ISL Final Project Phase II

Pooria Assarehha

2025-02-06

```r
library(ggplot2)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
library(tidyr)
library(corrplot)
```

```
## corrplot 0.95 loaded
```

```r
library(gridExtra)
```

```
##
## Attaching package: 'gridExtra'

## The following object is masked from 'package:dplyr':
##
##     combine
```

```r
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v forcats   1.0.0     v readr     2.1.5
## v lubridate 1.9.3     v stringr   1.5.1
## v purrr     1.0.2     v tibble    3.2.1

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x gridExtra::combine() masks dplyr::combine()
## x dplyr::filter()      masks stats::filter()
## x dplyr::lag()         masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become error
```

```r
library(reshape2)
```

```
##
## Attaching package: 'reshape2'
##
## The following object is masked from 'package:tidyr':
##
```

```
##     smiths
```

```r
library(moments)

library(caret)
```

```
## Loading required package: lattice
##
## Attaching package: 'caret'
##
## The following object is masked from 'package:purrr':
##
##     lift
```

```r
data_path <- "clean_data.csv"
data <- read.csv(data_path, stringsAsFactors = TRUE)
```

Data Preparation

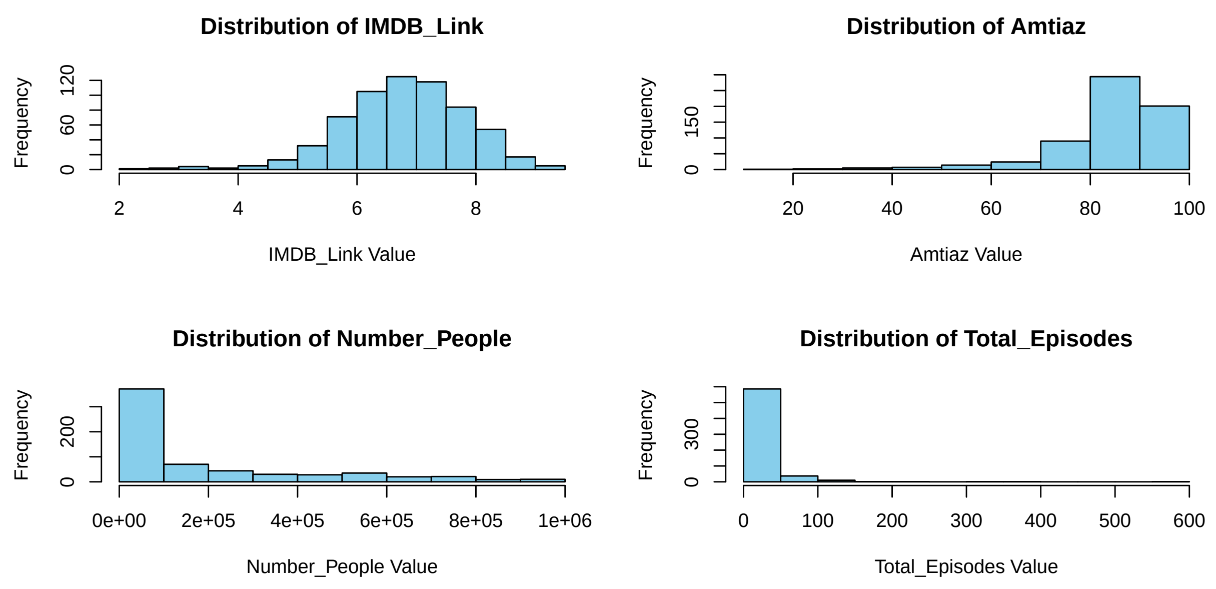
The first to columns assume no role in our estimation, they can be omitted. After reading the file each feature must take its datatype by definition. Then we separate out target variable from predictive features.

```r
# Remove the first two columns
data %>% select(!c(URL, Name)) -> data

# Convert binary columns to factors
for (col in names(data)){
  if (all(unique(data[,col]) == c(0,1) )  || all(unique(data[,col]) == c(1,0)))
    data[,col] = factor(data[,col])
}

# Define the target variable
target <- data$Amtiaz

# Remove the target variable column from the features
features <- data %>% select(!Amtiaz)
```

Exploratory Data Analysis

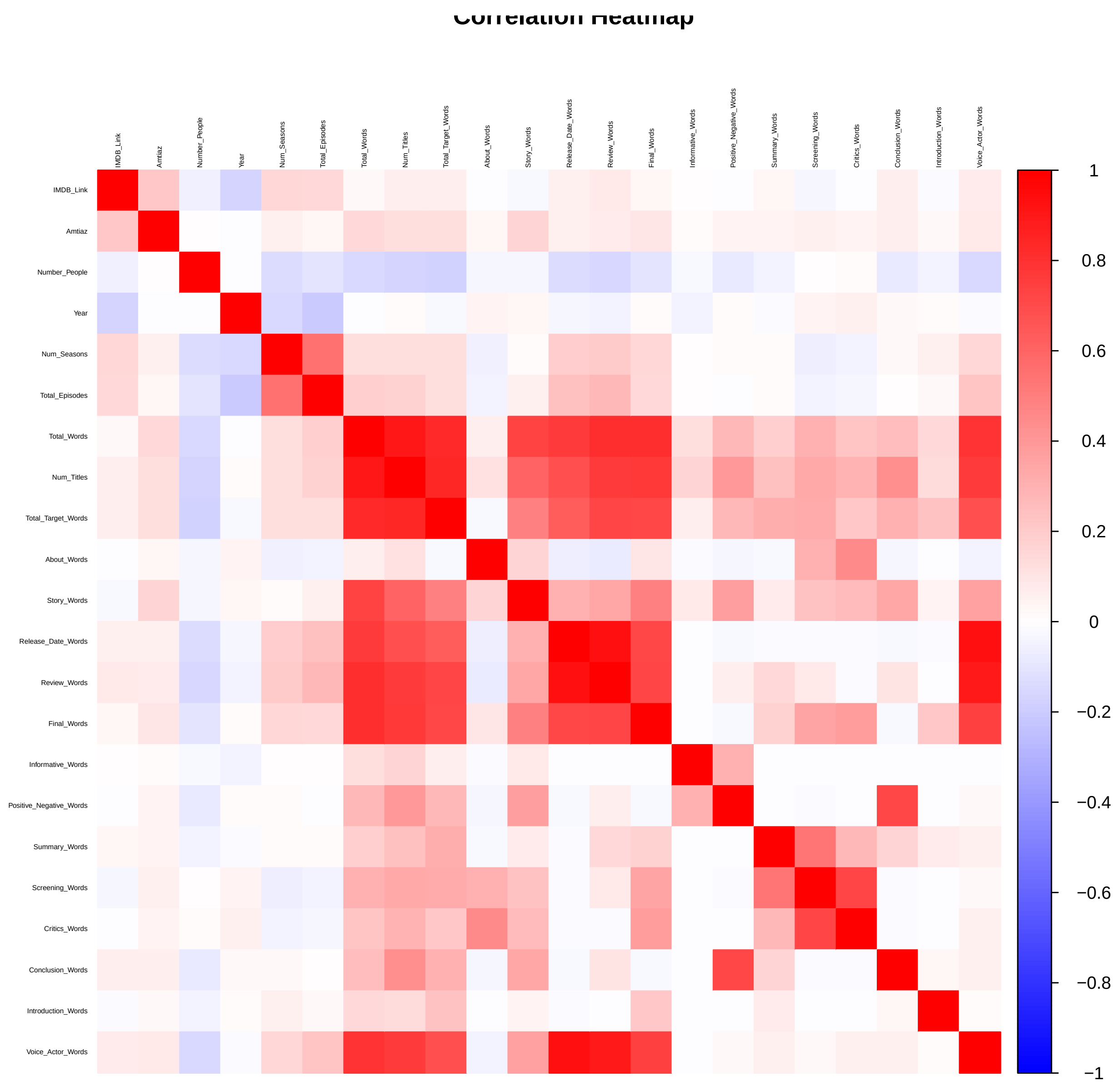
Histograms for key numerical variables

```r
num_cols <- c("IMDB_Link", "Amtiaz", "Number_People", "Total_Episodes")

par(mfrow=c(2,2))
for (col in num_cols) {
  hist(
    data[[col]],
    main=paste("Distribution of", col),
    xlab = paste(col, "Value"),
    col="skyblue",
    border="black")
}
```



Correlation Heatmap

```r
num_data <- data %>% select_if(is.numeric)
corr_matrix <- cor(num_data, use="complete.obs")

corrplot(
  corr_matrix,
  method="color",
  col=colorRampPalette(c("blue", "white", "red"))(200),
  tl.cex=0.35, tl.col="black",
  title="Correlation Heatmap")
```

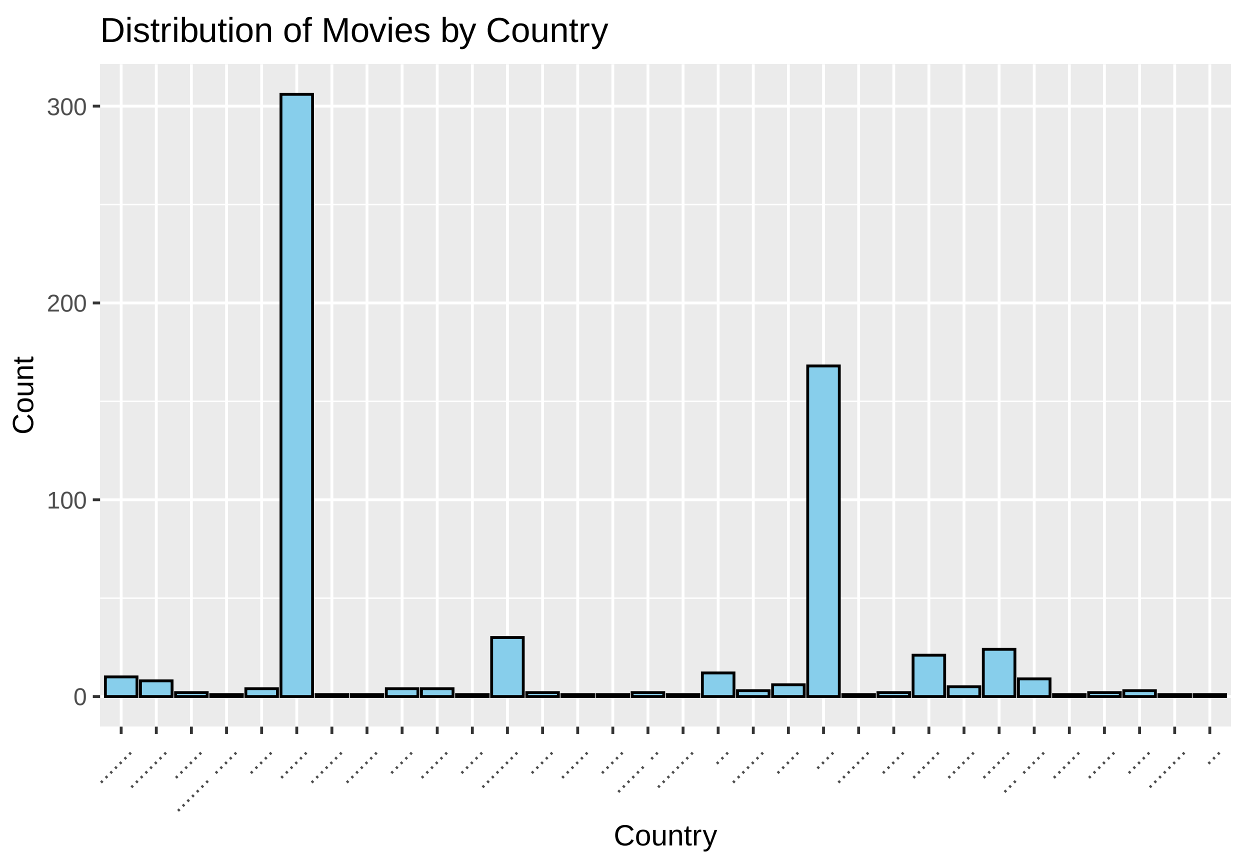


Strong correlations exist between IMDB_Link, Amtiaz, and Number_People, indicating possible relationships worth exploring in modeling.

Most of our features don’t show any relation to target variable or any other feature.

Bar chart for Categorical Features, Country distribution

```r
data %>%
  ggplot(aes(x=Country)) +
  geom_bar(fill="skyblue", color="black") +
  theme(axis.text.x = element_text(angle=45, hjust=1)) +
  labs(title="Distribution of Movies by Country", x="Country", y="Count")
```



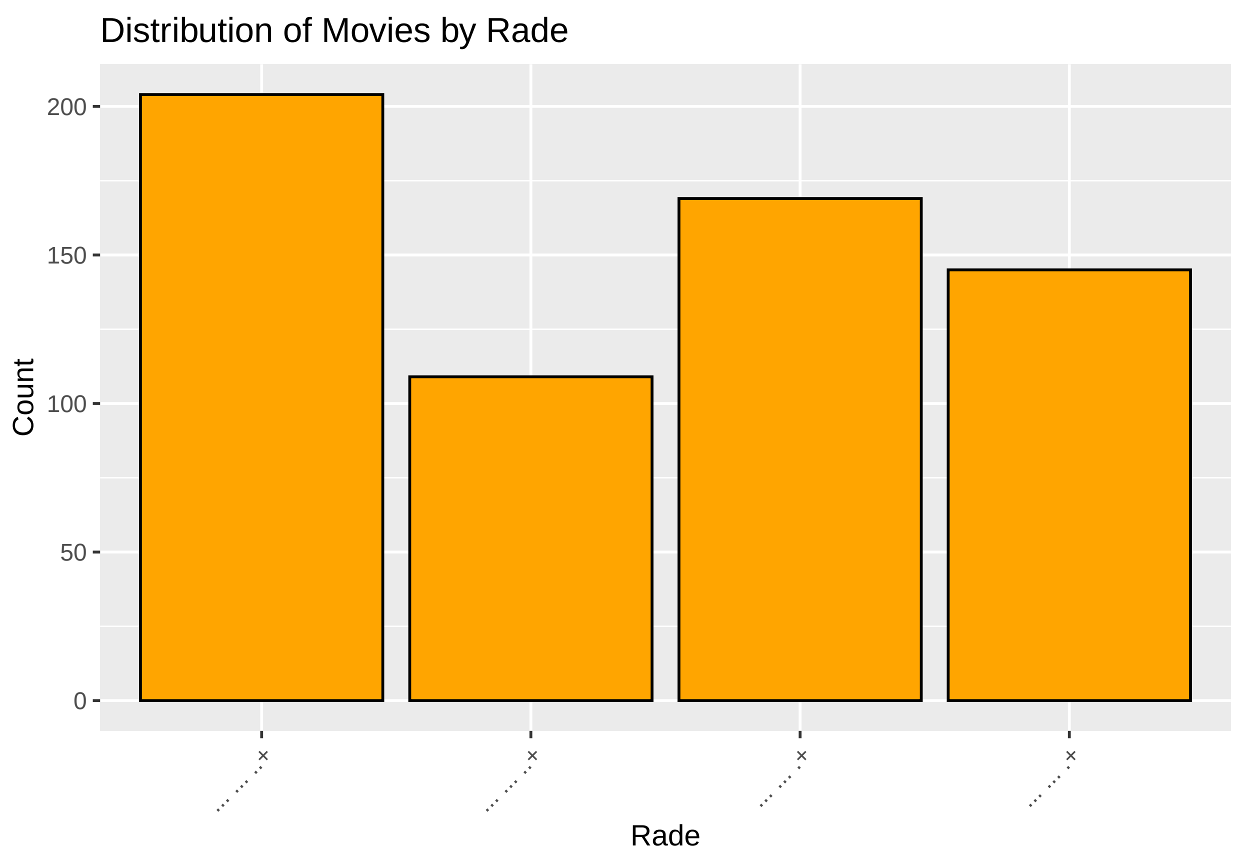
There are Countries that only appear once in our data, hence no inference or estimation can be done with them, let's simply omit those.

```
data %>%
  group_by(Country) %>%
  filter(n() > 1) -> data
nrow(data)
```

```
## [1] 627
```

Bar chart for Rade distribution

```
data %>%
  ggplot(aes(x=Rade)) +
  geom_bar(fill="orange", color="black") +
  theme(axis.text.x = element_text(angle=45, hjust=1)) +
  labs(title="Distribution of Movies by Rade", x="Rade", y="Count")
```



Data Overview: The dataset contains 638 rows and 61 columns. There are both numerical and categorical features. Columns like URL, Name, and Rade are categorical, while others like IMDB_Link, Amtiaz, and Number_People are numerical. The dataset has no missing values after preprocessing.

Numerical Features: IMDB_Link has values ranging from 2.1 to 9.3, with a mean of 6.84. Amtiaz ranges from 17 to 100, with a mean of 84.6. Number_People has high variance, ranging from 1,000 to 998,000. Year values range from 1940 to 2025, with most data points concentrated in recent years. Some numerical columns (like Total_Episodes) have skewed distributions, which might affect modeling.

Categorical Features: Country and Rade should be analyzed further with frequency counts. Many binary genre columns (e.g., Romance, SciFi, Anime) are mostly 0s, meaning most movies don't belong to these genres.

Metric Functions

We choose and define these functions to evaluate out models now on.

```r
MAE <- function(model, x_test, y_test) mean(abs(predict(model, x_test)- y_test))
MSE <- function(model, x_test, y_test) mean((predict(model, x_test)- y_test)^2)
Rsq <- function(model, x_test, y_test) 1 - sum((predict(model, x_test)- y_test)^2)/sum((y_test - mean(y
R2adj <- function(model, x_test, y_test) 1 - ((1 - Rsq(model, x_test, y_test) ) * (nrow(x_test) - 1) /
```

Predicting Amtiaz

From EDA and corr plot, we know no specific feature that has strong linear correlation with out response/target variable `Amtiaz`. This means simple linear regression won't give us a great prediction.

Simple linear regression

```r
set.seed(1)

n <- nrow(data)


train_idx <- sample(1:n, size = 0.9 * n)
test_idx <- setdiff(1:n, train_idx)

train_data <- data[train_idx, ]
test_data  <- data[test_idx, ]


lm_model <- lm(Amtiaz~., data = train_data)
res = summary(lm_model)
AIC(lm_model)
```

```
## [1] 4239.751
```

As of Linear Model summary, we see despite having many features, only 5 prove meaningful and there are a lot of features/parameters.

Feature Selection

Stepwise substest selection

```r
#res_step = step(lm_model, direction = 'both')
best_step_lm <- lm(formula = Amtiaz ~ IMDB_Link + Country + Rade + Is_Doblele +
    Story_Words + Series + Adventure + Comedy + Family + Action +
    ShortFilm + Korean, data = train_data)
res <- summary(best_step_lm)
```

```r
results <- c(
mean(abs(best_step_lm$residuals)),
MAE(best_step_lm, test_data %>% select(!'Amtiaz'), test_data$Amtiaz),
mean(best_step_lm$residuals^2),
MSE(best_step_lm, test_data %>% select(!'Amtiaz'), test_data$Amtiaz),
res$r.squared,
Rsq(best_step_lm, test_data %>% select(!'Amtiaz'), test_data$Amtiaz),
res$adj.r.squared,
R2adj(best_step_lm, test_data %>% select(!'Amtiaz'), test_data$Amtiaz)
)

results <- round(results, 2)

cat(paste(results, collapse = " | "))
```

```
## 6.17 | 6.52 | 83.96 | 78.9 | 0.31 | 0.11 | 0.27 | -10.06
```

Using Lasso Selection

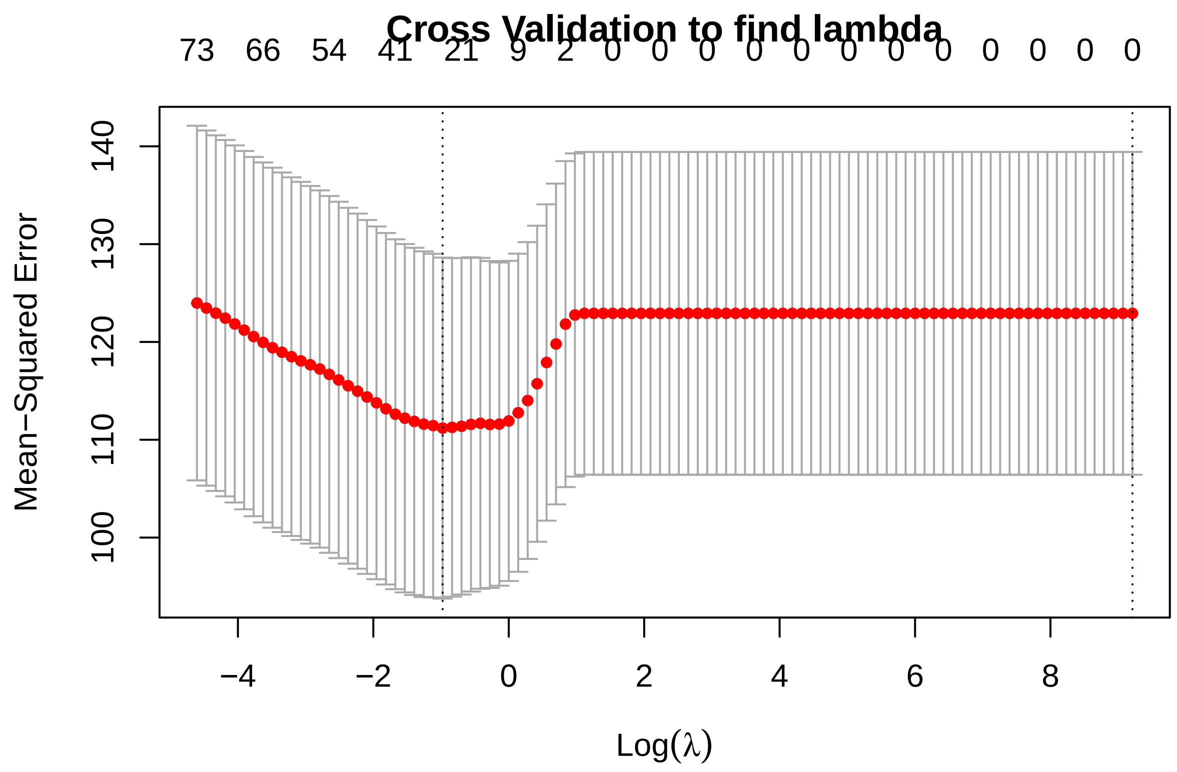
Lasso penalization can be used to select features.

```r
library(glmnet)
```

```
## Loading required package: Matrix

##
## Attaching package: 'Matrix'

## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack

## Loaded glmnet 4.1-8
```

```
x <- model.matrix(Amtiaz ~ ., data)[, -1]
y <- data$Amtiaz
x_train <- x[train_idx, ]
x_test <- x[test_idx, ]
y_train <- y[train_idx]
y_test <- y[test_idx]
```

```
lasso_cv <- cv.glmnet(
  x_train, y_train, alpha = 1, # Indicating Lasso
  lambda = 10^seq(4, -2, length = 100)
  )

plot(lasso_cv, main = "Cross Validation to find lambda")
```



```
best_lambda_lasso <- lasso_cv$lambda.min

cat("Optimal Lambda for Lasso: ", best_lambda_lasso, "\n")
```

```
## Optimal Lambda for Lasso:  0.3764936
```

```
lasso_model <- glmnet(x_train, y_train, alpha = 1, lambda = best_lambda_lasso)

y_pred <- predict(lasso_model, s = best_lambda_lasso, newx = x_test)

results <- c(
MAE(lasso_model,   x_train, y_train),
MAE(lasso_model,   x_test, y_test),
MSE(lasso_model,   x_train, y_train),
MSE(lasso_model,   x_test, y_test),
Rsq(lasso_model,   x_train, y_train),
Rsq(lasso_model,   x_test, y_test),
R2adj(lasso_model, x_train, y_train),
R2adj(lasso_model, x_test, y_test)
)

results <- round(results, 2)

cat(paste(results, collapse = " | "))
```

```
## 6.2 | 5.91 | 89.14 | 72.77 | 0.27 | 0.18 | 0.14 | 2.89
```

```
results <- c(
mean(abs(lm_model$residuals)),
MAE(lm_model, test_data %>% select(!'Amtiaz'), test_data$Amtiaz),
mean(lm_model$residuals^2),
MSE(lm_model, test_data %>% select(!'Amtiaz'), test_data$Amtiaz),
res$r.squared,
Rsq(lm_model, test_data %>% select(!'Amtiaz'), test_data$Amtiaz),
res$adj.r.squared,
R2adj(lm_model, test_data %>% select(!'Amtiaz'), test_data$Amtiaz)
)

results <- round(results, 2)

cat(paste(results, collapse = " | "))
```

```
## 6 | 6.9 | 81.68 | 87.44 | 0.31 | 0.01 | 0.27 | -11.25
```

Model	train MAE	test MAE	train MSE	test MSE	train R^2	test R^2	train Adjusted R^2	test Adjusted R^2
LinReg	6	6.9	81.68	87.44	0.33	0.01	0.23	-11.25
best_step	6.17	6.52	83.96	78.9	0.31	0.11	0.27	-10.06
best_lasso	6.55	6.06	99.04	80.52	0.19	0.09	0.04	3.09

As we see our Linear models (Comparing R^2) are doing no better job than the "Mean predictor" (mean response is the prediction for all). This means features are not predicting the response. So far our models ignored feature interactions, we can turn to models that include interactions well like trees. We know bagging can reduce the variance of trees and boosting can reduce bias.

XG Boost

```
library(xgboost)
```

```
##
## Attaching package: 'xgboost'

## The following object is masked from 'package:dplyr':
##
##     slice
```

```
#tuning parameters nrounds(number of repetitions), eta(learning rate), max_depth(trees depth), gamma(mi
grid <- expand.grid(
  nrounds = c(50, 100, 150),
  eta = c(0.01, 0.1, 0.3),
  max_depth = c(3, 6, 9),
  gamma = c(0, 1, 5),
  colsample_bytree = c(0.5, 0.7, 1),
  min_child_weight = c(1, 3, 5),
  subsample = c(0.6, 0.8, 1)
  )

#10-fold cross-validation
train_control <- trainControl(method = "cv", number = 10)
#xgb_tuned <- train(x = as.matrix(x_train), y = y_train, method = "xgbTree", trControl = train_control,
#best_params <- xgb_tuned$bestTune
#cat("Optimal Parameters for XG Boost : ",  paste(best_params, collapse = ", "), "\n")
cat("Optimal Parameters for XG Boost : ", "50, 3, 0.1, 5, 0.5, 5, 1", "\n")
```

```
## Optimal Parameters for XG Boost :  50, 3, 0.1, 5, 0.5, 5, 1
```

```
xgb_model <- xgboost(data = x_train,
  label = y_train,
            nrounds = 50, #best_params$nrounds,
                eta = 0.1,  #best_params$eta,
          max_depth = 3,#best_params$max_depth,
   min_child_weight = 5,  #best_params$min_child_weight,
          subsample = 0.5,#best_params$subsample,
   colsample_bytree = 1,  #best_params$colsample_bytree,
  objective = "reg:squarederror")
```

```
## [1]  train-rmse:76.361051
## [2]  train-rmse:68.919525
## [3]  train-rmse:62.207770
## [4]  train-rmse:56.180507
## [5]  train-rmse:50.782654
## [6]  train-rmse:45.963207
## [7]  train-rmse:41.708360
## [8]  train-rmse:37.933850
## [9]  train-rmse:34.520527
## [10] train-rmse:31.426356
## [11] train-rmse:28.691359
## [12] train-rmse:26.154651
## [13] train-rmse:23.966591
## [14] train-rmse:22.014074
## [15] train-rmse:20.272872
## [16] train-rmse:18.721471
## [17] train-rmse:17.396517
## [18] train-rmse:16.169573
## [19] train-rmse:15.093156
## [20] train-rmse:14.143467
## [21] train-rmse:13.351637
## [22] train-rmse:12.665086
## [23] train-rmse:12.028746
## [24] train-rmse:11.466131
## [25] train-rmse:11.011587
## [26] train-rmse:10.660174
## [27] train-rmse:10.322994
## [28] train-rmse:10.037061
## [29] train-rmse:9.811565
## [30] train-rmse:9.602965
## [31] train-rmse:9.416670
## [32] train-rmse:9.264992
## [33] train-rmse:9.170243
## [34] train-rmse:9.027349
## [35] train-rmse:8.932723
## [36] train-rmse:8.841477
## [37] train-rmse:8.792340
## [38] train-rmse:8.729397
## [39] train-rmse:8.667685
## [40] train-rmse:8.615669
## [41] train-rmse:8.566691
## [42] train-rmse:8.506988
## [43] train-rmse:8.449321
## [44] train-rmse:8.409287
## [45] train-rmse:8.381865
## [46] train-rmse:8.340125
## [47] train-rmse:8.316202
## [48] train-rmse:8.262338
## [49] train-rmse:8.229597
## [50] train-rmse:8.156256
```

```r
results <- c(
MAE(  xgb_model, x_train, y_train),
MAE(  xgb_model, x_test, y_test),
MSE(  xgb_model, x_train, y_train),
MSE(  xgb_model, x_test, y_test),
Rsq(  xgb_model, x_train, y_train),
Rsq(  xgb_model, x_test, y_test),
R2adj(xgb_model, x_train, y_train),
R2adj(xgb_model, x_test, y_test)
)

results <- round(results, 2)

cat(paste(results, collapse = " | "))
```

```
## 5.42 | 6.47 | 66.52 | 78.38 | 0.46 | 0.11 | 0.35 | 3.03
```

Optimal Parameters for XG Boost : 50, 3, 0.1, 5, 0.5, 5, 1 MSE for XG Boost : 111.038 MAE for XG Boost : 7.076772 R2 for XG Boost : 0.1469957

Model	train MAE	test MAE	train MSE	test MSE	train R^2	test R^2	train Adjusted R^2	test Adjusted R^2
LinReg	6	6.9	81.68	87.44	0.33	0.01	0.23	-11.25
best_step	6.17	6.52	83.96	78.9	0.31	0.11	0.27	-10.06
best_lasso	6.55	6.06	99.04	80.52	0.19	0.09	0.04	3.09
XGBoost	4.47	6.12	46.26	67.16	0.62	0.24	0.55	2.74

Significant Improvement from XGBoost currently the best candidate.

Random forest No Boost

```r
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 4.4.2

## randomForest 4.7-1.2

## Type rfNews() to see new features/changes/bug fixes.

##
## Attaching package: 'randomForest'

## The following object is masked from 'package:gridExtra':
##
##     combine

## The following object is masked from 'package:dplyr':
##
##     combine

## The following object is masked from 'package:ggplot2':
##
##     margin
```

```r
library(tidyverse)
library(caret)

rf_model <- randomForest(Amtiaz ~ ., data = train_data, ntree = 100, mtry = 13, importance = TRUE)
print(rf_model)
```

```
##
## Call:
##  randomForest(formula = Amtiaz ~ ., data = train_data, ntree = 100,      mtry = 13, importance = TRU
##                Type of random forest: regression
##                      Number of trees: 100
## No. of variables tried at each split: 13
##
##           Mean of squared residuals: 103.1444
##                     % Var explained: 15.76
```

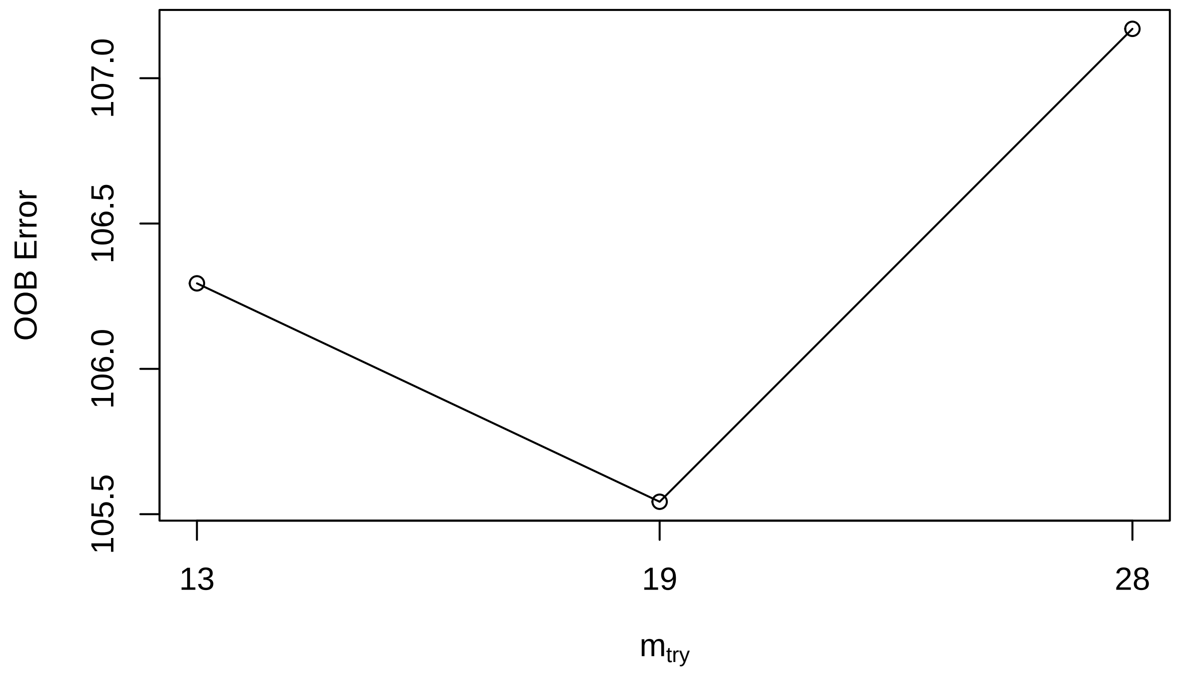
```r
results <- c(
MAE(  rf_model, train_data %>% select(!'Amtiaz'), train_data$Amtiaz),
MAE(  rf_model, test_data %>% select(!'Amtiaz'), test_data$Amtiaz),
MSE(  rf_model, train_data %>% select(!'Amtiaz'), train_data$Amtiaz),
MSE(  rf_model, test_data %>% select(!'Amtiaz'), test_data$Amtiaz),
Rsq(  rf_model, train_data %>% select(!'Amtiaz'), train_data$Amtiaz),
Rsq(  rf_model, test_data %>% select(!'Amtiaz'), test_data$Amtiaz),
R2adj(rf_model, train_data %>% select(!'Amtiaz'), train_data$Amtiaz),
R2adj(rf_model, test_data %>% select(!'Amtiaz'), test_data$Amtiaz)
)

results <- round(results, 2)

cat(paste(results, collapse = " | "))
```

```
## 3 | 5.79 | 21.53 | 69.02 | 0.82 | 0.22 | 0.8 | -8.67
```

Model	train MAE	test MAE	train MSE	test MSE	train R^2	test R^2	train Adjusted R^2	test Adjusted R^2
LinReg	6	6.9	81.68	87.44	0.33	0.01	0.23	-11.25
best_step	6.17	6.52	83.96	78.9	0.31	0.11	0.27	-10.06
best_lasso	6.55	6.06	99.04	80.52	0.19	0.09	0.04	3.09
XGBoost	4.47	6.12	46.26	67.16	0.62	0.24	0.55	2.74
RandomFrst	5.44	6.02	69.57	76.25	0.43	0.14	0.37	-9.69
tuned_RF	2.97	5.78	21.74	66.29	0.82	0.25	0.8	-8.29

```
tuned_rf <- tuneRF(train_data[-which(names(train_data) == "Amtiaz")], train_data$Amtiaz, stepFactor = 1
```

```
## mtry = 19  OOB error = 105.5428
## Searching left ...
## mtry = 13    OOB error = 106.2943
## -0.007119729 0.01
## Searching right ...
## mtry = 28    OOB error = 107.1698
## -0.01541572 0.01
```



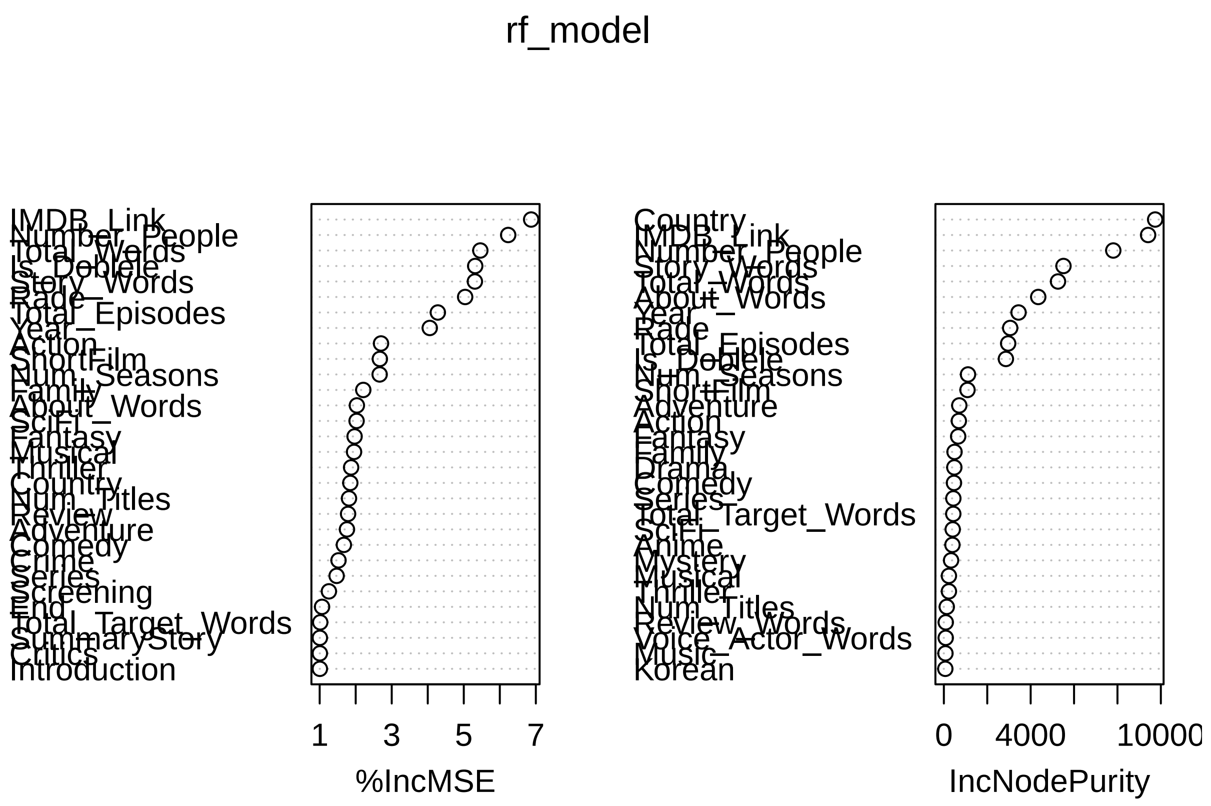
```r
print(tuned_rf)
```

```
##    mtry OOBError
## 13   13 106.2943
## 19   19 105.5428
## 28   28 107.1698
```

```r
importance(rf_model)
```

```
##                            %IncMSE IncNodePurity
## IMDB_Link               6.86980923  9411.9984239
## Number_People           6.23264181  7806.7938056
## Country                 1.84956446  9736.8278368
## Year                    4.05587350  3442.5423781
## Rade                    5.03762175  3056.8343337
## Num_Seasons             2.66482108  1113.0230016
## Total_Episodes          4.27983686  2963.2622250
## Publication            -0.98859051    11.9769972
## VoiceActors            -0.79659928    40.5119817
## Review                  1.78865762     9.0328034
## Tips                   -0.97352916    13.4917334
## End                     1.06564350    15.8053974
## Description             0.00000000     4.9207143
## Characters              0.00000000    11.4865656
## InformativeMessages     0.00000000     0.0000000
## PositiveAndNegative    -0.29170718     3.3603283
## SummaryStory            1.00503782     1.2293843
## Screening                1.25604068    1.7743589
## Critics                  1.00503782    9.6693790
## Conclusion               0.40072835    6.8573436
## Introduction             1.00503782    2.3618634
## Total_Words              5.45865211 5259.7008949
## Num_Titles               1.81379278  134.4616073
## Is_Doblele               5.31713832 2858.1818117
## Total_Target_Words       1.01584488  434.3939671
## About_Words              2.03293047 4349.3516623
## Story_Words              5.30784455 5511.9417265
## Release_Date_Words      -0.31715812   41.3675942
## Review_Words            -1.55472228   87.2551790
## Final_Words             -0.37695609   43.8995272
## Informative_Words        0.00000000    1.4948235
## Positive_Negative_Words -2.74759434    9.4112994
## Summary_Words            0.00000000    0.8829313
## Screening_Words         -0.61840898    3.0183040
## Critics_Words            1.00503782    3.8019608
## Conclusion_Words         0.09299504   19.2463358
## Introduction_Words       0.00000000    2.6948876
## Voice_Actor_Words        0.83594485   85.9195272
## Series                   1.47094883  437.0882420
## Animation                0.00000000    3.1083716
## Western                  0.00000000   19.3839333
## Adventure                1.75718091  711.1920082
## Comedy                   1.67118690  466.0390808
## Family                   2.21042966  491.3513558
## Fantasy                  1.96883284  658.7675112
## Mystery                  0.56229825  331.6981100
## Action                   2.70214819  690.2429097
## Romance                  0.53561455   48.1307434
## Drama                    0.68650845  482.0666584
## SciFi                    2.02440705  408.8109623
## ShortFilm                2.67019749 1088.7815991
## Crime                    1.52287050   50.9020843
## Musical                  1.95431528  233.0723932
## Korean                  -0.07271798   66.3105861
## Thriller                 1.87300207  233.0573546
## Anime                   -0.47305956  395.3543319
## Music                   -1.33364117   73.1127324
```

```r
varImpPlot(rf_model)
```



SVR

This Model needs separate data prep

```
library(e1071)
```

```
## Warning: package 'e1071' was built under R version 4.4.2
```

```
##
## Attaching package: 'e1071'
```

```
## The following objects are masked from 'package:moments':
##
##     kurtosis, moment, skewness
```

```
library(caret)
library(dplyr)
```

```
data = 'clean_data.csv'
data <- read.csv(data, stringsAsFactors = TRUE)

data %>%
  group_by(Country) %>%
  filter(n() > 1) -> data
nrow(data)
```

```
## [1] 627
```

```r
# Remove the first two columns
data %>% select(!c(URL, Name)) -> data
```

```r
# Convert columns
data$Country <- as.factor(data$Country)
data$Rade <- as.factor(data$Rade)
data$Amtiaz <- as.numeric(data$Amtiaz)
data$Year <- as.numeric(data$Year)
data$IMDB_Link <- as.numeric(data$IMDB_Link)
# Convert categorical variables to dummy variables
data <- dummyVars(~ ., data = data) %>% predict(data) %>% as.data.frame()
```

Split the data into features and target

```r
target <- "Amtiaz"
predictors <- setdiff(names(data), target)
```

train,test

```r
set.seed(1)
train_index <- sample(1:nrow(data), size = 0.7 * nrow(data))
svr_train_data <- data[train_index, ]
svr_test_data <- data[-train_index, ]

# Scale numerical features
preproc <- preProcess(svr_train_data[, predictors], method = c("center", "scale"))
train_data_scaled <- predict(preproc, svr_train_data)
test_data_scaled <- predict(preproc, svr_test_data)

train_data_scaled$Amtiaz <- svr_train_data$Amtiaz
test_data_scaled$Amtiaz <- svr_test_data$Amtiaz
```

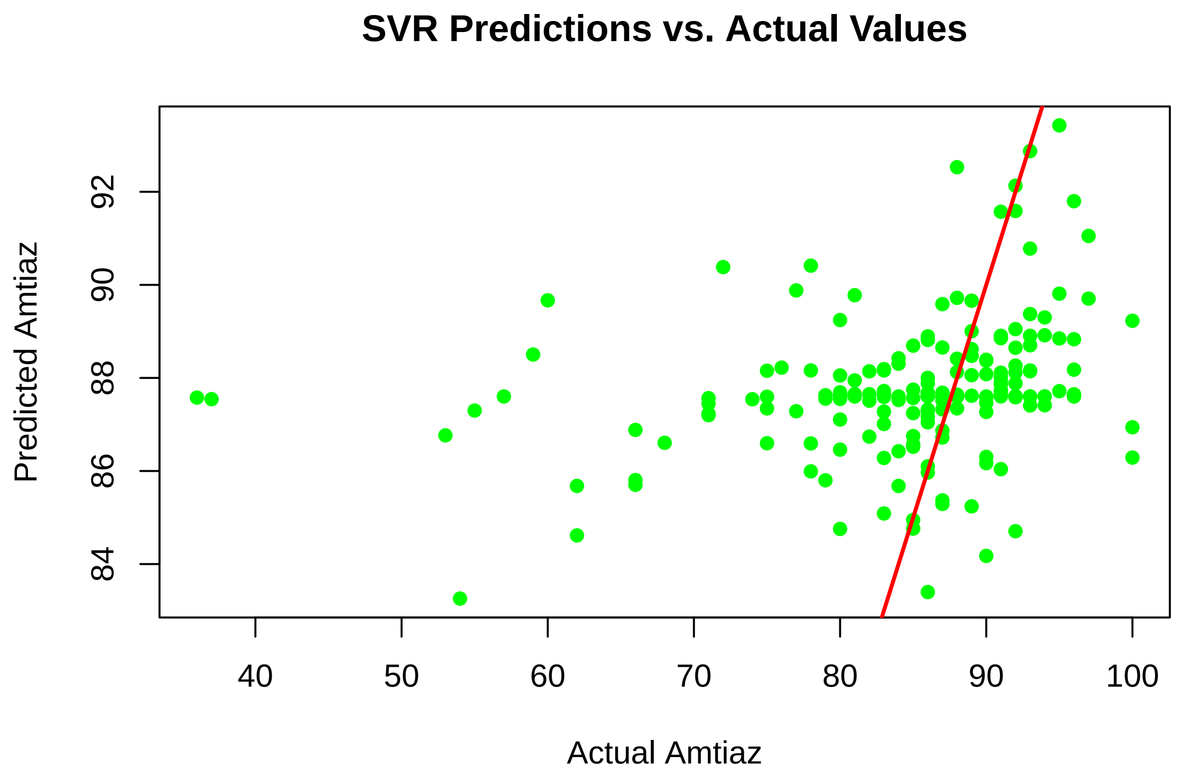
```r
svr_model <- svm(Amtiaz ~ ., data = train_data_scaled, kernel = "radial", cost = 1, gamma = 0.1)

plot(svr_model, train_data_scaled)
summary(svr_model)
```

```
##
## Call:
## svm(formula = Amtiaz ~ ., data = train_data_scaled, kernel = "radial",
##     cost = 1, gamma = 0.1)
##
##
## Parameters:
##    SVM-Type:  eps-regression
##  SVM-Kernel:  radial
##        cost:  1
##       gamma:  0.1
##     epsilon:  0.1
##
##
## Number of Support Vectors:  433
```

```r
results <- c(
MAE(  svr_model, train_data_scaled %>% select(!'Amtiaz'), train_data_scaled$Amtiaz),
MAE(  svr_model,  test_data_scaled %>% select(!'Amtiaz'),  test_data_scaled$Amtiaz),
MSE(  svr_model, train_data_scaled %>% select(!'Amtiaz'), train_data_scaled$Amtiaz),
MSE(  svr_model,  test_data_scaled %>% select(!'Amtiaz'),  test_data_scaled$Amtiaz),
Rsq(  svr_model, train_data_scaled %>% select(!'Amtiaz'), train_data_scaled$Amtiaz),
Rsq(  svr_model,  test_data_scaled %>% select(!'Amtiaz'),  test_data_scaled$Amtiaz),
R2adj(svr_model, train_data_scaled %>% select(!'Amtiaz'), train_data_scaled$Amtiaz),
R2adj(svr_model,  test_data_scaled %>% select(!'Amtiaz'),  test_data_scaled$Amtiaz)
)

results <- round(results, 2)

cat(paste(results, collapse = " | "))
```

```
## 5.82 | 6.43 | 118.31 | 105.28 | 0.07 | -0.03 | -0.18 | -0.99
```

```
predictions <- predict(svr_model,test_data_scaled)
# Plot actual vs predicted values
plot(test_data_scaled$Amtiaz, predictions,
     xlab = "Actual Amtiaz", ylab = "Predicted Amtiaz",
     main = "SVR Predictions vs. Actual Values",
     col = "green", pch = 16)
abline(0, 1, col = "red", lwd = 2)
```



```
tuned <- tune(svm, Amtiaz ~ ., data = train_data_scaled, kernel = "radial", ranges = list(cost =c(0.1,
best_model <- tuned$best.model
summary(best_model)
```

```
##
```

```
## Call:
## best.tune(METHOD = svm, train.x = Amtiaz ~ ., data = train_data_scaled,
##     ranges = list(cost = c(0.1, 1, 10), gamma = c(0.01, 0.1, 1)),
##     kernel = "radial")
##
##
## Parameters:
##    SVM-Type:  eps-regression
##  SVM-Kernel:  radial
##        cost:  10
##       gamma:  0.01
##     epsilon:  0.1
##
##
## Number of Support Vectors:  434
```

```
results <- c(
MAE(  best_model, train_data_scaled %>% select(!'Amtiaz'), train_data_scaled$Amtiaz),
MAE(  best_model,  test_data_scaled %>% select(!'Amtiaz'),  test_data_scaled$Amtiaz),
MSE(  best_model, train_data_scaled %>% select(!'Amtiaz'), train_data_scaled$Amtiaz),
MSE(  best_model,  test_data_scaled %>% select(!'Amtiaz'),  test_data_scaled$Amtiaz),
Rsq(  best_model, train_data_scaled %>% select(!'Amtiaz'), train_data_scaled$Amtiaz),
Rsq(  best_model,  test_data_scaled %>% select(!'Amtiaz'),  test_data_scaled$Amtiaz),
R2adj(best_model, train_data_scaled %>% select(!'Amtiaz'), train_data_scaled$Amtiaz),
R2adj(best_model,  test_data_scaled %>% select(!'Amtiaz'),  test_data_scaled$Amtiaz)
)

results <- round(results, 2)

cat(paste(results, collapse = " | "))
```

```
## 4.55 | 6.39 | 91.22 | 97.47 | 0.28 | 0.05 | 0.09 | -0.85
```

Model	train MAE	test MAE	train MSE	test MSE	train R^2	test R^2	train Adjusted R^2	test Adjusted R^2
LinReg	6	6.9	81.68	87.44	0.33	0.01	0.23	-11.25
best_step	6.17	6.52	83.96	78.9	0.31	0.11	0.27	-10.06
best_lasso	6.55	6.06	99.04	80.52	0.19	0.09	0.04	3.09
XGBoost	4.47	6.12	46.26	67.16	0.62	0.24	0.55	2.74
RandomFrst	5.44	6.02	69.57	76.25	0.43	0.14	0.37	-9.69
tuned_RF	2.97	5.78	21.74	66.29	0.82	0.25	0.8	-8.29
SVR	5.82	6.43	118.31	105.28	0.07	-0.03	-0.18	-0.99
tuned SVR	4.55	6.39	91.22	97.47	0.28	0.05	0.09	-0.85

Neural Network

An extensive notebook on fitting a neural network is given in python.

```
library(keras)
NN_model = keras_model_sequential() %>%
  layer_dense(units = 128, activation = "relu", input_shape = dim(x_train)[2] ) %>%
  layer_dense(units = 60, activation = "relu",) %>%
  layer_dense(units = 15, activation = "relu",) %>%
  layer_dense(units = 1)
```

```r
NN_model %>% compile(
  optimizer = "adam",
  loss = "mse"
)

summary(NN_model)
```

```
## Model: "sequential"
## ________________________________________________________________________________
##  Layer (type)                       Output Shape                    Param #
## ================================================================================
##  dense_3 (Dense)                    (None, 128)                     11520
##  dense_2 (Dense)                    (None, 60)                      7740
##  dense_1 (Dense)                    (None, 15)                      915
##  dense (Dense)                      (None, 1)                       16
## ================================================================================
## Total params: 20,191
## Trainable params: 20,191
## Non-trainable params: 0
## ________________________________________________________________________________
```

```r
history = NN_model %>% fit(
  x_train, y_train,
  epochs = 50, batch_size = 16,
  validation_data = list(x_test, y_test),
  verbose = 1
)
```

```
## Epoch 1/50
##
##  1/36 [..............................] - ETA: 32s - loss: 29078564.0000
## 20/36 [===============>..............] - ETA: 0s - loss: 7281915.0000
## 36/36 [==============================] - 1s 11ms/step - loss: 4494603.0000 - val_loss: 745295.7500
## Epoch 2/50
##
##  1/36 [..............................] - ETA: 0s - loss: 768078.1250
## 24/36 [===================>..........] - ETA: 0s - loss: 173454.9844
## 36/36 [==============================] - 0s 4ms/step - loss: 120478.1875 - val_loss: 5507.2095
## Epoch 3/50
##
##  1/36 [..............................] - ETA: 0s - loss: 7172.5132
## 23/36 [==================>...........] - ETA: 0s - loss: 2541.3982
## 36/36 [==============================] - 0s 4ms/step - loss: 2712.5859 - val_loss: 3588.6899
## Epoch 4/50
##
##  1/36 [..............................] - ETA: 0s - loss: 2830.7656
## 23/36 [==================>...........] - ETA: 0s - loss: 1890.5844
## 36/36 [==============================] - 0s 4ms/step - loss: 1594.3715 - val_loss: 1347.7404
## Epoch 5/50
##
##  1/36 [..............................] - ETA: 0s - loss: 3066.1587
## 22/36 [=================>............] - ETA: 0s - loss: 1138.4617
## 36/36 [==============================] - 0s 4ms/step - loss: 1084.3005 - val_loss: 683.6730
## Epoch 6/50
```

```
##
##  1/36 [..............................] - ETA: 0s - loss: 1064.8101
## 22/36 [=================>............] - ETA: 0s - loss: 925.2203
## 36/36 [==============================] - 0s 4ms/step - loss: 692.5080 - val_loss: 231.7179
## Epoch 7/50
##
##  1/36 [..............................] - ETA: 0s - loss: 214.1384
## 21/36 [================>.............] - ETA: 0s - loss: 278.9741
## 36/36 [==============================] - 0s 4ms/step - loss: 377.0692 - val_loss: 358.5136
## Epoch 8/50
##
##  1/36 [..............................] - ETA: 0s - loss: 735.0883
## 22/36 [=================>............] - ETA: 0s - loss: 1392.7395
## 36/36 [==============================] - 0s 4ms/step - loss: 1138.3546 - val_loss: 452.8837
## Epoch 9/50
##
##  1/36 [..............................] - ETA: 0s - loss: 293.6479
## 23/36 [==================>...........] - ETA: 0s - loss: 1033.0961
## 36/36 [==============================] - 0s 4ms/step - loss: 806.8029 - val_loss: 192.1606
## Epoch 10/50
##
##  1/36 [..............................] - ETA: 0s - loss: 123.0333
## 24/36 [===================>..........] - ETA: 0s - loss: 329.8636
## 36/36 [==============================] - 0s 3ms/step - loss: 475.9560 - val_loss: 258.0565
## Epoch 11/50
##
##  1/36 [..............................] - ETA: 0s - loss: 186.2753
## 25/36 [===================>..........] - ETA: 0s - loss: 1805.6366
## 36/36 [==============================] - 0s 3ms/step - loss: 1454.9088 - val_loss: 436.7007
## Epoch 12/50
##
##  1/36 [..............................] - ETA: 0s - loss: 654.5989
## 24/36 [===================>..........] - ETA: 0s - loss: 1287.7448
## 36/36 [==============================] - 0s 3ms/step - loss: 1580.7151 - val_loss: 170.8711
## Epoch 13/50
##
##  1/36 [..............................] - ETA: 0s - loss: 171.0851
## 23/36 [==================>...........] - ETA: 0s - loss: 161.1051
## 36/36 [==============================] - 0s 3ms/step - loss: 215.7298 - val_loss: 1056.9932
## Epoch 14/50
##
##  1/36 [..............................] - ETA: 0s - loss: 707.8987
## 24/36 [===================>..........] - ETA: 0s - loss: 6477.0435
## 36/36 [==============================] - 0s 4ms/step - loss: 6267.4810 - val_loss: 2589.7969
## Epoch 15/50
##
##  1/36 [..............................] - ETA: 0s - loss: 3458.0942
## 24/36 [===================>..........] - ETA: 0s - loss: 58314.9570
## 36/36 [==============================] - 0s 3ms/step - loss: 116668.3203 - val_loss: 898.3849
## Epoch 16/50
##
##  1/36 [..............................] - ETA: 0s - loss: 1587.4309
## 24/36 [===================>..........] - ETA: 0s - loss: 210383.2344
## 36/36 [==============================] - 0s 3ms/step - loss: 175543.5000 - val_loss: 4894.9971
## Epoch 17/50
##
##  1/36 [..............................] - ETA: 0s - loss: 3167.6975
## 25/36 [===================>..........] - ETA: 0s - loss: 14458.3486
## 36/36 [==============================] - 0s 3ms/step - loss: 10396.0674 - val_loss: 774.8774
## Epoch 18/50
##
##  1/36 [..............................] - ETA: 0s - loss: 251.7865
## 25/36 [===================>..........] - ETA: 0s - loss: 1090.9047
## 36/36 [==============================] - 0s 3ms/step - loss: 1224.0712 - val_loss: 2587.9136
## Epoch 19/50
##
##  1/36 [..............................] - ETA: 0s - loss: 457.5629
## 27/36 [=====================>........] - ETA: 0s - loss: 3455.2871
## 36/36 [==============================] - 0s 3ms/step - loss: 3598.9211 - val_loss: 263.5042
## Epoch 20/50
##
##  1/36 [..............................] - ETA: 0s - loss: 326.7899
## 27/36 [=====================>........] - ETA: 0s - loss: 648.5776
## 36/36 [==============================] - 0s 3ms/step - loss: 584.6530 - val_loss: 597.0762
## Epoch 21/50
##
##  1/36 [..............................] - ETA: 0s - loss: 471.8143
## 30/36 [========================>.....] - ETA: 0s - loss: 345.9782
## 36/36 [==============================] - 0s 3ms/step - loss: 318.3501 - val_loss: 200.6561
## Epoch 22/50
##
##  1/36 [..............................] - ETA: 0s - loss: 86.5989
## 31/36 [========================>.....] - ETA: 0s - loss: 381.1975
## 36/36 [==============================] - 0s 3ms/step - loss: 407.5741 - val_loss: 1374.4122
## Epoch 23/50
##
##  1/36 [..............................] - ETA: 0s - loss: 1184.1101
## 31/36 [========================>.....] - ETA: 0s - loss: 1403.6460
## 36/36 [==============================] - 0s 3ms/step - loss: 1255.8314 - val_loss: 226.9471
## Epoch 24/50
##
##  1/36 [..............................] - ETA: 0s - loss: 196.5114
## 30/36 [========================>.....] - ETA: 0s - loss: 637.4858
## 36/36 [==============================] - 0s 3ms/step - loss: 583.8217 - val_loss: 784.5626
## Epoch 25/50
##
##  1/36 [..............................] - ETA: 0s - loss: 1148.6637
## 33/36 [==========================>...] - ETA: 0s - loss: 342.2159
## 36/36 [==============================] - 0s 2ms/step - loss: 381.8099 - val_loss: 661.2903
## Epoch 26/50
##
##  1/36 [..............................] - ETA: 0s - loss: 735.9059
## 32/36 [=========================>....] - ETA: 0s - loss: 664.6176
## 36/36 [==============================] - 0s 2ms/step - loss: 648.5856 - val_loss: 1833.8574
## Epoch 27/50
##
##  1/36 [..............................] - ETA: 0s - loss: 2633.5972
## 30/36 [========================>.....] - ETA: 0s - loss: 4300.5488
## 36/36 [==============================] - 0s 3ms/step - loss: 3913.9556 - val_loss: 3697.2305
## Epoch 28/50
##
##  1/36 [..............................] - ETA: 0s - loss: 2024.4423
## 32/36 [=========================>....] - ETA: 0s - loss: 3024006.2500
## 36/36 [==============================] - 0s 2ms/step - loss: 3121405.0000 - val_loss: 10255892.0000
## Epoch 29/50
##
##  1/36 [..............................] - ETA: 0s - loss: 11418333.0000
## 32/36 [=========================>....] - ETA: 0s - loss: 1902047.2500
## 36/36 [==============================] - 0s 3ms/step - loss: 1751245.8750 - val_loss: 1653905.7500
## Epoch 30/50
##
##  1/36 [..............................] - ETA: 0s - loss: 3131980.5000
## 32/36 [=========================>....] - ETA: 0s - loss: 674016.3750
## 36/36 [==============================] - 0s 3ms/step - loss: 616324.9375 - val_loss: 43310.9609
## Epoch 31/50
##
##  1/36 [..............................] - ETA: 0s - loss: 27965.7246
## 31/36 [========================>.....] - ETA: 0s - loss: 40866.1992
## 36/36 [==============================] - 0s 3ms/step - loss: 36623.3398 - val_loss: 431.8567
## Epoch 32/50
##
##  1/36 [..............................] - ETA: 0s - loss: 504.9573
## 29/36 [=======================>......] - ETA: 0s - loss: 8026.5073
## 36/36 [==============================] - 0s 3ms/step - loss: 7965.6030 - val_loss: 4716.0317
## Epoch 33/50
##
##  1/36 [..............................] - ETA: 0s - loss: 5736.7729
## 31/36 [========================>.....] - ETA: 0s - loss: 20966.1953
## 36/36 [==============================] - 0s 3ms/step - loss: 18897.7832 - val_loss: 5787.0298
## Epoch 34/50
##
##  1/36 [..............................] - ETA: 0s - loss: 6869.6787
## 31/36 [========================>.....] - ETA: 0s - loss: 3750.0854
## 36/36 [==============================] - 0s 3ms/step - loss: 3423.6714 - val_loss: 1830.5870
## Epoch 35/50
##
##  1/36 [..............................] - ETA: 0s - loss: 2472.0723
## 30/36 [========================>.....] - ETA: 0s - loss: 728.5105
## 36/36 [==============================] - 0s 3ms/step - loss: 794.0740 - val_loss: 225.3091
## Epoch 36/50
##
##  1/36 [..............................] - ETA: 0s - loss: 163.2662
## 32/36 [=========================>....] - ETA: 0s - loss: 289.3835
## 36/36 [==============================] - 0s 3ms/step - loss: 278.3832 - val_loss: 226.4904
## Epoch 37/50
##
##  1/36 [..............................] - ETA: 0s - loss: 102.0483
## 33/36 [==========================>...] - ETA: 0s - loss: 188.7522
## 36/36 [==============================] - 0s 3ms/step - loss: 182.5669 - val_loss: 122.6891
## Epoch 38/50
##
##  1/36 [..............................] - ETA: 0s - loss: 66.6961
## 35/36 [============================>.] - ETA: 0s - loss: 213.7232
## 36/36 [==============================] - 0s 3ms/step - loss: 212.6218 - val_loss: 118.9804
## Epoch 39/50
##
##  1/36 [..............................] - ETA: 0s - loss: 117.8575
## 33/36 [==========================>...] - ETA: 0s - loss: 151.5381
## 36/36 [==============================] - 0s 2ms/step - loss: 167.3358 - val_loss: 184.7752
## Epoch 40/50
##
##  1/36 [..............................] - ETA: 0s - loss: 160.6389
## 32/36 [=========================>....] - ETA: 0s - loss: 153.2317
## 36/36 [==============================] - 0s 2ms/step - loss: 151.8735 - val_loss: 598.6144
## Epoch 41/50
##
##  1/36 [..............................] - ETA: 0s - loss: 745.3481
## 32/36 [=========================>....] - ETA: 0s - loss: 433.1099
## 36/36 [==============================] - 0s 3ms/step - loss: 474.7989 - val_loss: 999.3154
## Epoch 42/50
##
##  1/36 [..............................] - ETA: 0s - loss: 668.2374
## 33/36 [==========================>...] - ETA: 0s - loss: 947.2170
## 36/36 [==============================] - 0s 3ms/step - loss: 940.6713 - val_loss: 236.4437
## Epoch 43/50
##
##  1/36 [..............................] - ETA: 0s - loss: 387.2244
## 32/36 [=========================>....] - ETA: 0s - loss: 1348.8396
## 36/36 [==============================] - 0s 2ms/step - loss: 1244.7797 - val_loss: 474.3731
## Epoch 44/50
##
##  1/36 [..............................] - ETA: 0s - loss: 287.1522
## 32/36 [=========================>....] - ETA: 0s - loss: 207.6971
## 36/36 [==============================] - 0s 2ms/step - loss: 204.5589 - val_loss: 105.2682
## Epoch 45/50
##
##  1/36 [..............................] - ETA: 0s - loss: 46.7672
## 32/36 [=========================>....] - ETA: 0s - loss: 179.8824
## 36/36 [==============================] - 0s 3ms/step - loss: 173.5233 - val_loss: 179.1667
## Epoch 46/50
##
##  1/36 [..............................] - ETA: 0s - loss: 212.0311
## 31/36 [========================>.....] - ETA: 0s - loss: 169.2046
## 36/36 [==============================] - 0s 3ms/step - loss: 173.0320 - val_loss: 259.3883
## Epoch 47/50
##
##  1/36 [..............................] - ETA: 0s - loss: 177.6667
## 32/36 [=========================>....] - ETA: 0s - loss: 160.5437
## 36/36 [==============================] - 0s 3ms/step - loss: 168.3491 - val_loss: 129.4354
## Epoch 48/50
##
##  1/36 [..............................] - ETA: 0s - loss: 178.1673
## 32/36 [=========================>....] - ETA: 0s - loss: 158.7173
## 36/36 [==============================] - 0s 3ms/step - loss: 164.0982 - val_loss: 103.2517
## Epoch 49/50
##
```

```
##  1/36 [..............................] - ETA: 0s - loss: 129.9230
## 34/36 [===========================>..] - ETA: 0s - loss: 197.9192
## 36/36 [==============================] - 0s 2ms/step - loss: 200.3070 - val_loss: 131.4492
## Epoch 50/50
##
##  1/36 [..............................] - ETA: 0s - loss: 95.2978
## 31/36 [========================>.....] - ETA: 0s - loss: 172.3391
## 36/36 [==============================] - 0s 3ms/step - loss: 162.2402 - val_loss: 231.2364
```

```
results <- c(
MAE(  NN_model, x_train, y_train),
MAE(  NN_model, x_test, y_test),
MSE(  NN_model, x_train, y_train),
MSE(  NN_model, x_test, y_test),
Rsq(  NN_model, x_train, y_train),
Rsq(  NN_model, x_test, y_test),
R2adj(NN_model, x_train, y_train),
R2adj(NN_model, x_test, y_test)
)
```

```
## 18/18 - 0s - 114ms/epoch - 6ms/step
## 2/2 - 0s - 22ms/epoch - 11ms/step
## 18/18 - 0s - 38ms/epoch - 2ms/step
## 2/2 - 0s - 22ms/epoch - 11ms/step
## 18/18 - 0s - 37ms/epoch - 2ms/step
## 2/2 - 0s - 22ms/epoch - 11ms/step
## 18/18 - 0s - 36ms/epoch - 2ms/step
## 2/2 - 0s - 23ms/epoch - 12ms/step
```

```
results <- round(results, 2)

cat(paste(results, collapse = " | "))
```

```
## 12.04 | 11.36 | 328.26 | 231.23 | -1.68 | -1.61 | -2.18 | 7
```

Model	train MAE	test MAE	train MSE	test MSE	train R^2	test R^2	train Adjusted R^2	test Adjusted R^2
LinReg	6	6.9	81.68	87.44	0.33	0.01	0.23	-11.25
best_step	6.17	6.52	83.96	78.9	0.31	0.11	0.27	-10.06
best_lasso	6.55	6.06	99.04	80.52	0.19	0.09	0.04	3.09
XGBoost	4.47	6.12	46.26	67.16	0.62	0.24	0.55	2.74
RandomFrst	5.44	6.02	69.57	76.25	0.43	0.14	0.37	-9.69
tuned_RF	2.97	5.78	21.74	66.29	0.82	0.25	0.8	-8.29
SVR	5.82	6.43	118.31	105.28	0.07	-0.03	-0.18	-0.99
tuned SVR	4.55	6.39	91.22	97.47	0.28	0.05	0.09	-0.85
Neural Net	12.13	13.29	266.01	301.92	-1.17	-2.41	-1.58	8.84

Neural Net when not tuned, performs worse.