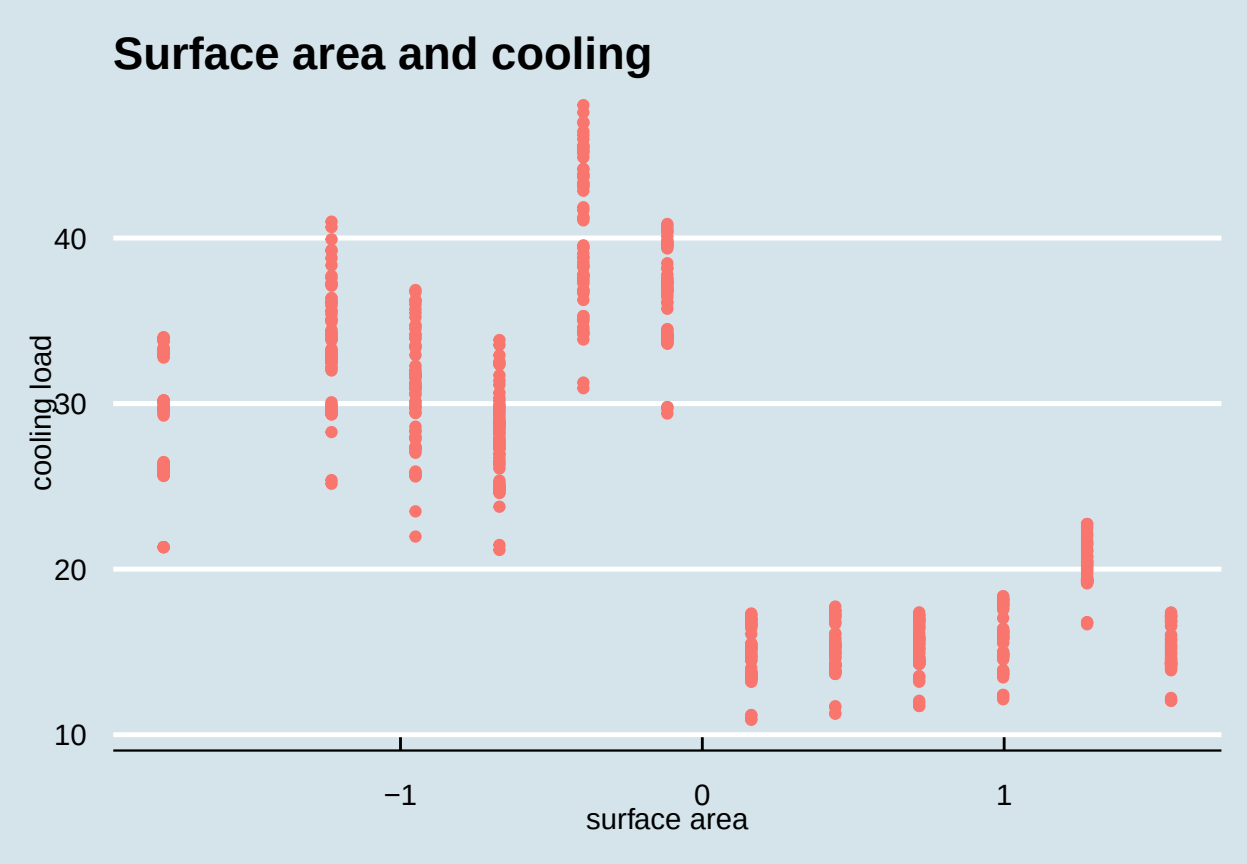
scatter plot of compactness and heating load

```
energy %>% ggplot(aes(Relative_Compactness,Heating_Load)) +
  geom_point(aes(color = "red")) +
  xlab("relative compactness") +
  ylab("heating load") +
  ggtitle("Relative Compactness and Heating Load") +
  theme_economist() +
  theme(legend.position = "none")
```



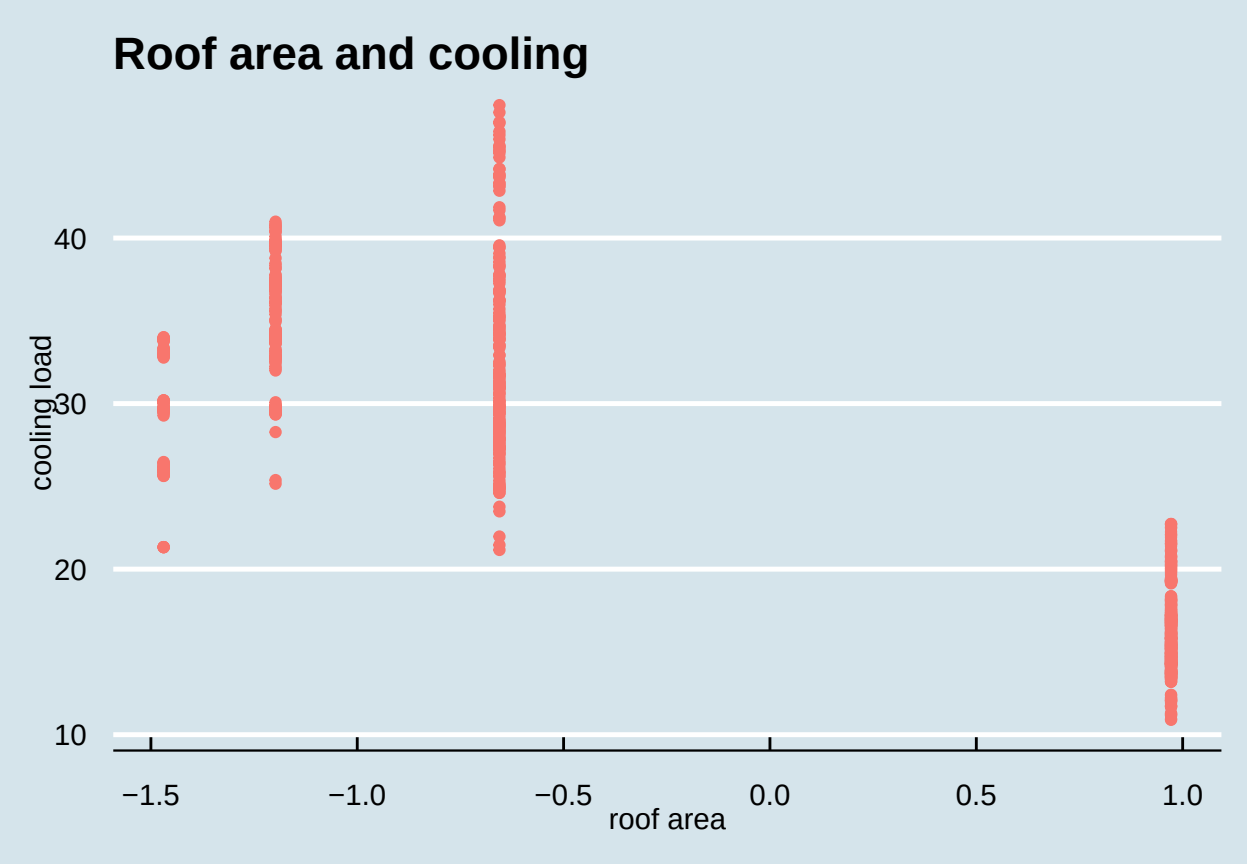
scatter plot of surface area and cooling load

```
energy %>% ggplot(aes(Surface_Area,Cooling_Load)) +
  geom_point(aes(color = "blue")) +
  xlab("surface area") +
  ylab("cooling load")+
  ggtitle("Surface area and cooling") +
  theme_economist() +
  theme(legend.position = "none")
```



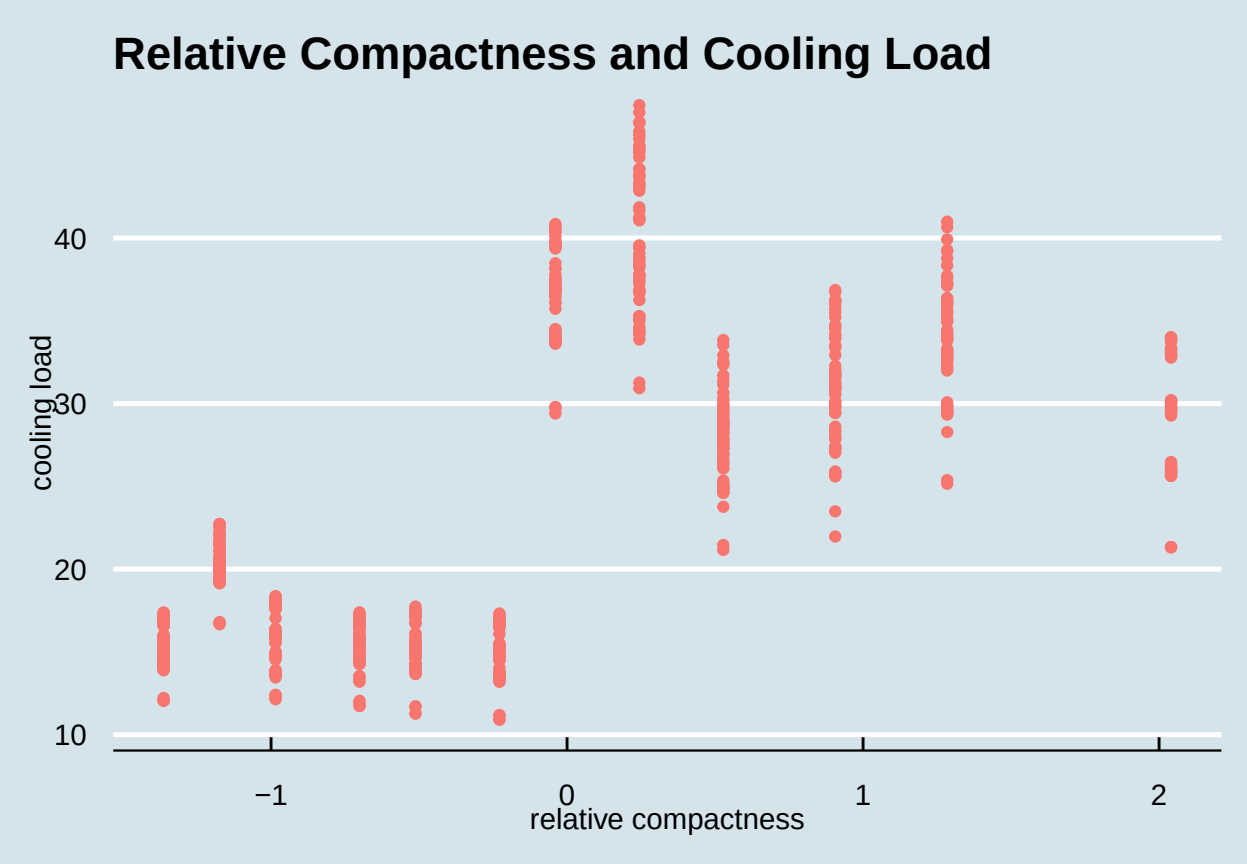
scatter plot of roof area and cooling load

```
energy %>% ggplot(aes(Roof_Area,Cooling_Load)) +
  geom_point(aes(color = "blue")) +
  xlab("roof area") +
  ylab("cooling load")+
  ggtitle("Roof area and cooling") +
  theme_economist() +
  theme(legend.position = "none")
```



scatter plot of compactness and cooling load

```
energy %>% ggplot(aes(Relative_Compactness,Cooling_Load)) +
  geom_point(aes(color = "blue")) +
  xlab("relative compactness") +
  ylab("cooling load") +
  ggtitle("Relative Compactness and Cooling Load") +
  theme_economist() +
  theme(legend.position = "none")
```



4. Models

4.1 train and test sets

We are going to split the data into a training set and a test set. The training set will be 80% of the total data set and the test will be 20%. We will be using the training set to train our regression models and then we will test those models on new data with in this case will be the test set that make up the remaining 20% of the data.

```
#setting seed
set.seed(1, sample.kind = "Rounding")

#splitting data into test and train data sets
test_index <- createDataPartition(energy$Heating_Load, times = 1, p = 0.2, list = FALSE)
test <- energy[test_index,]
train <- energy[-test_index,]
```

checking to see if the test and train have similar outcomes

```
mean(train$Heating_Load)
```

```
## [1] 22.3
```

```
mean(test$Heating_Load)
```

```
## [1] 22.4
```

Will be using k-fold cross validation on all the algorithms

```
#creating the k-fold parameters, k is 10
set.seed(7, sample.kind = "Rounding")
control <- trainControl(method = "cv", number = 10, p = .9)
```

4.2 Linear Regression

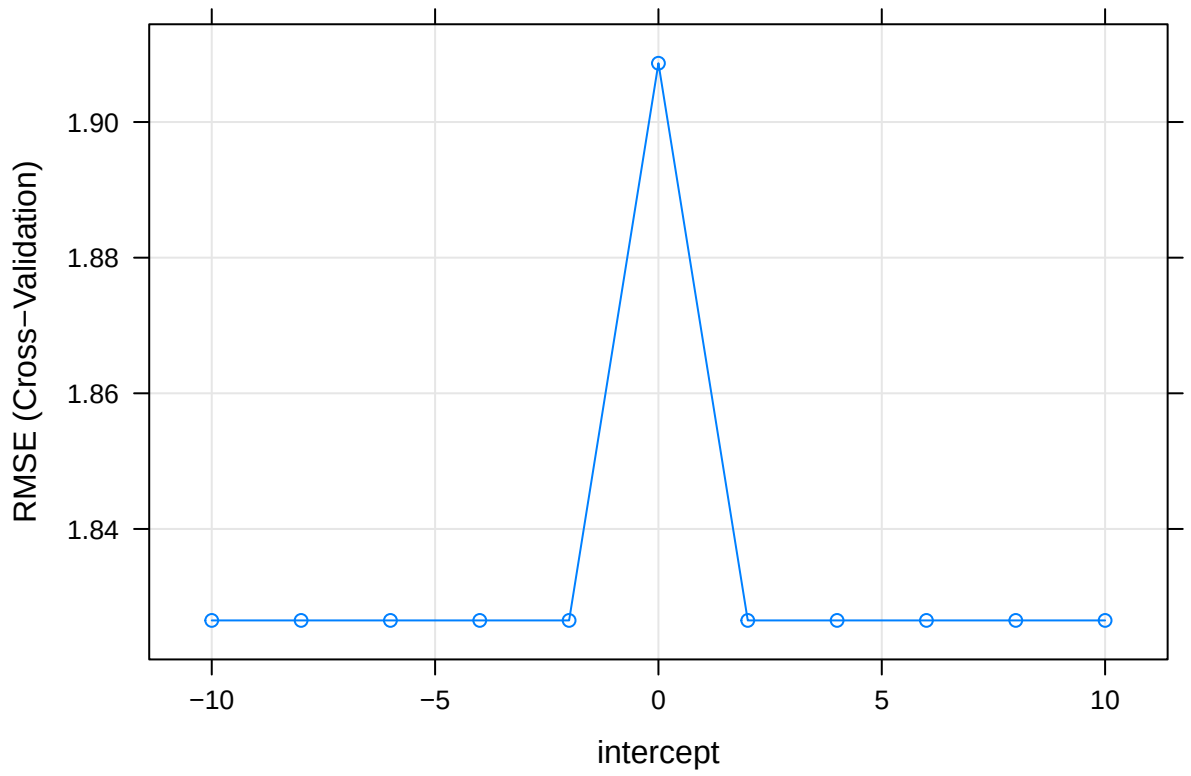
Predicting the heating load using linear regression

```
#training the model using train set
set.seed(9, sample.kind = "Rounding")
train_lm <- train(Heating_Load ~ .,
                  data = train,
                  method = "lm",
                  tuneGrid = data.frame(intercept = seq(-10,10,2)),
                  trControl = control)

#viewing training results
train_lm
```

```
## Linear Regression
##
## 612 samples
##   9 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 552, 549, 551, 551, 551, 551, ...
## Resampling results across tuning parameters:
##
##   intercept  RMSE  Rsquared  MAE
##   -10        1.83  0.969     1.26
##    -8        1.83  0.969     1.26
##    -6        1.83  0.969     1.26
##    -4        1.83  0.969     1.26
##    -2        1.83  0.969     1.26
##     0        1.91  0.967     1.38
##     2        1.83  0.969     1.26
##     4        1.83  0.969     1.26
##     6        1.83  0.969     1.26
##     8        1.83  0.969     1.26
##    10        1.83  0.969     1.26
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was intercept = -10.
```

plotting training results

```
plot(train_lm)
```



creating predictions

```
lm_preds_hl <- predict(train_lm, test)
```

calculating the RMSE for the linear regression model

```
lm_rmse_hl <- RMSE(lm_preds_hl,test$Heating_Load)
```

Now we will apply linear regression to get a prediction for the cooling load

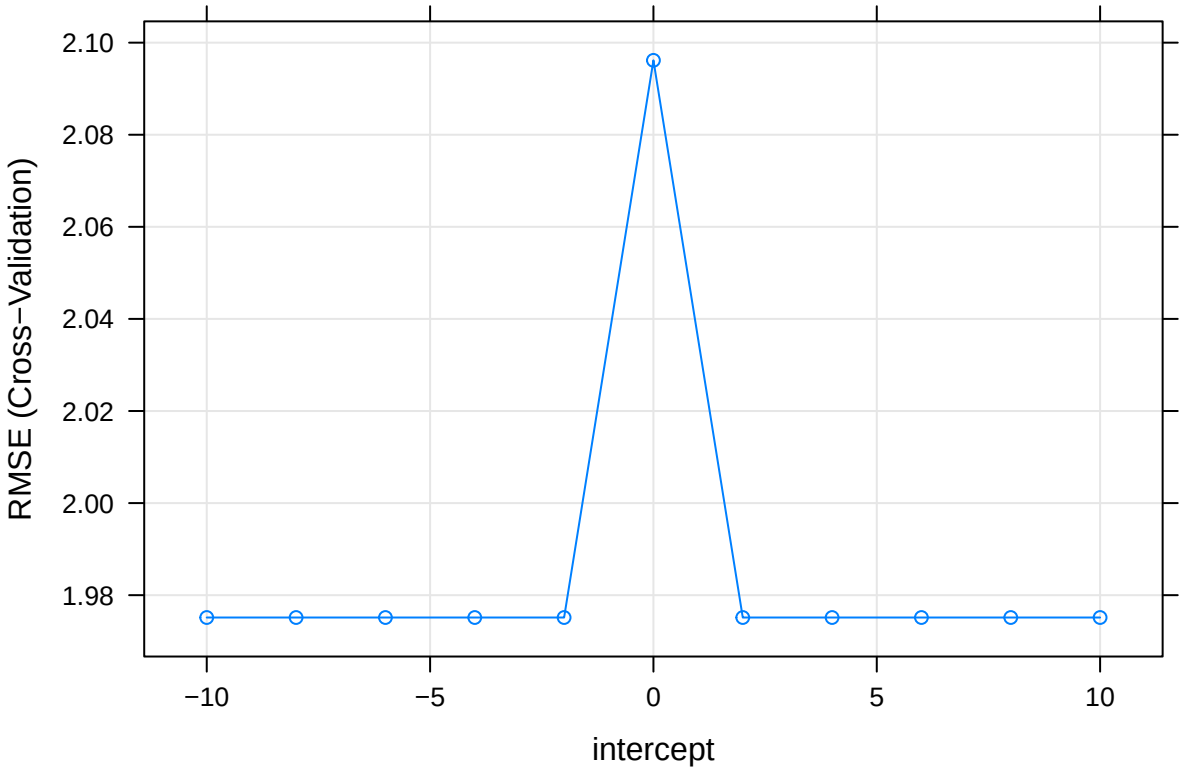
training the model using train set

```
set.seed(9, sample.kind = "Rounding")
train_lm <- train(Cooling_Load ~ .,
                  data = train,
                  method = "lm",
                  tuneGrid = data.frame(intercept = seq(-10,10,2)),
                  trControl = control)

#viewing training results
train_lm
```

```
## Linear Regression
##
## 612 samples
##   9 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 552, 549, 551, 551, 551, 551, ...
## Resampling results across tuning parameters:
##
##   intercept  RMSE  Rsquared  MAE
##   -10        1.98  0.959     1.45
##    -8        1.98  0.959     1.45
##    -6        1.98  0.959     1.45
##    -4        1.98  0.959     1.45
##    -2        1.98  0.959     1.45
##     0        2.10  0.955     1.52
##     2        1.98  0.959     1.45
##     4        1.98  0.959     1.45
##     6        1.98  0.959     1.45
##     8        1.98  0.959     1.45
##    10        1.98  0.959     1.45
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was intercept = -10.
```

plotting training results

```
plot(train_lm)
```



creating predictions

```
lm_preds_cl <- predict(train_lm, test)
```

calculating the RMSE for the linear regression model

```
lm_rmse_cl <- RMSE(lm_preds_cl,test$Cooling_Load)
```

4.3 ridge regression

training the model using train set

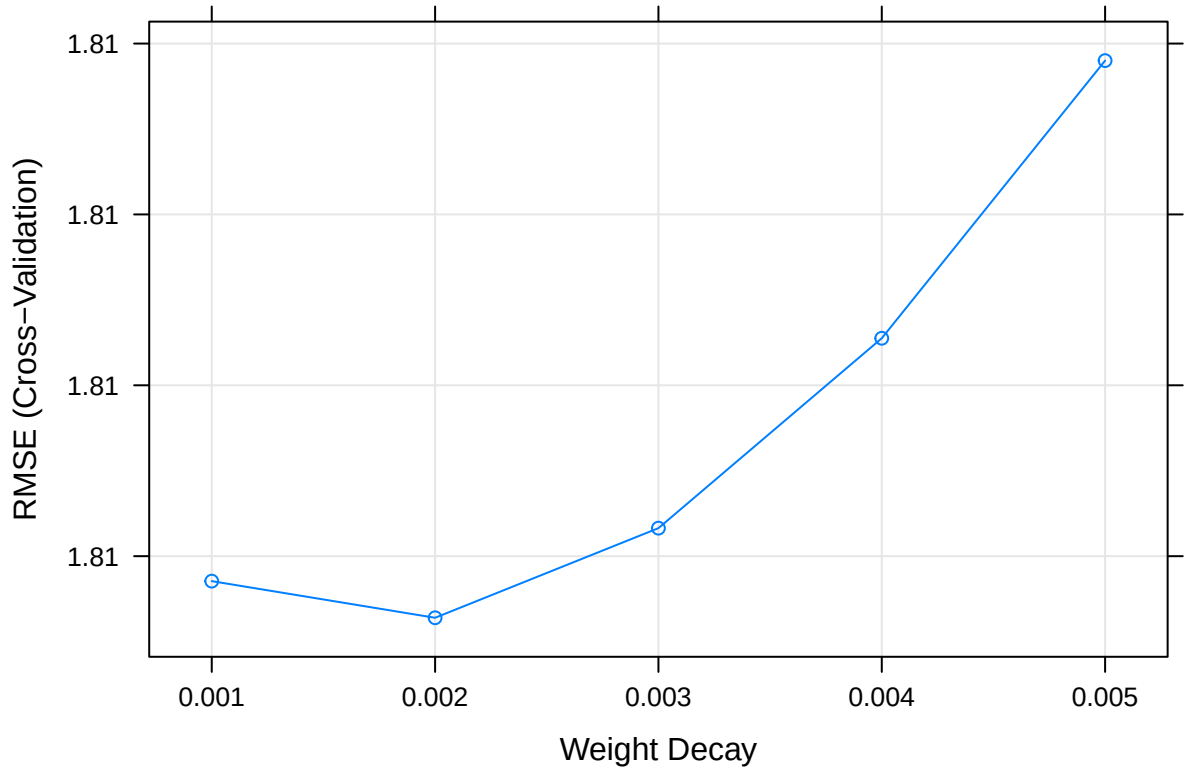
```
set.seed(10, sample.kind = "Rounding")
train_ridge <- train(Heating_Load ~ .,
                     data = train,
                     method = "ridge",
                     tuneGrid = data.frame(lambda = seq(.001,.005,.001)),
                     trControl = control)

#viewing training results
train_ridge
```

```
## Ridge Regression
```

```
##
## 612 samples
##   9 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 550, 552, 551, 552, 549, 551, ...
## Resampling results across tuning parameters:
##
##   lambda  RMSE  Rsquared  MAE
##   0.001   1.81  0.969     1.25
##   0.002   1.81  0.969     1.25
##   0.003   1.81  0.969     1.25
##   0.004   1.81  0.969     1.25
##   0.005   1.81  0.969     1.24
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was lambda = 0.002.
```

plotting training results

```
plot(train_ridge)
```



creating predictions

```
ridge_preds_hl <- predict(train_ridge,test)
```

creating RMSE for ridge regression model

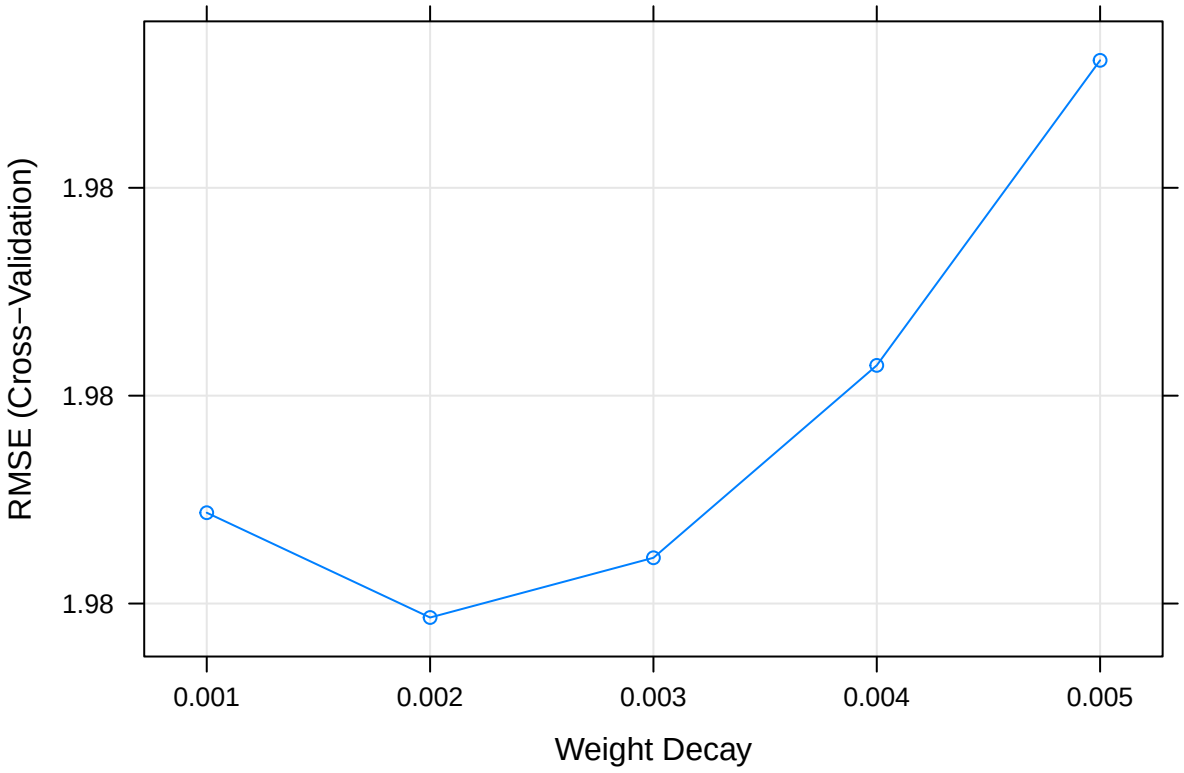
```
ridge_rmse_hl <- RMSE(ridge_preds_hl, test$Heating_Load)
```

```
#training the model using train set
set.seed(10, sample.kind = "Rounding")
train_ridge <- train(Cooling_Load ~ .,
                     data = train,
                     method = "ridge",
                     tuneGrid = data.frame(lambda = seq(.001,.005,.001)),
                     trControl = control)

#viewing training results
train_ridge
```

```
## Ridge Regression
##
## 612 samples
##   9 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 550, 552, 551, 552, 549, 551, ...
## Resampling results across tuning parameters:
##
##   lambda  RMSE  Rsquared  MAE
##   0.001   1.98  0.959     1.45
##   0.002   1.98  0.959     1.45
##   0.003   1.98  0.959     1.45
##   0.004   1.98  0.959     1.45
##   0.005   1.98  0.959     1.45
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was lambda = 0.002.
```

plotting training results

```
plot(train_ridge)
```



creating predictions

```
ridge_preds_cl <- predict(train_ridge,test)
```

creating RMSE for ridge regression model

```
ridge_rmse_cl <- RMSE(ridge_preds_cl, test$Cooling_Load)
```

4.4 Random Forest

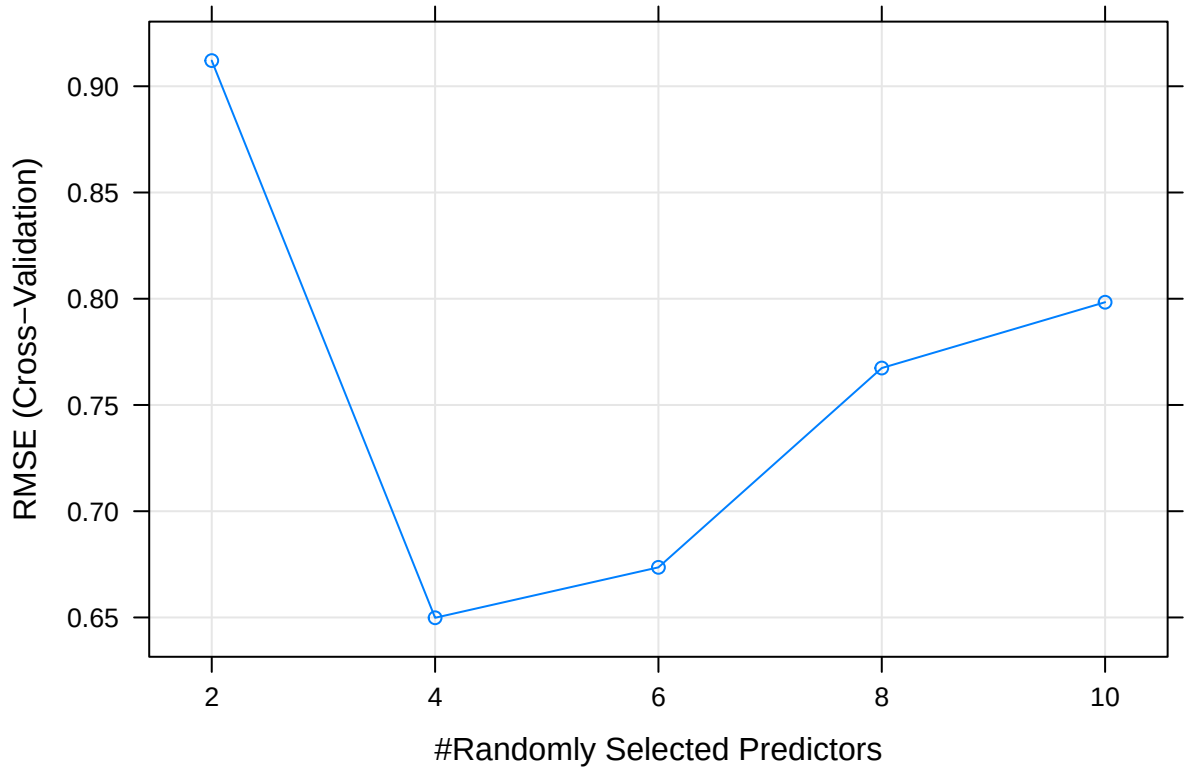
```
#training the model using training set
set.seed(12, sample.kind = "Rounding")
train_rf <- train(Heating_Load ~ .,
                  data = train,
                  method = "rf",
                  tuneGrid = data.frame(mtry = seq(2,10,2)),
                  trControl = control)

#veiwing training results
train_rf
```

```
## Random Forest
```

```
##
## 612 samples
##   9 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 548, 551, 551, 551, 552, 552, ...
## Resampling results across tuning parameters:
##
##   mtry  RMSE   Rsquared  MAE
##    2    0.912  0.992     0.663
##    4    0.650  0.996     0.389
##    6    0.674  0.995     0.380
##    8    0.767  0.994     0.410
##   10    0.798  0.993     0.424
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was mtry = 4.
```

plotting training results

```
plot(train_rf)
```



creating predictions

```
rf_preds_hl <- predict(train_rf,test)
```

creating RMSE for random forest model

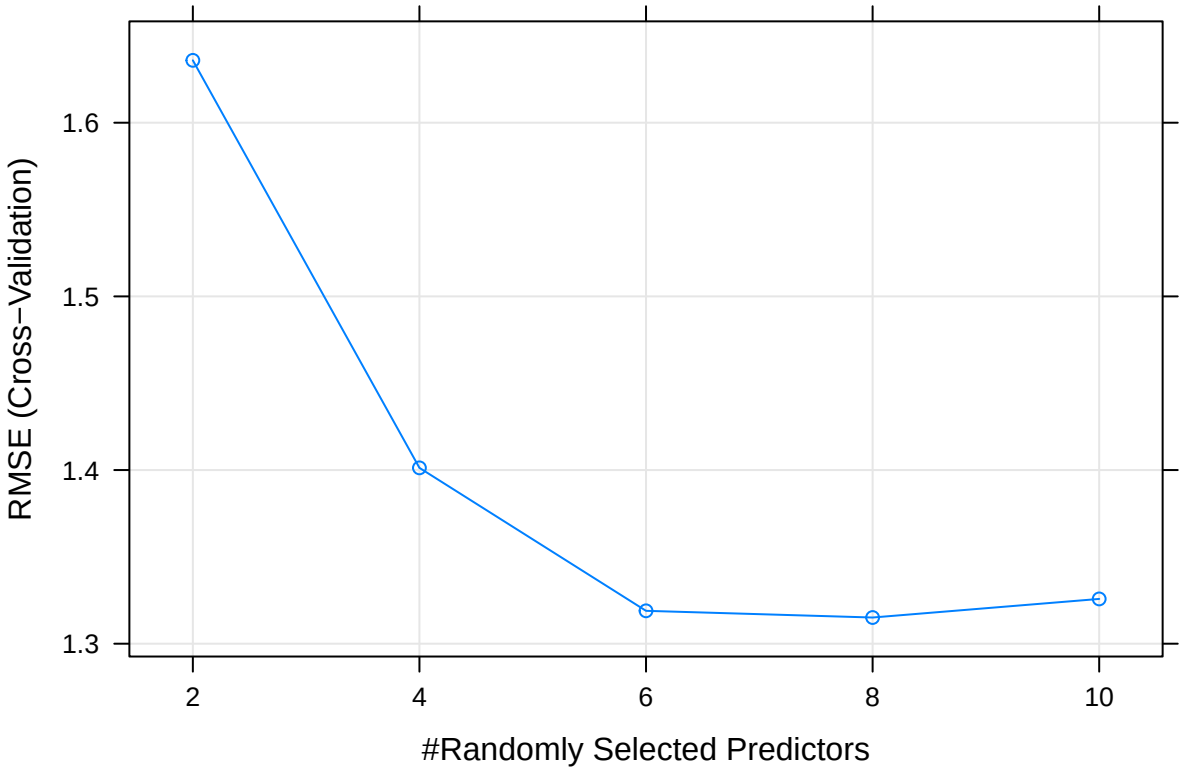
```
rf_rmse_hl <- RMSE(rf_preds_hl,test$Heating_Load)
```

```
#training the model using training set
set.seed(12, sample.kind = "Rounding")
train_rf <- train(Cooling_Load ~ .,
                  data = train,
                  method = "rf",
                  tuneGrid = data.frame(mtry = seq(2,10,2)),
                  trControl = control)

#viewing training results
train_rf
```

```
## Random Forest
##
## 612 samples
##   9 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 548, 551, 551, 551, 552, 552, ...
## Resampling results across tuning parameters:
##
##   mtry  RMSE  Rsquared  MAE
##    2    1.64  0.972     1.110
##    4    1.40  0.979     0.864
##    6    1.32  0.981     0.780
##    8    1.32  0.982     0.759
##   10    1.33  0.981     0.763
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was mtry = 8.
```

plotting training results

```
plot(train_rf)
```



creating predictions

```
rf_preds_cl <- predict(train_rf,test)
```

creating RMSE for random forest model

```
rf_rmse_cl <- RMSE(rf_preds_cl,test$Cooling_Load)
```

4.5 Ensemble

```
heating_preds <- data.frame("lm" = lm_preds_hl,
                            "ridge" = ridge_preds_hl,
                            "rf" = rf_preds_hl)

ensemble_preds_hl <- rowMeans(heating_preds)

heating_preds$ensemble <- ensemble_preds_hl

ensemble_rmse_hl <- RMSE(ensemble_preds_hl,test$Heating_Load)
```

Ensemble for cooling load

```
cooling_preds <- data.frame("lm" = lm_preds_cl,
                            "ridge" = ridge_preds_cl,
                            "rf" = rf_preds_cl)

ensemble_preds_cl <- rowMeans(cooling_preds)

cooling_preds$ensemble <- ensemble_preds_cl

ensemble_rmse_cl <- RMSE(ensemble_preds_cl,test$Cooling_Load)
```

5. Results

5.1 table of results

```
options(digits = 3)
results <- data.frame(Model = c("Linear Regression",
                                "Ridge Regression",
                                "Random Forest",
                                "Ensemble"),
                      Heating = c(lm_rmse_hl,
                                  ridge_rmse_hl,
                                  rf_rmse_hl,
                                  ensemble_rmse_hl),
                      Cooling =  c(lm_rmse_cl,
                                  ridge_rmse_cl,
                                  rf_rmse_cl,
                                  ensemble_rmse_cl))

kable(results)
```

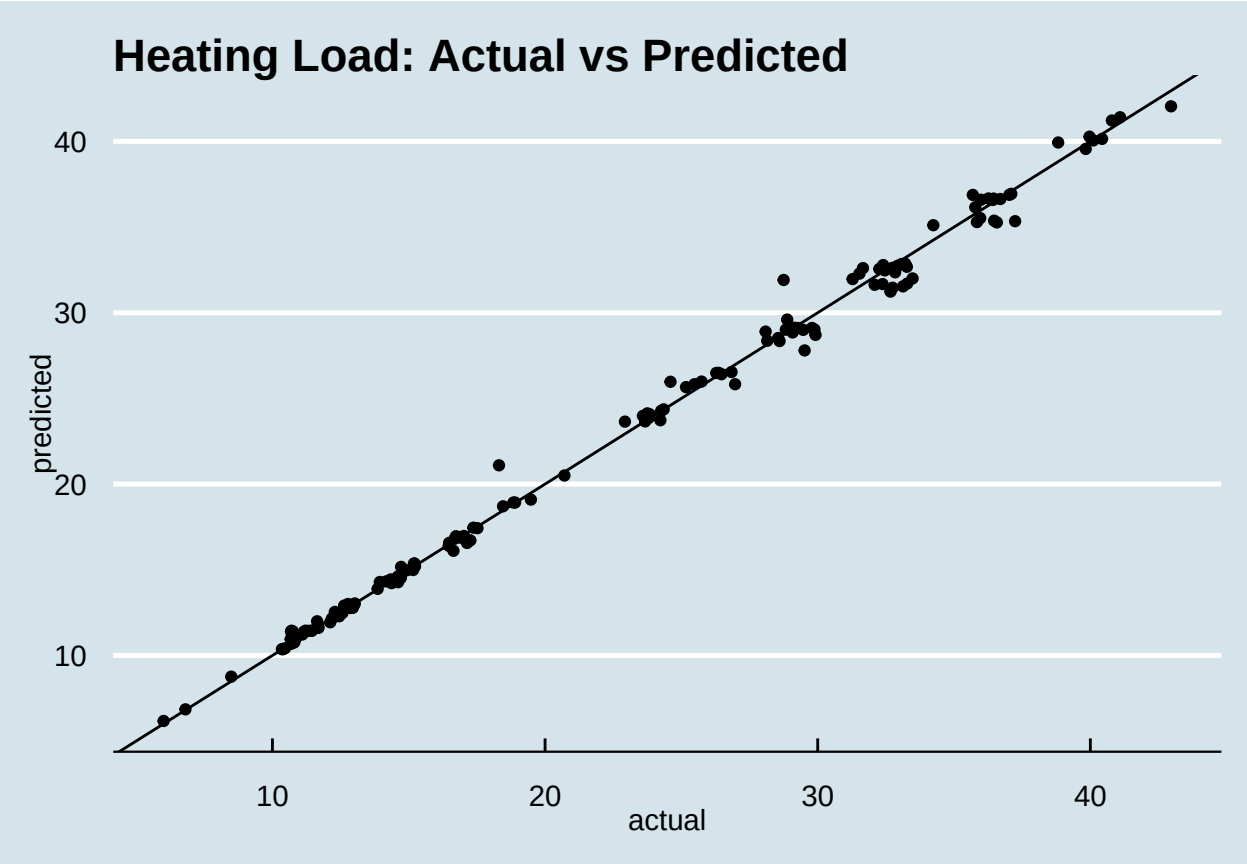
Model	Heating	Cooling
Linear Regression	1.675	1.86
Ridge Regression	1.673	1.87
Random Forest	0.631	1.26
Ensemble	1.246	1.56

5.2 plot of results

For the heating load, our best model was random forest. Here is a plot of the random forest predictions against the actual results.

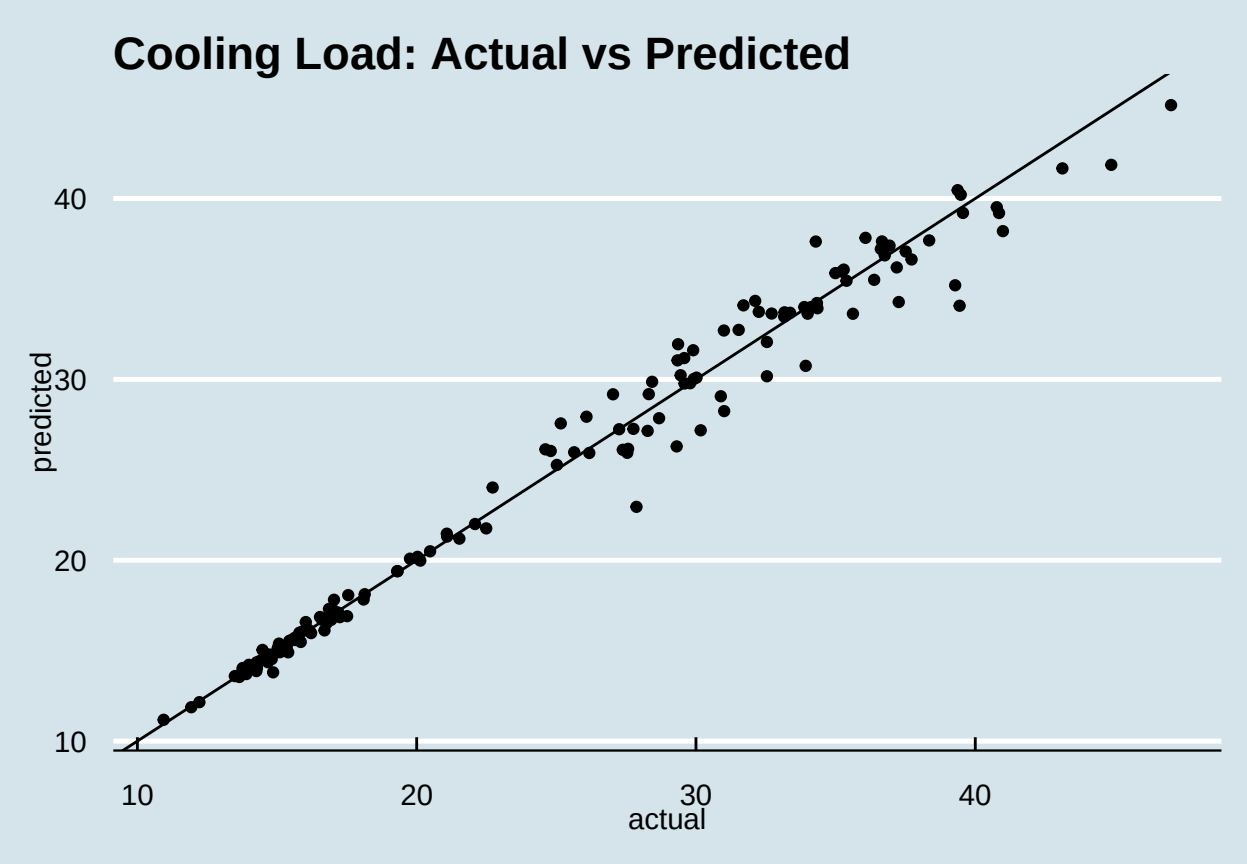
```
data.frame(actual = test$Heating_Load,
           predicted = rf_preds_hl) %>%
  ggplot(aes(actual,predicted)) +
  geom_point() +
  geom_abline(intercept = 0, slope = 1) +
  xlab("actual") +
  ylab("predicted") +
```

```
  ggtitle("Heating Load: Actual vs Predicted") +
  theme_economist()
```



For the cooling load, our best model was also random forest. Here is a plot of the random forest predictions against the actual results.

```
data.frame(actual = test$Cooling_Load,
           predicted = rf_preds_cl) %>%
  ggplot(aes(actual,predicted)) +
  geom_point() +
  geom_abline(intercept = 0, slope = 1) +
  xlab("actual") +
  ylab("predicted") +
  ggtitle("Cooling Load: Actual vs Predicted") +
  theme_economist()
```



5.3 brief thoughts on results

I was not surprised that the linear regression model had the highest RMSE. Although the linear regression model is a powerful one, its weakness is when there is variety in the data that doesn't quite line up linearly. I did think this was a good starting point because it gave us a baseline of the relation between the features and the outputs.

I really didn't know what to expect from the ridge regression model. I was surprised that it wasn't that much better than the linear model. Ridge regression models do well when the features are highly correlated, I was thinking that might be the case so I wanted to try it out and see the results.

I was not surprised that the random forest model was the best performing model. It is a very powerful model that seems to do well for both regression and classification models. I found it very interesting that the model was able to better predict the heating load over the cooling load. The model's prediction was better for cooling loads under 25 but struggled a bit when it was over 25. My hypothesis is that cooling load is more difficult to predict because of sun beating down on the beating. This might also explain why the cooling loads are higher than the heating loads. It might be more difficult to maintain room temperature if the sunlight is countering the cooling affects.

The ensemble landed up having a RMSE that landed in the middle of the results. This made sense to me sense we were taking the mean from the three models. I think the ensemble would do really well when applied to a large data set due to its middle of the road approach.

6. Conclusion

6.1 summary

We were able to predict the heating and cooling load of buildings by using a dataset of building features and heating and cooling loads. We used supervised machine learning to create predictions. We had a total of 4 regression models: linear regression, ridge regression, random forest, an ensemble of first 3 models. Random forest was our best model for predicting heating load with a RMSE of .63. The random forest model was also the best model for predicting the cooling load with a RMSE of 1.26.

6.2 limitations

The limitation of this model is the size of the dataset. We are only looking at a sample of 768 buildings. With more data with more variety of buildings I think we could see a more robust model.

I would also like to see a model with more features. I think there was a lot of information that was missing from the dataset. Where are these buildings located? Do they experience the same climate, etc...

6.3 next steps

The next step would be to use this model to predict the heating and cooling load of the next home or building that you plan on purchasing!