PSTAT 131 Final Project: Model comparison for predicting the salary of a data science job

Nicholas Axl Andrian

2023-11-23

Dataset used: https://www.kaggle.com/datasets/arnabchaki/data-science-salaries-2023

In this project, we will be fitting several different machine learning algorithms to find out which method of prediction is the most accurate in getting the predicted salary(in usd).

About the data set's variables (excerpt from the kaggle site)

- work_year: The year the salary was paid.
- experience_level: The experience level in the job during the year
- employment_type: The type of employment for the role
- job_title: The role worked in during the year.
- salary: The total gross salary amount paid.
- salary_currency: The currency of the salary paid as an ISO 4217 currency code.
- salary_in_usd: The salary in USD
- employee_residence: Employee's primary country of residence in during the work + year as an ISO 3166 country code.
- remote_ratio: The overall amount of work done remotely
- company_location: The country of the employer's main office or contracting branch
- company_size: The median number of people that worked for the company during the year

```r
library(dplyr)
library(randomForest)
library(gbm)
library(ISLR)
library(tree)
library(tidyverse)
library(ggplot2)
library(gridExtra)
library(class)
library(FNN)
library(tibble)
library(recipes)
```

PSTAT 131 helper for Cross Validation

```r
do.chunk <- function(chunkid, folddef, Xdat, Ydat, ...){
# Get training index
train = (folddef!=chunkid)
# Get training set by the above index
Xtr = Xdat[train,]
```

```r
# Get responses in training set
Ytr = Ydat[train]
# Get validation set
Xvl = Xdat[!train,]
# Get responses in validation set
Yvl = Ydat[!train]
# Predict training labels
predYtr = knn(train=Xtr, test=Xtr, cl=Ytr, ...)
# Predict validation labels
predYvl = knn(train=Xtr, test=Xvl, cl=Ytr, ...)
data.frame(fold = chunkid,
train.error = mean(predYtr != Ytr), # Training error for each fold
val.error = mean(predYvl != Yvl)) # Validation error for each fold
}
```

Part 1: Exploratory Data Analysis

Loading the dataset

```r
salaries <- read.csv("ds_salaries.csv")
head(salaries)
```

Checking the structure of the dataset

```r
str(salaries)
```

Already we can see an issue that needs to be worked on. Several variables seem to supposedly be read in as factors. We will finish conducting checks on the dataset before converting said columns.

Checking the summary of the dataset

```r
summary(salaries)
```

Checking for null values

```r
colSums(is.na(salaries))
```

```
##          work_year     experience_level     employment_type           job_title
##                  0                    0                   0                   0
##             salary     salary_currency       salary_in_usd employee_residence
##                  0                    0                   0                   0
##       remote_ratio     company_location        company_size
##                  0                    0                   0
```

Fortunately, we have no null values so imputing is not required

Checking potential factor columns for their unique values

```r
factor_cols <- salaries[, c(1, 2, 3, 4, 6, 8, 10, 11)]

# finding unique values, referenced code from https://www.kaggle.com/code/abdulfaheem11/data-science-sa

# output ommitted to prevent too much space being taken up
sapply(factor_cols, function(col) unique(col))
```

Changing said variables to become factors

```
salaries[, c(1, 2, 3, 4, 6, 8, 10, 11)] <- lapply(factor_cols, factor)
str(salaries)
```

Visualization to search for patterns with regards to the salary_in_usd

Prioritizing focus on work_year, experience_level, employment_type, job_title, employee_residence, remote_ratio, company_location, company_size

```
yearplot <- ggplot(salaries, aes(x = work_year, y = salary_in_usd)) +
  geom_point(color = "red", size = 3) +
  labs(x = "Work Year", y = "Salary in USD", title = "Salary vs Work Year")
expplot <- ggplot(salaries, aes(x = experience_level, y = salary_in_usd)) +
  geom_boxplot(fill = "skyblue") +
  labs(x = "Experience Level", y = "Salary in USD", title = "Salary vs Experience Level")
grid.arrange(yearplot, expplot, ncol = 2)
```


- We can see that the average salary in usd increases as the years go by, as the line congests further upwards towards the end.
- Experience level does not really show much of a trend as it goes towards seniority, We can tell though that EX has the highest average and MI has the highest peak

```r
employplot <- ggplot(salaries, aes(x = employment_type, y = salary_in_usd)) +
  geom_boxplot(fill = "skyblue") +
  labs(x = "Employment Type", y = "Salary in USD", title = "Salary vs Employment Type")
remoteplot <- ggplot(salaries, aes(x = remote_ratio, y = salary_in_usd)) +
  geom_point(color = "red", size = 3, shape = 19) +
  labs(x = "Remote Ratio", y = "Salary in USD", title = "Salary vs Remote Ratio")
grid.arrange(employplot, remoteplot, ncol = 2)
```


+ In employment type, FT has the highest average as well as higher peaks + Remote ratio consists of 0, 50 and 100. The highest points as well as average are in the order 0>100>50

```r
salaryplot <- ggplot(salaries, aes(x = salary, y = salary_in_usd)) +
  geom_point() +
  labs(title = "Salary vs. Salary in USD", x = "Salary", y = "Salary in USD")

# Salary vs. Salary Currency plot
currencyplot <- ggplot(salaries, aes(x = salary_in_usd, y = salary_currency)) +
  geom_point() +
  labs(title = "Salary(USD) vs Currency", x = "Salary", y = "Salary Currency")

# Combine plots into a grid using facet_grid
grid.arrange(salaryplot, currencyplot, ncol = 2)
```
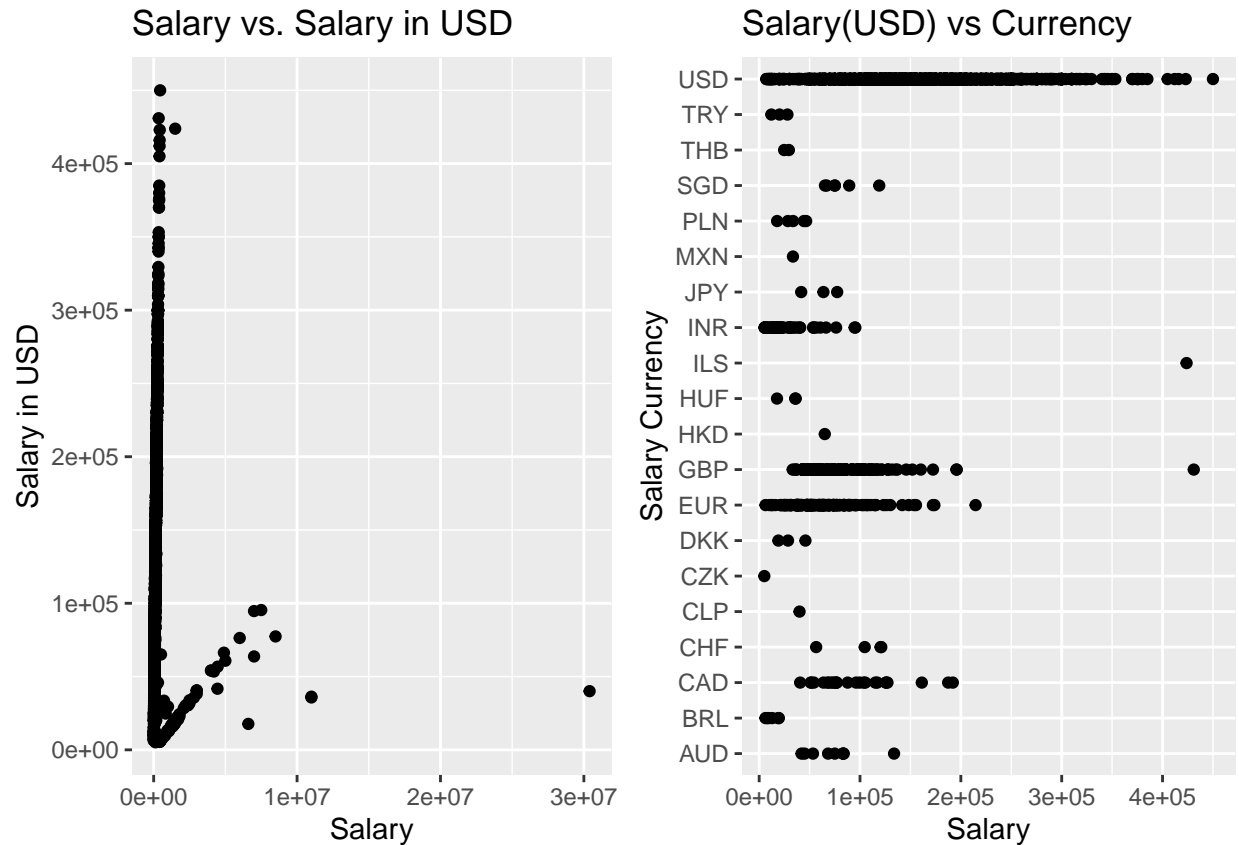
+ In the first plot, we see that there is a point that is way further to the right than any of the other points. We will have to check that out right after this + On the 2nd plot, aside from seeing that we might have the most observations in USD, we can also see that it has the highest salary just based on numbers

Checking out the "outlier" observation

```
max(salaries$salary)
```

```
## [1] 30400000
```

```
outlierindex <- which(salaries$salary == max(salaries$salary))
salaries$salary_currency[outlierindex]
```

```
## [1] CLP
## 20 Levels: AUD BRL CAD CHF CLP CZK DKK EUR GBP HKD HUF ILS INR JPY MXN ... USD
```

It seems to be an observation from the CLP currency. and connecting it back to the graph, we can see that it also only has that one observation. I will deem it insignificant for now, to avoid a potential leverage point affecting the models I will be removing said observation.
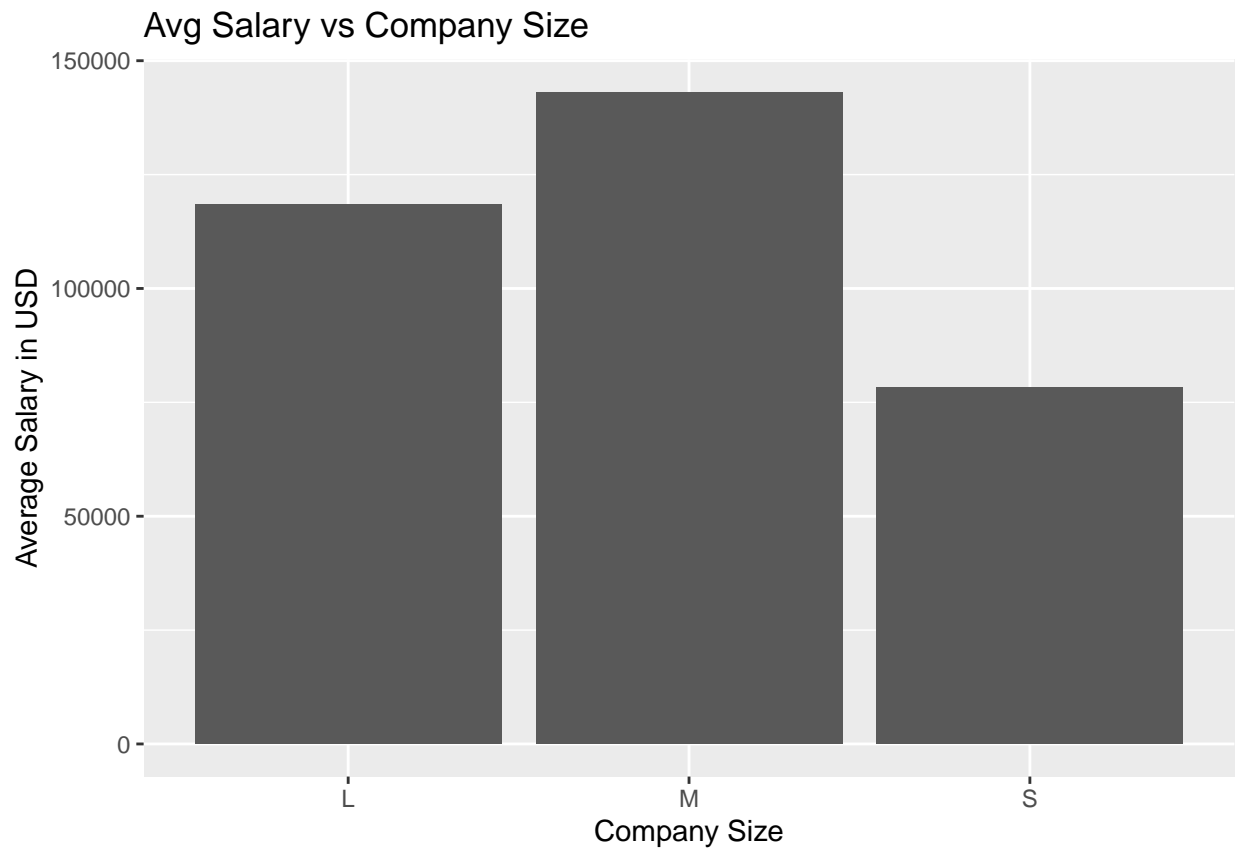
```
salaries <- salaries[-outlierindex, ]
```

Plot showing salary vs Company Size

```r
usd_salary_by_size <- salaries%>%
  group_by(company_size)%>%
  summarise(Avg_sal=mean(salary_in_usd))

sizeplot <- ggplot(usd_salary_by_size, aes(x=company_size, y=Avg_sal)) +
  geom_col() +
  labs(title='Avg Salary vs Company Size', x='Company Size', y='Average Salary in USD')
sizeplot
```


+ From this plot we can also see that medium sized companies pay the largest on average, followed by large then small

Tabling the 6 highest and lowest paying jobs

```r
top_6_job_salaries<-salaries%>%
  group_by(job_title)%>%
  summarise(Avg_Sal=mean(salary_in_usd))%>%
  arrange(desc(Avg_Sal))%>%
  head()
top_6_job_salaries
bottom_6_job_salaries<-salaries%>%
  group_by(job_title)%>%
  summarise(Avg_Sal=mean(salary_in_usd))%>%
  arrange(Avg_Sal)%>%
  head()
bottom_6_job_salaries
```

Plotting the 6 highest and lowest paying jobs

```
top6jobplot <- ggplot(top_6_job_salaries, aes(x = reorder(job_title, Avg_Sal), y = Avg_Sal)) +
  geom_bar(stat = "identity", fill = "skyblue") +
  labs(title = "Top 6 Job Salaries", x = "Job Title", y = "Average Salary (USD)")
bot6jobplot <- ggplot(bottom_6_job_salaries, aes(x = reorder(job_title, Avg_Sal), y = Avg_Sal)) +
  geom_bar(stat = "identity", fill = "salmon") +
  labs(title = "Bottom 6 Job Salaries", x = "Job Title", y = "Average Salary (USD)")
top6jobplot <- top6jobplot + theme(axis.text.x = element_text(angle = 90, vjust = 0.75, size=7, hjust=1)
bot6jobplot <- bot6jobplot + theme(axis.text.x = element_text(angle = 90, vjust = 0.75, size=7, hjust=1)
grid.arrange(top6jobplot,bot6jobplot, ncol = 2)
```


+ We can tell that the data science tech lead job has the highest average pay by far + we can also tell that the power bi developer has the lowest pay out of all the jobs

Tabling the 6 highest and lowest paying company locations

```
top_6_loc<-salaries%>%
  group_by(company_location)%>%
  summarise(Avg_Sal=mean(salary_in_usd))%>%
  arrange(desc(Avg_Sal))%>%
  head()
top_6_loc
bottom_6_loc<-salaries%>%
  group_by(company_location)%>%
  summarise(Avg_Sal=mean(salary_in_usd))%>%
  arrange(Avg_Sal)%>%
```

```
  head()
bottom_6_loc
```

Plotting the 6 highest and lowest paying company locations

```
top6locplot <- ggplot(top_6_loc, aes(x = reorder(company_location, Avg_Sal), y = Avg_Sal)) +
  geom_bar(stat = "identity", fill = "skyblue") +
  labs(title = "Top 6 Company Locations", x = "Location Code", y = "Average Salary (USD)")
bot6locplot <- ggplot(bottom_6_loc, aes(x = reorder(company_location, Avg_Sal), y = Avg_Sal)) +
  geom_bar(stat = "identity", fill = "salmon") +
  labs(title = "Bottom 6 Company Locations", x = "Location Code", y = "Average Salary (USD)")
top6resplot <- top6locplot + theme(axis.text.x = element_text(angle = 90, vjust = 0.75, size=7, hjust=1)
bot6resplot <- bot6locplot + theme(axis.text.x = element_text(angle = 90, vjust = 0.75, size=7, hjust=1)
grid.arrange(top6locplot,bot6locplot, ncol = 2)
```
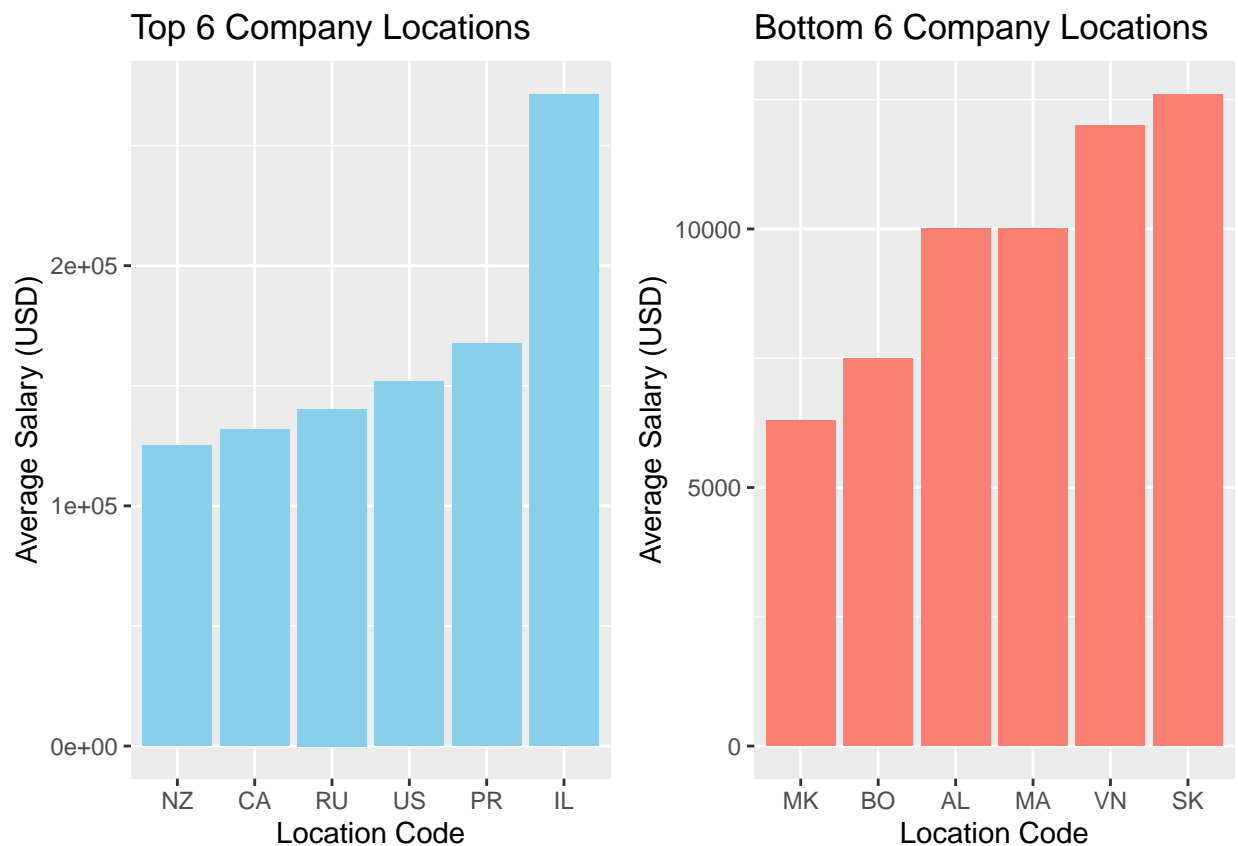

+ It seems that IL has the highest paying jobs when it comes to company location + MK has the lowest paying job out of all the locations by far

Part 2: Problem formulation and discussion of statistical learning algorithms used

The main goal of this project, as mentioned before is to compare several statistical learning methods in predicting the salary given the parameters. We will be focusing on the following methods:

- K-Nearest Neighbors

- Using K-NN means
- Scaling the predictors

- Linear Regression

  - Test with adding polynomials, etc

- Ridge and LASSO Regression

  - Using CV methods to find the best lambda
  - Conducting variable selection with LASSO

- Regression Trees

  - Decision Trees
  - Random-Forest
  - Boosting

- Support Vector Machines

  - Testing SVMs with different OSH methods

I want to start off by deciding that the salary_currency and salary are not significant to our prediction of the salary_in_usd. Similarly for employee_residence, as the company_location should be sufficient

```
salaries <- subset(salaries, select = -c(salary, salary_currency, employee_residence))
head(salaries)
```

Due to a huge amount of levels in factors, our training dataset may not have the columns present in the testing dataset and vice versa. To prevent this, I will create a new data set where I use one-hot encoding on those categorical variables

```
library(caret)

cat_cols <- c("job_title", "company_location")
formula <- as.formula(paste("~ ."))
encoded_data <- dummyVars(formula, data = salaries[, cat_cols])
encoded_salaries <- predict(encoded_data, newdata = salaries)
encoded_salaries <- cbind(salaries, encoded_salaries)
encoded_salaries <- subset(encoded_salaries, select = -c(job_title, company_location))
head(encoded_salaries)
```

Preparing the encoded training and testing data set with Train/Test/Split

```
set.seed(123)
nrow(encoded_salaries)
encoded_train = sample(1:nrow(encoded_salaries), 3003)
encoded_salaries_train = encoded_salaries[encoded_train, ]
encoded_salaries_test = encoded_salaries[-encoded_train, ]

encoded_YTrain = encoded_salaries_train$salary_in_usd
encoded_XTrain = encoded_salaries_train %>% select(-salary_in_usd)

encoded_YTest = encoded_salaries_test$salary_in_usd
encoded_XTest = encoded_salaries_test %>% select(-salary_in_usd)
```

K-NN Regression

We need to encode all of the categorical variables for this. To do so, create a new dataset just for the KNN's use

```r
cat_cols_2 <- c("job_title", "company_location", "experience_level", "employment_type", "company_size",
formula <- as.formula(paste("~ ."))
encoded_data2 <- dummyVars(formula, data = salaries[, cat_cols])
encoded_salaries2 <- predict(encoded_data2, newdata = salaries)
encoded_salaries2 <- cbind(salaries, encoded_salaries2)
encoded_salaries2 <- subset(encoded_salaries2, select = -c(job_title, company_location, experience_level

set.seed(123)
nrow(encoded_salaries2)
encoded_train2 = sample(1:nrow(encoded_salaries2), 3003)
knn_salaries_train = encoded_salaries2[encoded_train2, ]
knn_salaries_test = encoded_salaries2[-encoded_train2, ]

knn_YTrain = knn_salaries_train$salary_in_usd
knn_XTrain = knn_salaries_train %>% select(-salary_in_usd)

knn_YTest = knn_salaries_test$salary_in_usd
knn_XTest = knn_salaries_test %>% select(-salary_in_usd)

head(knn_YTrain)
```

Training the reggresor using the encoded training set, for now we will use k=10, decide afterwards if it is worth spending time on cross-validation with this method.

```r
options(max.print = 1000)
set.seed(123)
pred_YTrain = knn.reg(train=knn_XTrain, test=knn_XTrain, y=knn_YTrain, k=10)
```

Calculating the Training MSE

```r
mean((pred_YTrain$pred - knn_YTrain)^2) #3065374515
```

Calculating test MSE

```r
pred.YTest = knn.reg(train=knn_XTrain, test=knn_XTest, y=knn_YTrain, k=10)
mean((pred.YTest$pred - knn_YTest)^2) #3544608428
```

After some thought and considerations, I believe that the KNN regression method is not worth for this sort of data set. Firstly, the original data set is full of categorical variables, we were only able to fit it into the model after encoding basically everything. Due to this, our training and test MSE seems to be very high. (train mse stays, test mse lowers as k goes from 1 to 10 which is an interesting connection) Visualization is inaccurate too due to KNN regression visualizations only being able to be graphed with 1 response, and in this case the only variable I can graph on would be the work_year, which wouldn't really show anything. Due to the above reasons, I will not be spending time in tuning the lambda for said method.

Linear Regression

Using log transformation on the response variable to follow a normal

```
log.train <- encoded_salaries_train
log.train$salary_in_usd <- log1p(encoded_salaries_train$salary_in_usd)
```

Fitting the model

```
lmod <- lm(salary_in_usd ~ ., log.train)
summary(lmod)
```

Checking Predictions

```
options(max.print = 6)
predicted_values <- predict(lmod)
comparison_log_df <- data.frame(Actual = log.train$salary_in_usd, Predicted = predicted_values)
comparison_log_df
```

```
##        Actual Predicted
## 2463 12.38840  11.89292
## 2511 12.04356  11.89292
## 2227 12.10072  11.72904
##  [ reached 'max' / getOption("max.print") -- omitted 3000 rows ]
```

Creating plot of predicted values vs the actual values (displayed with the non log plot below)

```
gg_comparison_log <- ggplot(comparison_log_df, aes(x = log.train$salary_in_usd, y = predicted_values)) +
  geom_point() +
  geom_abline(slope = 1, intercept = 0, color = "red", linetype = "dashed") +
  labs(title = "Log Predicted vs Log Actual", x = "Log Actual", y = "Log Predicted")
```

checking non log'd predictions

```
options(max.print = 6)
comparison_df <- data.frame(Actual = encoded_salaries_train$salary_in_usd, Predicted = exp(predicted_val
comparison_df
```

```
##      Actual Predicted
## 2463 240000  146227.7
## 2511 170000  146227.7
## 2227 180000  124124.8
##  [ reached 'max' / getOption("max.print") -- omitted 3000 rows ]
```

Creating the non log plot

```
gg_comparison <- ggplot(comparison_df, aes(x = Actual, y = Predicted)) +
  geom_point() +
  geom_abline(slope = 1, intercept = 0, color = "red", linetype = "dashed") +
  labs(title = "Predicted vs Actual", x = "Actual", y = "Predicted")
```

Displaying the 2 plots

11

```
grid.arrange(gg_comparison_log,gg_comparison, ncol = 2)
```
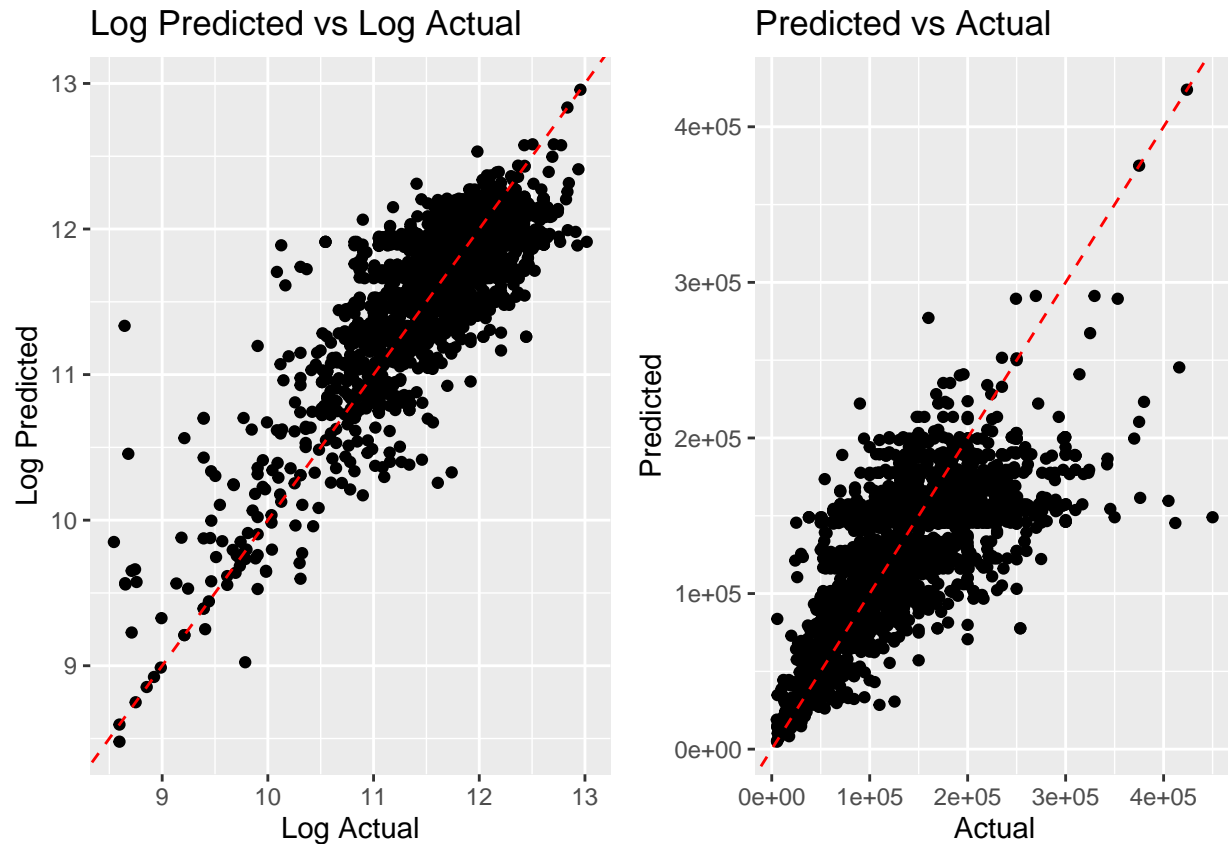

Getting R^2, Mean absolute error and Mean Squared Error for the log training response

```
rsquared_log <- summary(lmod)$r.squared
residuals_log <- residuals(lmod)
mae_log <- mean(abs(residuals_log))
mse_log <- mean(residuals_log^2)
rsquared_log #0.6678302
mae_log #0.2639758
mse_log #0.1206688
```

MSE of the non log training response

```
predicted_values_nonlog <- exp(predict(lmod))
residuals_nonlog <- encoded_salaries_train$salary_in_usd - predicted_values_nonlog
mse_nonlog <- mean(residuals_nonlog^2)
mse_nonlog #2110682376
```

Predicting the test data & its MSE using the model we built

```
lmod_test_predicted <- predict(lmod, newdata = encoded_XTest)
```

```
## Warning in predict.lm(lmod, newdata = encoded_XTest): prediction from
## rank-deficient fit; attr(*, "non-estim") has doubtful cases
```

```r
mse_test <- mean((lmod_test_predicted - encoded_YTest)^2)
mse_test #23967797489
```

Overall, although we are able to fit the response variable quite well into a linear model thanks to the logarithmic transformation, the overall prediction is not very good judging from the very high MSE we get. This is probably due to the dataset mostly only having categorical predictor variables (factors with 70+ levels), making it really bad for fitting a linear model.