Final Project: Report 2

Eric Wu, Caden Summers, Jacob O'Leary, Yu-Hsin Liao, Matthew Brady

2023-05-05

Introduction

Background

The purpose of this project is to take a deep look into the use of single-layer neural network. In the part one of the project, we have examine the lab on single-layer neural network on Hitter's data. The author is trying to compare and contrast model performances of three different models based on the mean absolute error of each model. The result is significant, despite the fact that LASSO outperform the multiple linear regression and single-layer neural network, we still get to see how single-layer neural network produces a loss function graph and its basic structure and parameters. For the second part of the project, our goal was to fit a similar single-layer neural network to another data set.

Dataset

We decided to use the "House Rent Prediction" (HRP) dataset found on Kaggle, a data science community known for its enriched various data sources. The HRP dataset consists of 4700+ observations and 11 attributes that allow us to make a prediction on the number of rents given a variety of attributes. Here's the list of variables used and explained by the contributor of the dataset:

- BHK: Number of Bedrooms, Hall, Kitchen
- Rent: Rent of the Houses/Apartments/Flats
- Size: Size of the Houses/Apartments/Flats in Square Feet
- Floor: Houses/Apartments/Flats situated on which Floor and Total Number of Floors (Example: Ground out of 2, 3 out of 5, etc.)
- Area Type: Size of the Houses/Apartments/Flats calculated on either Super Area or Carpet Area or Build Area.
- Area Locality (Area.Locality): Locality of the Houses/Apartments/Flats
- City: City where the Houses/Apartments/Flats are Located
- Furnishing Status: Furnishing Status of the Houses/Apartments/Flats, either it is Furnished or Semi-Furnished or Unfurnished
- Tenant Preferred: Type of Tenant Preferred by the Owner or Agent
- Bathroom: Number of Bathrooms
- Point of Contact (Posted.On): Whom should you contact for more information regarding the Houses/Apartments/Flats

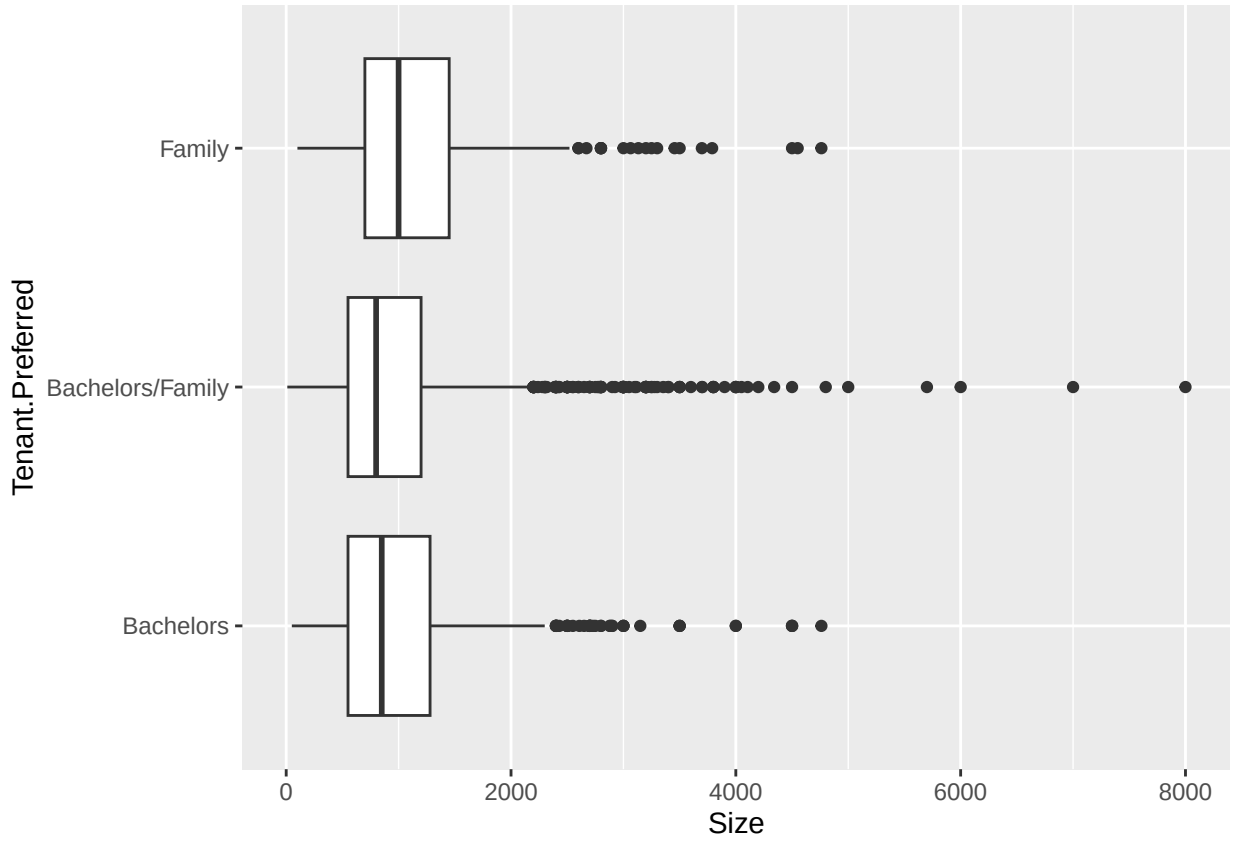
Exploratory Data Analysis

Data Visualization

```
hrp %>%
  group_by(Tenant.Preferred) %>%
  count()
```

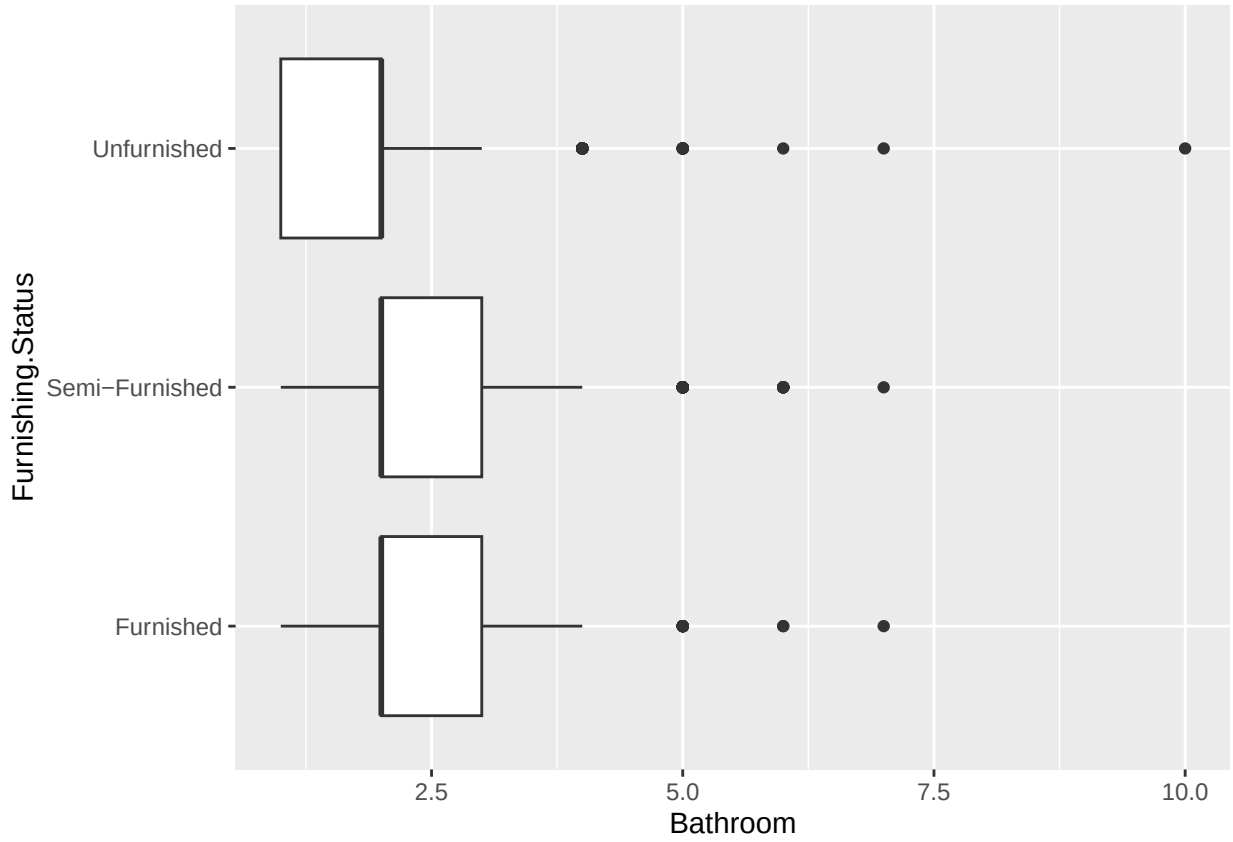
```
## # A tibble: 3 x 2
## # Groups:   Tenant.Preferred [3]
##   Tenant.Preferred     n
##   <chr>            <int>
## 1 Bachelors          830
## 2 Bachelors/Family  3444
## 3 Family             472
```

```
hrp %>%
  ggplot(mapping=aes(x=Size, y=Tenant.Preferred)) +
  geom_boxplot()
```



It seems as though houses that prefer families as tenants may be slightly larger on average than those that prefer bachelors, although the difference is not significant. Houses with no preference, however, seem to be slightly smaller than both on average but there are many more upper outliers. These results may be skewed however, as there are significantly more houses with no preference than those with a preference either way.

```
hrp %>%
  ggplot(mapping=aes(x=Bathroom, y=Furnishing.Status)) +
  geom_boxplot()
```



Clearly Unfurnished houses tend to have less bathrooms than Semi-Furnished and Furnished houses, while Semi-Furnished and Furnished houses seem to generally have the same amount of bathrooms as each other.

Hierarchical clustering

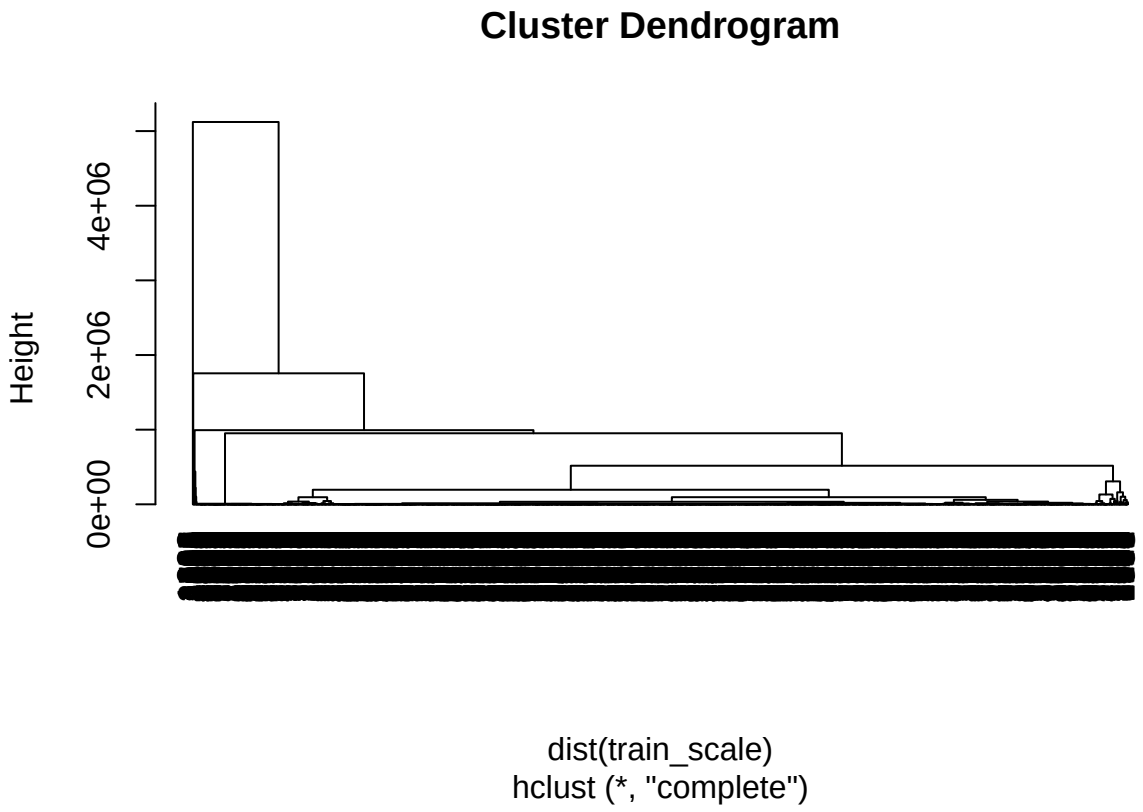
```
hc <- hclust(dist(train_scale))
```

```
## Warning in dist(train_scale): NAs introduced by coercion
```

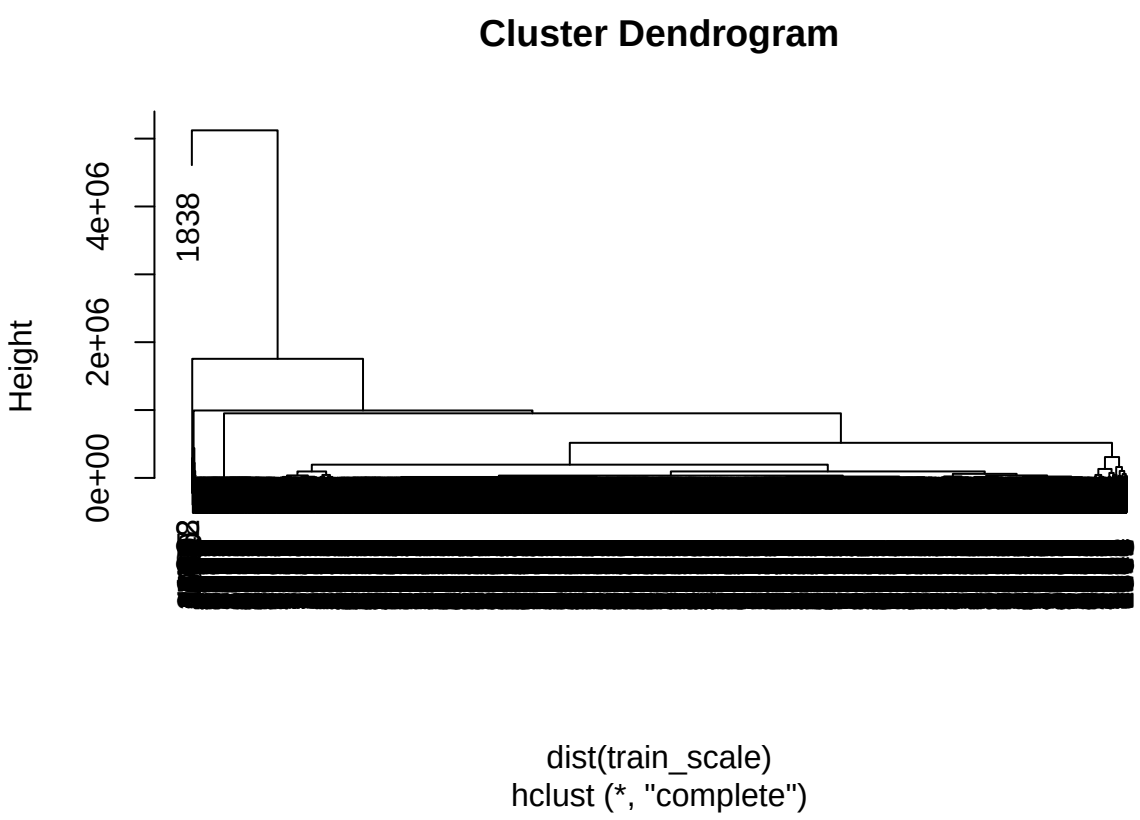
```
hc
```

```
##
## Call:
## hclust(d = dist(train_scale))
##
## Cluster method   : complete
## Distance         : euclidean
## Number of objects: 3796
```

```r
# plot dedrogram (complete linkage)
plot(hc, hang=-1)
```




```r
cut_avg <- cutree(hc, k = 2000)
plot(hc)
```



Due to the high amount of variables in this dataset conduction hierarchical clustering is extremely difficult resluting in a denograph that is impossible to read.

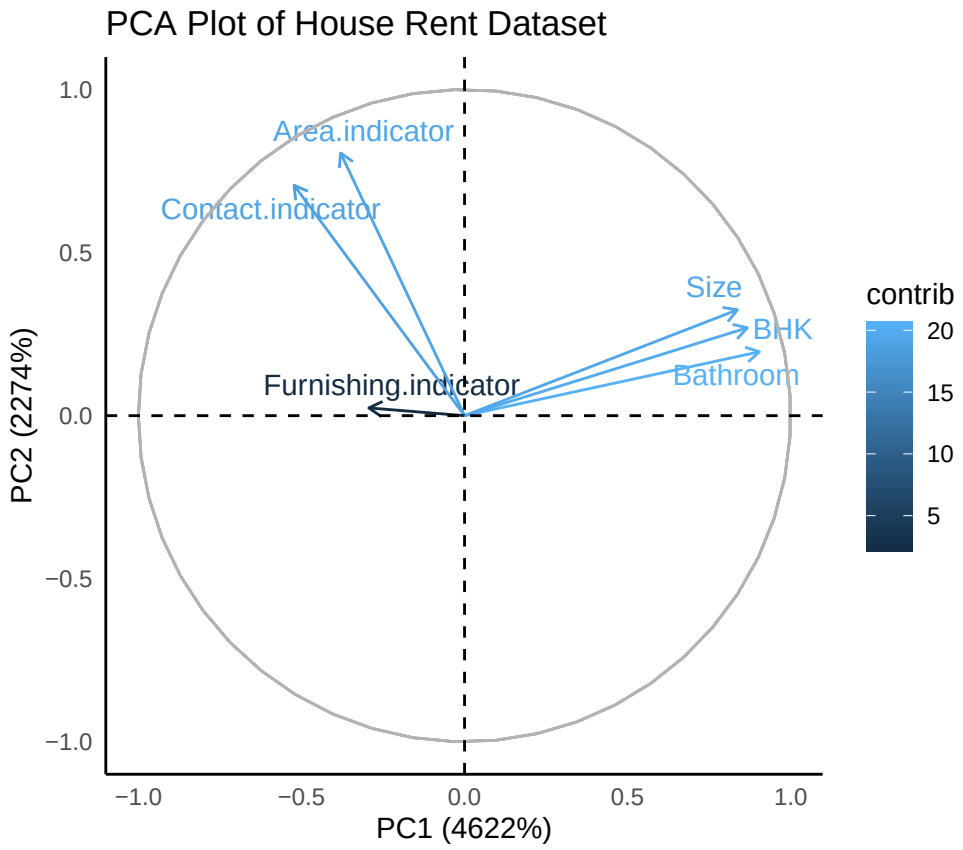
Principal Component Analysis (PCA)

```
# Create indicator variables
hrp_update <- hrp %>%
  mutate(
    Area.indicator = case_when(
      Area.Type == "Built Area" ~ 1,
      Area.Type == "Carpet Area" ~ 2,
      Area.Type == "Super Area" ~ 3
    ),
    Furnishing.indicator = case_when(
      Furnishing.Status == "Furnished" ~ 1,
      Furnishing.Status == "Semi-Furnished" ~ 2,
      Furnishing.Status == "Unfurnished" ~ 3
    ),
    Contact.indicator = case_when(
      Point.of.Contact == "Contact Agent" ~ 1,
      Point.of.Contact == "Contact Builder" ~ 2,
      Point.of.Contact == "Contact Owner" ~ 3
    )
  )
# Scale the data
```

```r
xvars <- c("BHK", "Size", "Area.indicator", "Furnishing.indicator", "Bathroom", "Contact.indicator")
hrp_update[ , xvars] <- scale(hrp_update[ , xvars], center = TRUE, scale = TRUE)

# Perform principal component analysis
pca <- PCA(hrp_update[ , xvars], graph = FALSE)

# Plot the first two principal components
fviz_pca_var(pca, col.var = "contrib", col.ind = "cos2", select.var = list(contrib = 10), repel = TRUE)
  theme_minimal() +
  theme(axis.line = element_line(colour = "black"),
        panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        panel.border = element_blank(),
        panel.background = element_blank()) +
  ggtitle("PCA Plot of House Rent Dataset") +
  xlab(paste0("PC1 (", round(pca$eig[1, 2], 2) * 100, "%)")) +
  ylab(paste0("PC2 (", round(pca$eig[2, 2], 2) * 100, "%)"))
```



Methodology

Our plan is to compare and contrast a multiple regression model, a KNN regression model, and a single-layer neural network in predicting the rent based using other predictors in the dataset. All models will involve some sort of variable selection, whether it's using other functions or included in model building, this is very beneficial when we are doing model selections. The primary focus is to compare the RMSE produced by each model.

Multiple Linear Regression

```
base_model <- lm(Rent ~ . - Area.Locality - Floor - Posted.On, data=train)
base_model2 <- lm(Rent ~ . - Area.Locality - Floor - Posted.On, data=train_alter_lin)
linear_model <- step(base_model, direction="backward")
```

```
## Start:  AIC=84620.79
## Rent ~ (Posted.On + BHK + Size + Floor + Area.Type + Area.Locality +
##     City + Furnishing.Status + Tenant.Preferred + Bathroom +
##     Point.of.Contact) - Area.Locality - Floor - Posted.On
##
##                       Df  Sum of Sq        RSS   AIC
## - Area.Type            2 8.4516e+09 1.8071e+13 84619
## - BHK                  1 5.4526e+09 1.8068e+13 84620
## - Furnishing.Status    2 1.8056e+10 1.8080e+13 84621
## <none>                              1.8062e+13 84621
## - Tenant.Preferred     2 2.1561e+10 1.8084e+13 84621
## - Point.of.Contact     2 3.9625e+10 1.8102e+13 84625
## - Bathroom             1 9.2255e+10 1.8154e+13 84638
## - Size                 1 8.5590e+11 1.8918e+13 84795
## - City                 5 1.2257e+12 1.9288e+13 84860
##
## Step:  AIC=84618.57
## Rent ~ BHK + Size + City + Furnishing.Status + Tenant.Preferred +
##     Bathroom + Point.of.Contact
##
##                       Df  Sum of Sq        RSS   AIC
## - BHK                  1 6.6024e+09 1.8077e+13 84618
## - Furnishing.Status    2 1.8471e+10 1.8089e+13 84618
## <none>                              1.8071e+13 84619
## - Tenant.Preferred     2 1.9721e+10 1.8090e+13 84619
## - Point.of.Contact     2 6.3143e+10 1.8134e+13 84628
## - Bathroom             1 9.1734e+10 1.8162e+13 84636
## - Size                 1 8.4887e+11 1.8920e+13 84791
## - City                 5 1.2770e+12 1.9348e+13 84868
##
## Step:  AIC=84617.96
## Rent ~ Size + City + Furnishing.Status + Tenant.Preferred + Bathroom +
##     Point.of.Contact
##
##                       Df  Sum of Sq        RSS   AIC
## - Furnishing.Status    2 1.8159e+10 1.8095e+13 84618
## - Tenant.Preferred     2 1.8828e+10 1.8096e+13 84618
## <none>                              1.8077e+13 84618
## - Point.of.Contact     2 6.3087e+10 1.8140e+13 84627
## - Bathroom             1 1.8758e+11 1.8265e+13 84655
## - Size                 1 9.6237e+11 1.9040e+13 84813
## - City                 5 1.2704e+12 1.9348e+13 84866
##
## Step:  AIC=84617.77
## Rent ~ Size + City + Tenant.Preferred + Bathroom + Point.of.Contact
##
##                     Df  Sum of Sq        RSS   AIC
```

```
## - Tenant.Preferred  2 1.8475e+10 1.8114e+13 84618
## <none>                           1.8095e+13 84618
## - Point.of.Contact  2 6.5553e+10 1.8161e+13 84627
## - Bathroom          1 1.8746e+11 1.8283e+13 84655
## - Size              1 9.8759e+11 1.9083e+13 84817
## - City              5 1.3177e+12 1.9413e+13 84875
##
## Step:  AIC=84617.64
## Rent ~ Size + City + Bathroom + Point.of.Contact
##
##                     Df  Sum of Sq        RSS   AIC
## <none>                           1.8114e+13 84618
## - Point.of.Contact  2 6.4514e+10 1.8178e+13 84627
## - Bathroom          1 1.8254e+11 1.8296e+13 84654
## - Size              1 9.8726e+11 1.9101e+13 84817
## - City              5 1.3129e+12 1.9427e+13 84873
```

```
linear_model2 <- step(base_model2, direction="backward")
```

```
## Start:  AIC=84673.5
## Rent ~ (Posted.On + BHK + Size + Floor + Area.Type + Area.Locality +
##     City + Furnishing.Status + Tenant.Preferred + Bathroom +
##     Point.of.Contact) - Area.Locality - Floor - Posted.On
##
##                     Df  Sum of Sq        RSS   AIC
## - Area.Type          2 5.6933e+09 1.8320e+13 84671
## - Furnishing.Status  2 1.4280e+10 1.8329e+13 84672
## - BHK                1 5.3648e+09 1.8320e+13 84673
## <none>                            1.8315e+13 84673
## - Tenant.Preferred   2 2.4185e+10 1.8339e+13 84675
## - Point.of.Contact   2 3.5665e+10 1.8350e+13 84677
## - Bathroom           1 7.6548e+10 1.8391e+13 84687
## - Size               1 9.4728e+11 1.9262e+13 84863
## - City               5 1.2884e+12 1.9603e+13 84922
##
## Step:  AIC=84670.68
## Rent ~ BHK + Size + City + Furnishing.Status + Tenant.Preferred +
##     Bathroom + Point.of.Contact
##
##                     Df  Sum of Sq        RSS   AIC
## - Furnishing.Status  2 1.4382e+10 1.8335e+13 84670
## - BHK                1 6.0776e+09 1.8326e+13 84670
## <none>                            1.8320e+13 84671
## - Tenant.Preferred   2 2.3215e+10 1.8344e+13 84671
## - Point.of.Contact   2 5.3567e+10 1.8374e+13 84678
## - Bathroom           1 7.6372e+10 1.8397e+13 84684
## - Size               1 9.4241e+11 1.9263e+13 84859
## - City               5 1.3418e+12 1.9662e+13 84929
##
## Step:  AIC=84669.66
## Rent ~ BHK + Size + City + Tenant.Preferred + Bathroom + Point.of.Contact
##
##                     Df  Sum of Sq        RSS   AIC
## - BHK                1 5.7767e+09 1.8341e+13 84669
```

```
## <none>                           1.8335e+13 84670
## - Tenant.Preferred  2 2.2712e+10 1.8358e+13 84670
## - Point.of.Contact  2 5.4085e+10 1.8389e+13 84677
## - Bathroom          1 7.7270e+10 1.8412e+13 84684
## - Size              1 9.6131e+11 1.9296e+13 84862
## - City              5 1.3913e+12 1.9726e+13 84937
##
## Step:  AIC=84668.85
## Rent ~ Size + City + Tenant.Preferred + Bathroom + Point.of.Contact
##
##                     Df  Sum of Sq        RSS   AIC
## <none>                           1.8341e+13 84669
## - Tenant.Preferred  2 2.1788e+10 1.8362e+13 84669
## - Point.of.Contact  2 5.4099e+10 1.8395e+13 84676
## - Bathroom          1 1.5450e+11 1.8495e+13 84699
## - Size              1 1.0803e+12 1.9421e+13 84884
## - City              5 1.3861e+12 1.9727e+13 84935
```

```
linpred <- predict(linear_model, newdata=test)
test_linpred <-
  test %>%
  cbind(linpred) %>%
  mutate(Resid.Square = (Rent-linpred)^2)
sqrt(sum(test_linpred$Resid.Square)/nrow(test_linpred))
```

```
## [1] 42803.14
```

```
linpred2 <- predict(linear_model2, newdata=test_alter_lin)
test_linpred2 <-
  test_alter_lin %>%
  cbind(linpred2) %>%
  mutate(Resid.Square = (Rent-linpred2)^2)
sqrt(sum(test_linpred2$Resid.Square)/nrow(test_linpred2))
```

```
## [1] 39529.23
```

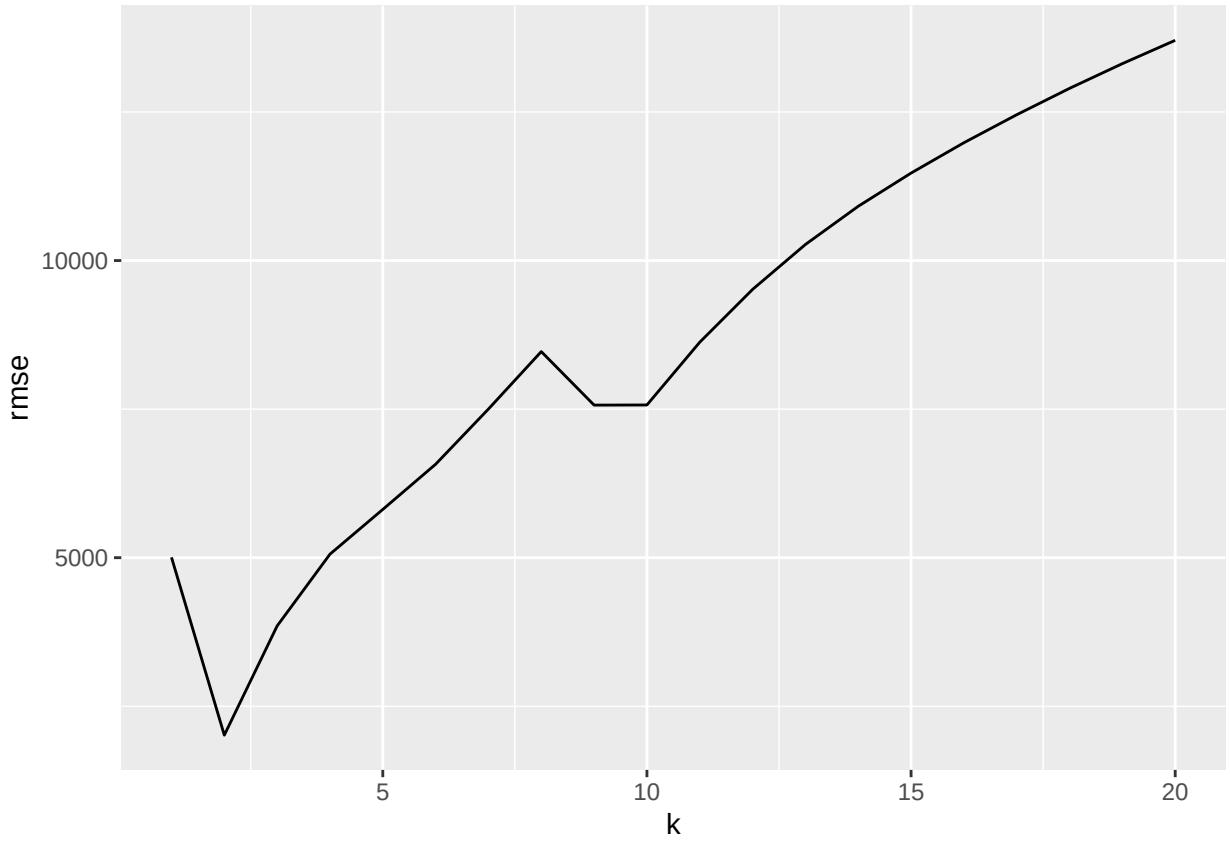
RMSE with seed 123 for multiple linear regression is 42803.14, while with seed 12345 RMSE is 39529.23. So, there is a potential for slight variations in the RMSE based on the random seed, but not too significant, as this is a less than 10% difference in RMSE.

KNN Regression

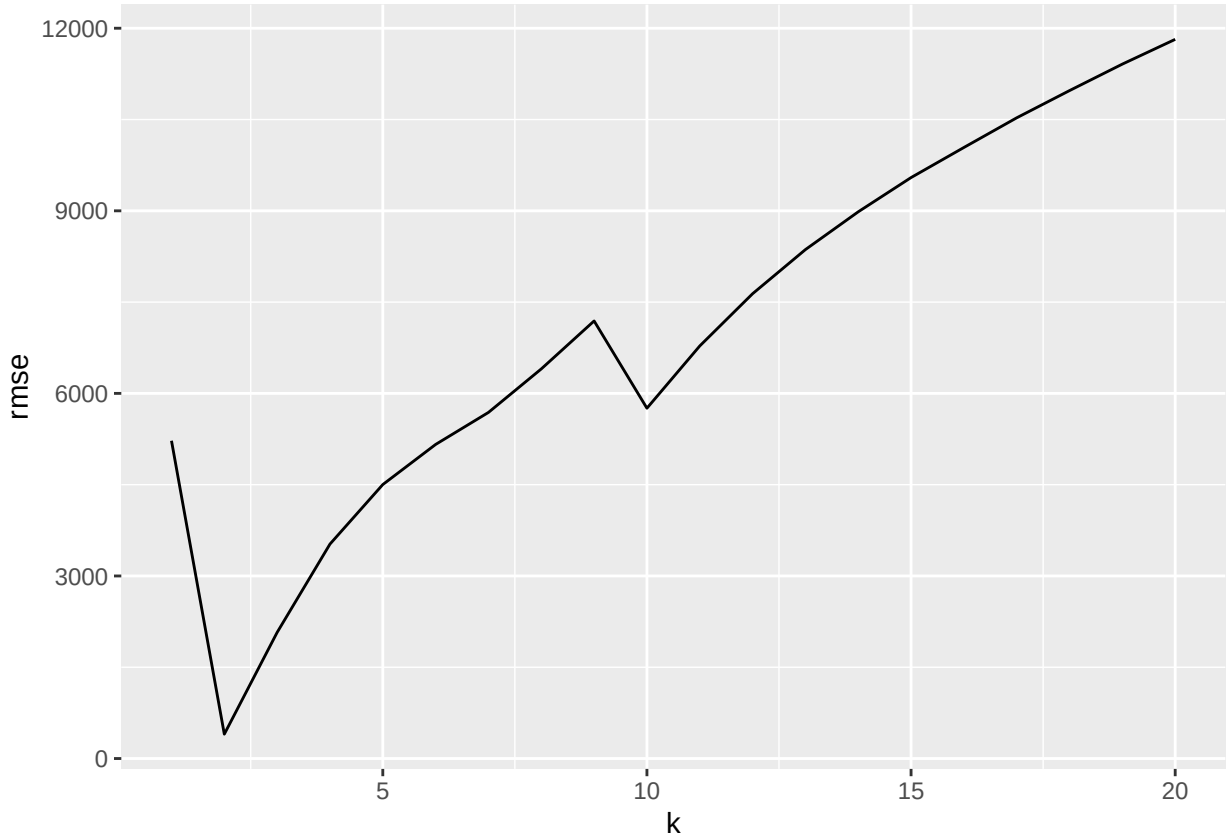
```
xvars <- names(select_if(train_scale, is.numeric))
maxK <- 20
mse_vec <- rep(NA, maxK)
mse_vec2 <- rep(NA, maxK)
rmse_vec <- rep(NA, maxK)
rmse_vec2 <- rep(NA, maxK)
for(i in 1:maxK){
  knn_res <- knn.reg(train = train_scale[ , xvars, drop = FALSE],
```

```
                  test = test_scale[ , xvars, drop = FALSE],
                  y = train_scale$Rent,
                  k = i)
  knn_res2 <- knn.reg(train = train_alter[ , xvars, drop = FALSE],
                  test = test_alter[ , xvars, drop = FALSE],
                  y = train_alter$Rent,
                  k = i)

  mse_vec[i] <- mean((test_scale$Rent - knn_res$pred)^2)
  mse_vec2[i] <- mean((test_alter$Rent - knn_res2$pred)^2)
  rmse_vec[i] <- sqrt(mse_vec[i])
  rmse_vec2[i] <- sqrt(mse_vec2[i])
}
choice_k <- data.frame(k = 1:maxK, rmse = rmse_vec)
ggplot(data = choice_k, mapping = aes(x = k, y = rmse)) +
   geom_line()
```




```
choice_k2 <- data.frame(k = 1:maxK, rmse = rmse_vec2)
ggplot(data = choice_k2, mapping = aes(x = k, y = rmse)) +
   geom_line()
```




```
# Based on the graph the best K value seems to be the lowest point on the graph.
```

```
# Based on the front of this vector the best K value is 2.
head(choice_k)
```

```
##   k     rmse
## 1 1 5005.750
## 2 2 2014.333
## 3 3 3851.691
## 4 4 5059.494
## 5 5 5812.080
## 6 6 6572.819
```

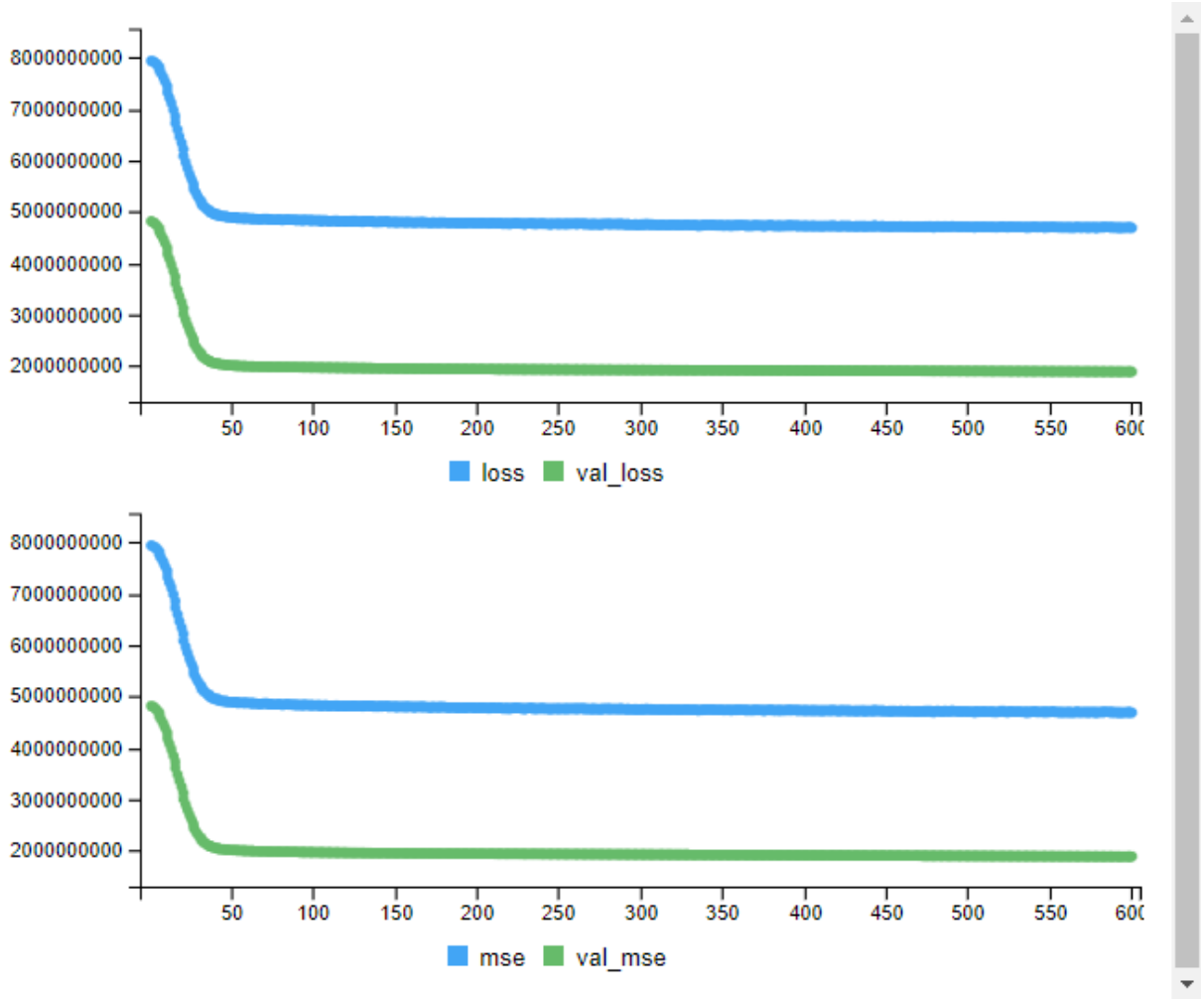
```
head(choice_k2)
```

```
##   k      rmse
## 1 1 5221.6286
## 2 2  400.5021
## 3 3 2072.5758
## 4 4 3524.0956
## 5 5 4501.9773
## 6 6 5160.7870
```

```
# The KNN regression is also not immune to seed changes and in this case results in an RMSE that is 160
# Although the RMSE has changed significantly, it is still far better than the other methods presented
```

Single-Layer Neural Network

```
modnn <- keras_model_sequential() %>%
     layer_dense(units = 2351, activation = "relu",
        input_shape = ncol(x)) %>%
     layer_dropout(rate = 0.4) %>%
     layer_dense(units = 1)
modnn %>% compile(loss = "mse",
    optimizer = optimizer_rmsprop(),
    metrics = list("mse")
   )
#history <- modnn %>% fit(
#     x[train_ind, ], y[train_ind], epochs = 600, batch_size = 32, #
#    validation_data = list(x[-train_ind, ], y[-train_ind])
#  )
#npred <- predict(modnn, x[-train_ind, ])
#nn_mse <- mean((y[-train_ind] - npred)^2)
#sqrt(nn_mse)
## 43418.85
```



The structure of the single-layer neural network is very similar to the lab, except for the 2351 hidden units, and we are measuring the Mean Square Error(MSE) instead of the Mean Absolute Error(MAE). The RMSE resulting from the training end up being 43418.85. And base on the loss function graph, the MSE of the validation(testing) set is significantly smaller than the training set. Which indicates better performance in the testing set.

```
#modnn2 <- keras_model_sequential() %>%
#     layer_dense(units = 2351, activation = "relu",
#        input_shape = ncol(x)) %>%
#     layer_dropout(rate = 0.4) %>%
#     layer_dense(units = 1)
#modnn2 %>% compile(loss = "mse",
#    optimizer = optimizer_rmsprop(),
#    metrics = list("mse")
#   )
#history2 <- modnn2 %>% fit(
#     x[train_ind_alter, ], y[train_ind_alter], epochs = 600, batch_size = 32, #
#    validation_data = list(x[-train_ind_alter, ], y[-train_ind_alter])
#  )
#npred2 <- predict(modnn2, x[-train_ind_alter, ])
#nn_mse2 <- mean((y[-train_ind_alter] - npred2)^2)
#sqrt(nn_mse2)
```

```
# 37620.59
```

On the other hand, the single-layer neural network are not resistant to changed of seed, as the MSE is greatly reduced and the RMSE decreases from 43418.85 to 37620.59. Unfortunately, the picture of the loss function graph is distorted, and we wouldn't able to placed the page here but rather at the end.

Discussion

After testing our three methods on the House Rent dataset with the seed set to 123, we got RMSE values of 42,803.14, 2014.33, and 43,418.85 for Multiple Linear regression, KNN regression, and Single-Layer Neural Network respectively. Based on these values it is very clear that the KNN method with a k-value of 2 is significantly better than both of these methods. The performance is roughly 20 times better with this metric alone. We also wanted to test the models accuracy against varying seed values to ensure that they are robust and not easily affected by small changes. From this testing we confirmed that none of the models were immune to changes in the seed. The KNN regression in particular was able to get its RMSE as low as 400 which is far lower than the neural networks ~37,620.

Conclusion

Overall, the project is very successful and we get to review and learn more about the implementation of various methods that was taught in class. In addition, the project allows us to examine the parameters of single-layer neural network, and the optimization function that goes along with the train process.

Code Appendix

```
# Load essential libraries
library(tidyverse)
library(ggplot2)
library(keras)
library(ISLR2)
library(FNN)
library(FactoMineR)
library(factoextra)
library(dplyr)

# Load the dataset
hrp <- read.csv("House_Rent_Dataset.csv")

# Create indicator variables
hrp_update <- hrp %>%
  mutate(
    Area.indicator = case_when(
      Area.Type == "Built Area" ~ 1,
      Area.Type == "Carpet Area" ~ 2,
      Area.Type == "Super Area" ~ 3
    ),
    Furnishing.indicator = case_when(
      Furnishing.Status == "Furnished" ~ 1,
      Furnishing.Status == "Semi-Furnished" ~ 2,
      Furnishing.Status == "Unfurnished" ~ 3
```

```
    ),
    Contact.indicator = case_when(
      Point.of.Contact == "Contact Agent" ~ 1,
      Point.of.Contact == "Contact Builder" ~ 2,
      Point.of.Contact == "Contact Owner" ~ 3
    )
  )

# Set up all parameters
seed <- 123
seed_alter <- 12345

# Scale the data
xvars <- c("BHK", "Size", "Area.indicator", "Furnishing.indicator", "Bathroom", "Contact.indicator")
hrp_update[ , xvars] <- scale(hrp_update[ , xvars], center = TRUE, scale = TRUE)

# Set up the necessary matrix for neural network
x <- scale(model.matrix(Rent ~ BHK + Size + Area.indicator + Furnishing.indicator + Bathroom + Contact.
y <- hrp_update$Rent

# Set up the training/testing split
set.seed(seed)
train_ind <- sample(1:nrow(hrp_update), floor(0.8 * nrow(hrp_update)))
set.seed(NULL)
set.seed(seed_alter)
train_ind_alter <- sample(1:nrow(hrp_update), floor(0.8 * nrow(hrp_update)))
set.seed(NULL)

# Not scaled data
train <- hrp[train_ind, ]
test <- hrp[-train_ind, ]

# Not scaled data with alternative seed
train_alter_lin <- hrp[train_ind_alter, ]
test_alter_lin <- hrp[-train_ind_alter, ]

# Scaled data
train_scale <- hrp_update[train_ind, ]
test_scale <- hrp_update[-train_ind, ]

# Scaled data with alternative seed
train_alter <- hrp_update[train_ind_alter, ]
test_alter <- hrp_update[-train_ind_alter, ]

hrp %>%
  group_by(Tenant.Preferred) %>%
  count()

hrp %>%
  ggplot(mapping=aes(x=Size, y=Tenant.Preferred)) +
  geom_boxplot()

hrp %>%
```

```
  ggplot(mapping=aes(x=Bathroom, y=Furnishing.Status)) +
  geom_boxplot()

hc <- hclust(dist(train_scale))
hc

# plot dedrogram (complete linkage)
plot(hc, hang=-1)

cut_avg <- cutree(hc, k = 2000)
plot(hc)

# Create indicator variables
hrp_update <- hrp %>%
  mutate(
    Area.indicator = case_when(
      Area.Type == "Built Area" ~ 1,
      Area.Type == "Carpet Area" ~ 2,
      Area.Type == "Super Area" ~ 3
    ),
    Furnishing.indicator = case_when(
      Furnishing.Status == "Furnished" ~ 1,
      Furnishing.Status == "Semi-Furnished" ~ 2,
      Furnishing.Status == "Unfurnished" ~ 3
    ),
    Contact.indicator = case_when(
      Point.of.Contact == "Contact Agent" ~ 1,
      Point.of.Contact == "Contact Builder" ~ 2,
      Point.of.Contact == "Contact Owner" ~ 3
    )
  )
# Scale the data
xvars <- c("BHK", "Size", "Area.indicator", "Furnishing.indicator", "Bathroom", "Contact.indicator")
hrp_update[ , xvars] <- scale(hrp_update[ , xvars], center = TRUE, scale = TRUE)

# Perform principal component analysis
pca <- PCA(hrp_update[ , xvars], graph = FALSE)

# Plot the first two principal components
fviz_pca_var(pca, col.var = "contrib", col.ind = "cos2", select.var = list(contrib = 10), repel = TRUE)
  theme_minimal() +
  theme(axis.line = element_line(colour = "black"),
        panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        panel.border = element_blank(),
        panel.background = element_blank()) +
  ggtitle("PCA Plot of House Rent Dataset") +
  xlab(paste0("PC1 (", round(pca$eig[1, 2], 2) * 100, "%)")) +
  ylab(paste0("PC2 (", round(pca$eig[2, 2], 2) * 100, "%)"))

base_model <- lm(Rent ~ . - Area.Locality - Floor - Posted.On, data=train)
base_model2 <- lm(Rent ~ . - Area.Locality - Floor - Posted.On, data=train_alter_lin)
linear_model <- step(base_model, direction="backward")
```

```
linear_model2 <- step(base_model2, direction="backward")

linpred <- predict(linear_model, newdata=test)
test_linpred <-
  test %>%
  cbind(linpred) %>%
  mutate(Resid.Square = (Rent-linpred)^2)
sqrt(sum(test_linpred$Resid.Square)/nrow(test_linpred))

linpred2 <- predict(linear_model2, newdata=test_alter_lin)
test_linpred2 <-
  test_alter_lin %>%
  cbind(linpred2) %>%
  mutate(Resid.Square = (Rent-linpred2)^2)
sqrt(sum(test_linpred2$Resid.Square)/nrow(test_linpred2))

xvars <- names(select_if(train_scale, is.numeric))
maxK <- 20
mse_vec <- rep(NA, maxK)
mse_vec2 <- rep(NA, maxK)
rmse_vec <- rep(NA, maxK)
rmse_vec2 <- rep(NA, maxK)
for(i in 1:maxK){
  knn_res <- knn.reg(train = train_scale[ , xvars, drop = FALSE],
                     test = test_scale[ , xvars, drop = FALSE],
                     y = train_scale$Rent,
                     k = i)
  knn_res2 <- knn.reg(train = train_alter[ , xvars, drop = FALSE],
                     test = test_alter[ , xvars, drop = FALSE],
                     y = train_alter$Rent,
                     k = i)

  mse_vec[i] <- mean((test_scale$Rent - knn_res$pred)^2)
  mse_vec2[i] <- mean((test_alter$Rent - knn_res2$pred)^2)
  rmse_vec[i] <- sqrt(mse_vec[i])
  rmse_vec2[i] <- sqrt(mse_vec2[i])
}
choice_k <- data.frame(k = 1:maxK, rmse = rmse_vec)
ggplot(data = choice_k, mapping = aes(x = k, y = rmse)) +
   geom_line()
choice_k2 <- data.frame(k = 1:maxK, rmse = rmse_vec2)
ggplot(data = choice_k2, mapping = aes(x = k, y = rmse)) +
   geom_line()
# Based on the graph the best K value seems to be the lowest point on the graph.

# Based on the front of this vector the best K value is 2.
head(choice_k)
head(choice_k2)
# The KNN regression is also not immune to seed changes and in this case results in an RMSE that is 160
# Although the RMSE has changed significantly, it is still far better than the other methods presented

modnn <- keras_model_sequential() %>%
     layer_dense(units = 2351, activation = "relu",
```

```
        input_shape = ncol(x)) %>%
     layer_dropout(rate = 0.4) %>%
     layer_dense(units = 1)
modnn %>% compile(loss = "mse",
    optimizer = optimizer_rmsprop(),
    metrics = list("mse")
   )
#history <- modnn %>% fit(
#     x[train_ind, ], y[train_ind], epochs = 600, batch_size = 32, #
#    validation_data = list(x[-train_ind, ], y[-train_ind])
#  )
#npred <- predict(modnn, x[-train_ind, ])
#nn_mse <- mean((y[-train_ind] - npred)^2)
#sqrt(nn_mse)
## 43418.85

#modnn2 <- keras_model_sequential() %>%
#     layer_dense(units = 2351, activation = "relu",
#        input_shape = ncol(x)) %>%
#     layer_dropout(rate = 0.4) %>%
#     layer_dense(units = 1)
#modnn2 %>% compile(loss = "mse",
#    optimizer = optimizer_rmsprop(),
#    metrics = list("mse")
#   )
#history2 <- modnn2 %>% fit(
#     x[train_ind_alter, ], y[train_ind_alter], epochs = 600, batch_size = 32, #
#    validation_data = list(x[-train_ind_alter, ], y[-train_ind_alter])
#  )
#npred2 <- predict(modnn2, x[-train_ind_alter, ])
#nn_mse2 <- mean((y[-train_ind_alter] - npred2)^2)
#sqrt(nn_mse2)
# 37620.59
```

Contributions

Caden: KNN regression, Discussion Jacob: Multiple Linear Regression, Data Visualization Eric: Introduction, Neural network, conclusion Yu-Hsin: PCA Plot Matt: Hierarchical Clustering

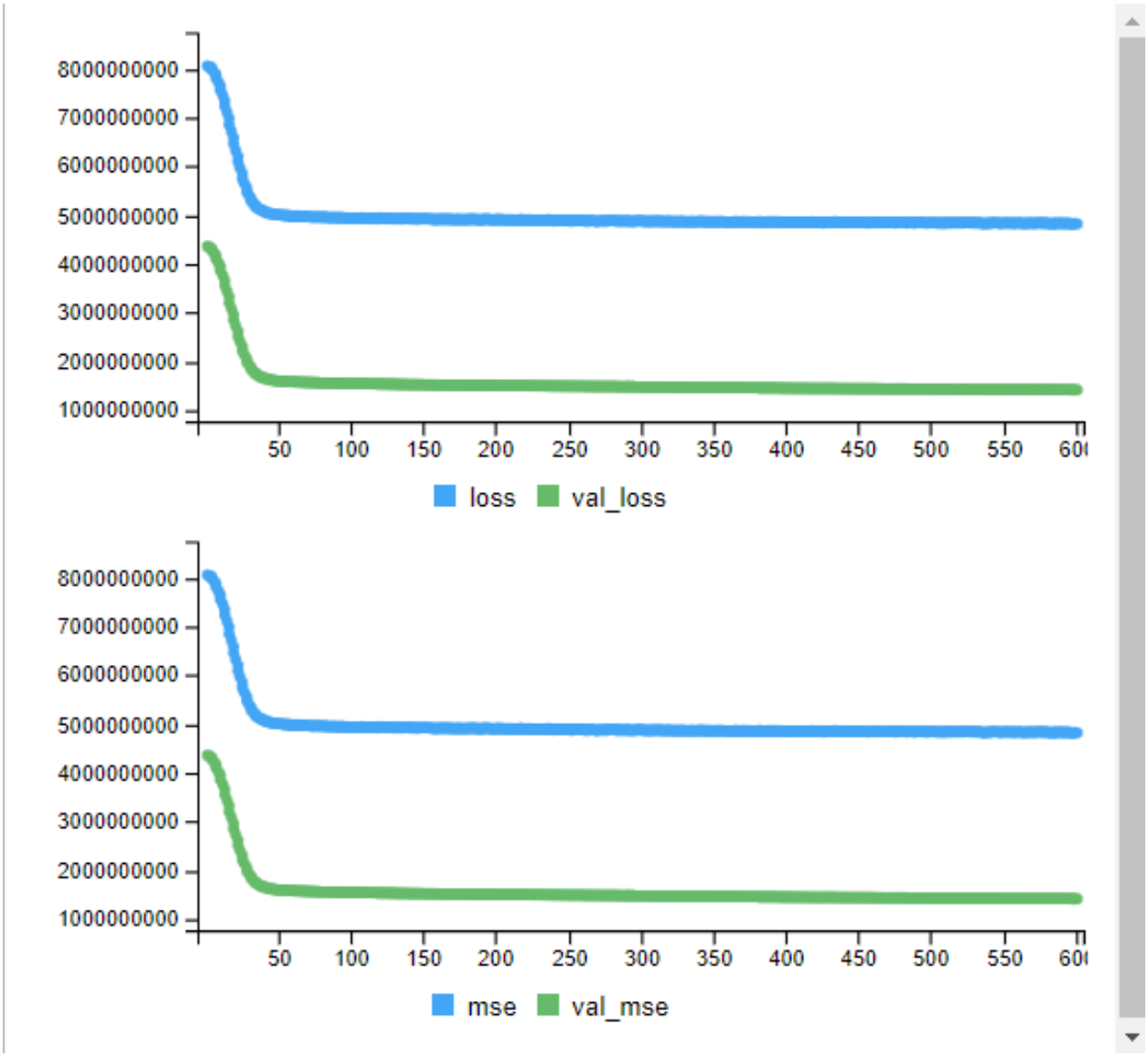


Figure 1: Image 2: Training Results from single-layer Neural Network using alternative seed