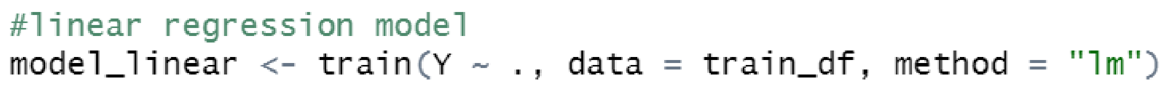
Predictive Model

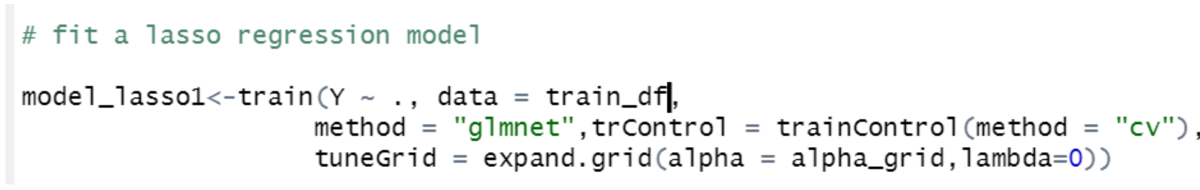
Fitting Models

- Linear Regression model



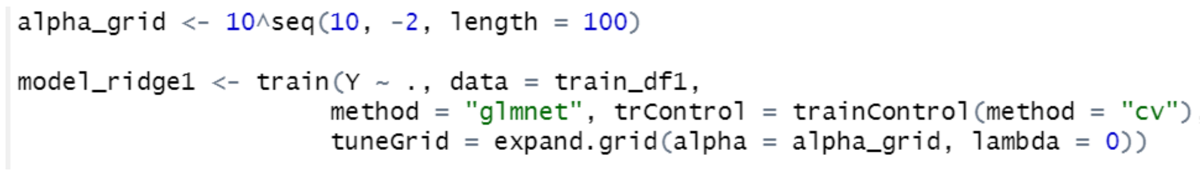
```
#linear regression model
model_linear <- train(Y ~ ., data = train_df, method = "lm")
```

- Lasso Model



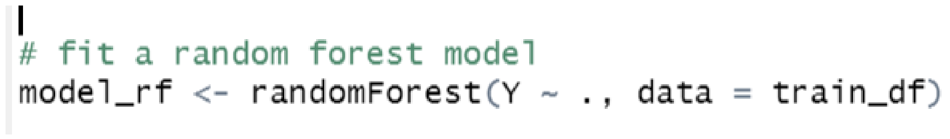
```
# fit a lasso regression model

model_lasso1<-train(Y ~ ., data = train_df,
                       method = "glmnet",trControl = trainControl(method = "cv"),
                       tuneGrid = expand.grid(alpha = alpha_grid,lambda=0))
```

- Ridge Model



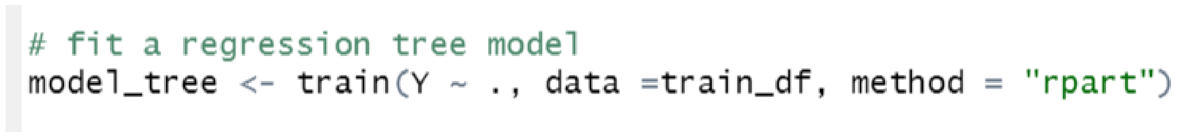
```
alpha_grid <- 10^seq(10, -2, length = 100)

model_ridge1 <- train(Y ~ ., data = train_df1,
                        method = "glmnet", trControl = trainControl(method = "cv")
                        tuneGrid = expand.grid(alpha = alpha_grid, lambda = 0))
```

- Random Forest



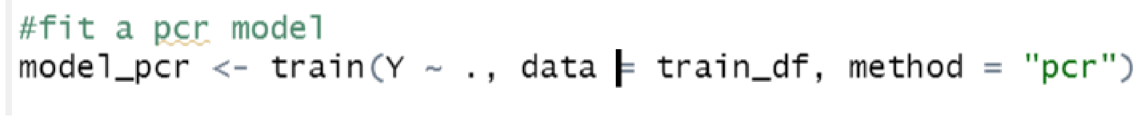
```
# fit a random forest model
model_rf <- randomForest(Y ~ ., data = train_df)
```

- Regression Tree Model



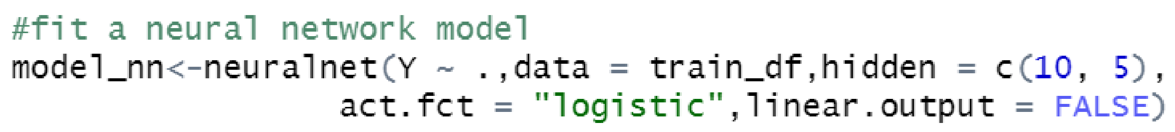
```
# fit a regression tree model
model_tree <- train(Y ~ ., data =train_df, method = "rpart")
```

- PCR model



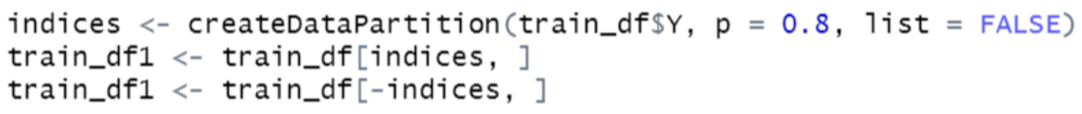
```
#fit a pcr model
model_pcr <- train(Y ~ ., data = train_df, method = "pcr")
```

- Neural Network

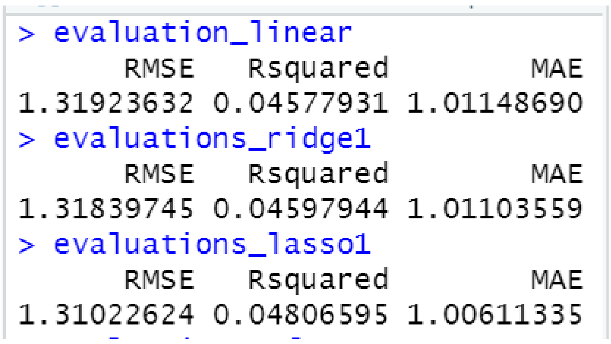


```
#fit a neural network model
model_nn<-neuralnet(Y ~ .,data = train_df,hidden = c(10, 5),
                      act.fct = "logistic",linear.output = FALSE)
```

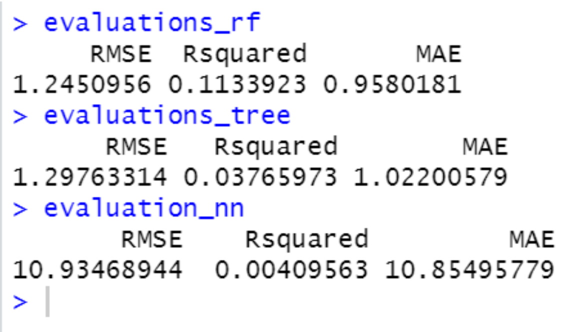
Evaluating Models



```
indices <- createDataPartition(train_df$Y, p = 0.8, list = FALSE)
train_df1 <- train_df[indices, ]
train_df1 <- train_df[-indices, ]
```



```
> evaluation_linear
      RMSE   Rsquared        MAE
1.31923632 0.04577931 1.01148690
> evaluations_ridge1
      RMSE   Rsquared        MAE
1.31839745 0.04597944 1.01103559
> evaluations_lasso1
      RMSE   Rsquared        MAE
1.31022624 0.04806595 1.00611335
```

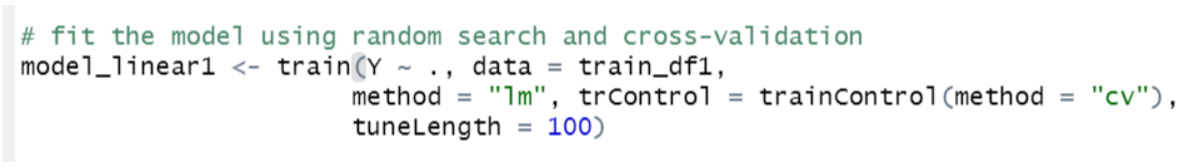


```
> evaluations_rf
     RMSE  Rsquared       MAE
1.2450956 0.1133923 0.9580181
> evaluations_tree
      RMSE   Rsquared        MAE
1.29763314 0.03765973 1.02200579
> evaluation_nn
       RMSE    Rsquared         MAE
10.93468944  0.00409563 10.85495779
>
```

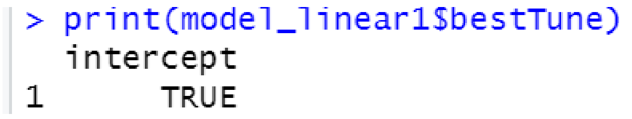
To evaluate all of my models, I split the training set into training and testing set with 80:20 ratio. Then I evaluated all my models using postReSample which gave me the R Squared, RMSE and MAE.

- RMSE
  - RMSE is Root Mean Square Error. Its shows the standard deviation of residuals for the data.
  - For the best prediction, you generally choose the least RMSE which in this case is random Forest model.
  - The highest is neural network so for our dataset, that would be the worst choice if we took the RMSE.
- R Squared
  - R Squared is the coefficient of determination. It tells you how good of a fit can be described by the model between dependent and independent variables.
  - The higher the R-squared, the better the model fit so for this dataset, random Forest is the best with 11.34%
  - The lowest is again neural network with 0.4%
- MAE
  - MAE is the measure of average error between the prediction and actual value which in this case is the Y values of 20% of the training set.
  - The lower the MAE, the better the model and again random Forest has the least MAE compared to all the other models.
  - The highest is for neural network.

Tuning Methods

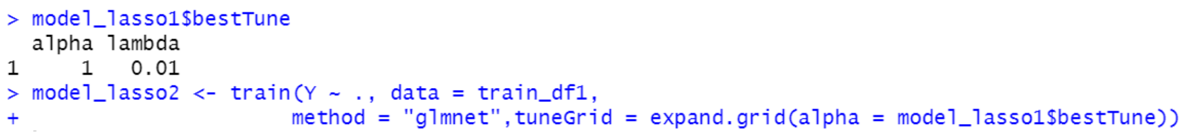


```
# fit the model using random search and cross-validation
model_linear1 <- train(Y ~ ., data = train_df1,
                       method = "lm", trControl = trainControl(method = "cv"),
                       tuneLength = 100)
```



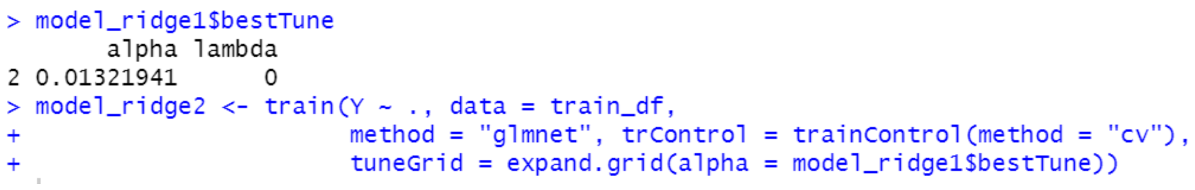
```
> print(model_linear1$bestTune)
  intercept
1      TRUE
```

- The train() function is given the trControl argument which is set to trainControl(method=”cv”) to specify that cross validation is used to evaluate this model.
- In this case, random search was used to determine the value of the intercept hyperparameter in linear regression model. The Value was True which means that the intercept form should be included in the model.



```
> model_lasso1$bestTune
  alpha lambda
1     1   0.01
> model_lasso2 <- train(Y ~ ., data = train_df1,
+                       method = "glmnet",tuneGrid = expand.grid(alpha = model_lasso1$bestTune))
```

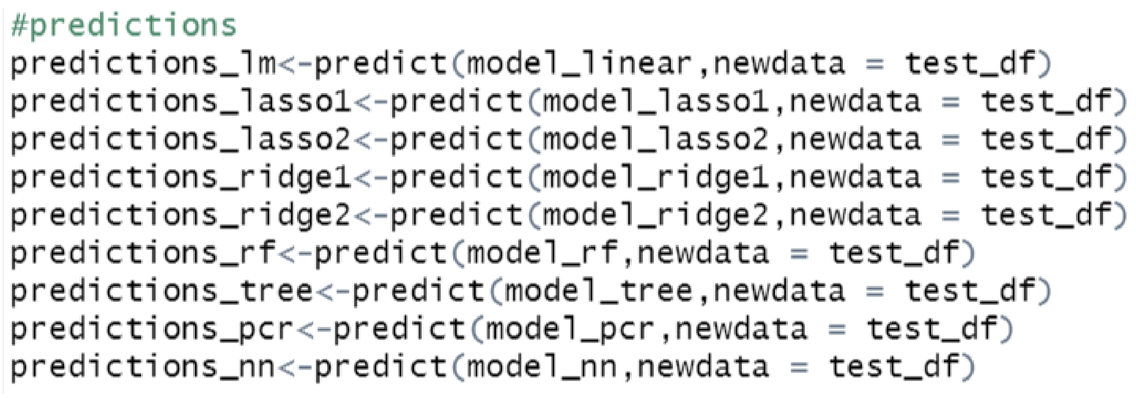
- The train() function is given the trControl argument which is set to trainControl(method=”cv”) to specify that cross validation is used to evaluate this model.
- Also, we have used tuneGrid in which we have specified the optimal alpha, lamda and length values .
- We found these by using bestTune



```
> model_ridge1$bestTune
       alpha lambda
2 0.01321941      0
> model_ridge2 <- train(Y ~ ., data = train_df,
+                       method = "glmnet", trControl = trainControl(method = "cv"),
+                       tuneGrid = expand.grid(alpha = model_ridge1$bestTune))
```

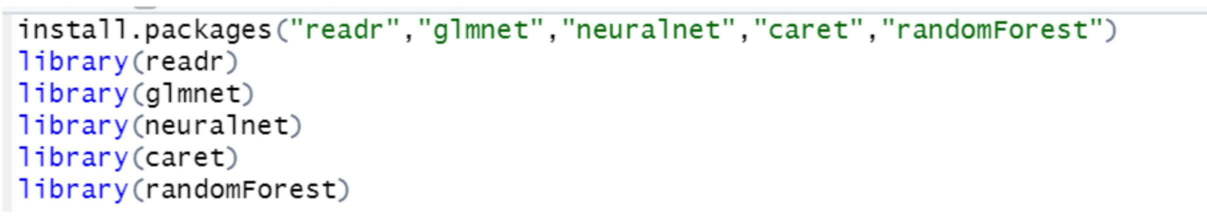
- The train() function is given the trControl argument which is set to trainControl(method=”cv”) to specify that cross validation is used to evaluate this model to help prevent overfitting and improve the model’s performance.
- Also, we have used tuneGrid in which we have specified the optimal alpha, lamda and length values .
- We found these by using bestTune

Chosen Prediction Model



```
#predictions
predictions_lm<-predict(model_linear,newdata = test_df)
predictions_lasso1<-predict(model_lasso1,newdata = test_df)
predictions_lasso2<-predict(model_lasso2,newdata = test_df)
predictions_ridge1<-predict(model_ridge1,newdata = test_df)
predictions_ridge2<-predict(model_ridge2,newdata = test_df)
predictions_rf<-predict(model_rf,newdata = test_df)
predictions_tree<-predict(model_tree,newdata = test_df)
predictions_pcr<-predict(model_pcr,newdata = test_df)
predictions_nn<-predict(model_nn,newdata = test_df)
```

- We chose the random Forest model to get our final result as it had the lowest RMSE,MAE and the highest R squared compared to all the other models even after tuning in few models
- There were no tuning parameters for this model
- I used set.seed(123) so that the sequence of random numbers generated will be same every time I run my code allowing me to get compare my result and verify that they are consistent.
- Variable names
    - nn-neural network
    - rf-random Forest
    - tree-regression tree
- Library used



```
install.packages("readr","glmnet","neuralnet","caret","randomForest")
library(readr)
library(glmnet)
library(neuralnet)
library(caret)
library(randomForest)
```