Binary Logistic Regression

Data 621: Homework 3 - Group 4

Sachid Deshmukh
Michael Yampol
Vishal Arora
Ann Liu-Ferrara

Oct. 20, 2019

Contents

1. Data Exploration

Let's load training dataset and preview.

Load Data

```
##   zn indus chas   nox    rm   age    dis rad tax ptratio lstat medv target
## 1  0 19.58    0 0.605 7.929  96.2 2.0459   5 403    14.7  3.70 50.0      1
## 2  0 19.58    1 0.871 5.403 100.0 1.3216   5 403    14.7 26.82 13.4      1
## 3  0 18.10    0 0.740 6.485 100.0 1.9784  24 666    20.2 18.85 15.4      1
## 4 30  4.93    0 0.428 6.393   7.8 7.0355   6 300    16.6  5.19 23.7      0
## 5  0  2.46    0 0.488 7.155  92.2 2.7006   3 193    17.8  4.82 37.9      0
## 6  0  8.56    0 0.520 6.781  71.3 2.8561   5 384    20.9  7.67 26.5      0
```

```
## [1] "Number of columns =  13"
```

```
## [1] "Number of rows =  466"
```

As we can see in the preview, training data has 13 columns and 466 rows

Let's analyze datatypes of each column.

Analyze Datatypes

```
## 'data.frame':    466 obs. of  13 variables:
##  $ zn     : num  0 0 0 30 0 0 0 0 0 80 ...
##  $ indus  : num  19.58 19.58 18.1 4.93 2.46 ...
##  $ chas   : int  0 1 0 0 0 0 0 0 0 0 ...
##  $ nox    : num  0.605 0.871 0.74 0.428 0.488 0.52 0.693 0.693 0.515 0.392 ...
##  $ rm     : num  7.93 5.4 6.49 6.39 7.16 ...
##  $ age    : num  96.2 100 100 7.8 92.2 71.3 100 100 38.1 19.1 ...
##  $ dis    : num  2.05 1.32 1.98 7.04 2.7 ...
##  $ rad    : int  5 5 24 6 3 5 24 24 5 1 ...
##  $ tax    : int  403 403 666 300 193 384 666 666 224 315 ...
##  $ ptratio: num  14.7 14.7 20.2 16.6 17.8 20.9 20.2 20.2 20.2 16.4 ...
##  $ lstat  : num  3.7 26.82 18.85 5.19 4.82 ...
##  $ medv   : num  50 13.4 15.4 23.7 37.9 26.5 5 7 22.2 20.9 ...
##  $ target : int  1 1 1 0 0 0 1 1 0 0 ...
```

All the columns are of numeric types. This indicates that all the variables are quantitative. There are no categorical variables in the training dataset.

Let's check if any variables have missing values. Values which are NULL or NA.

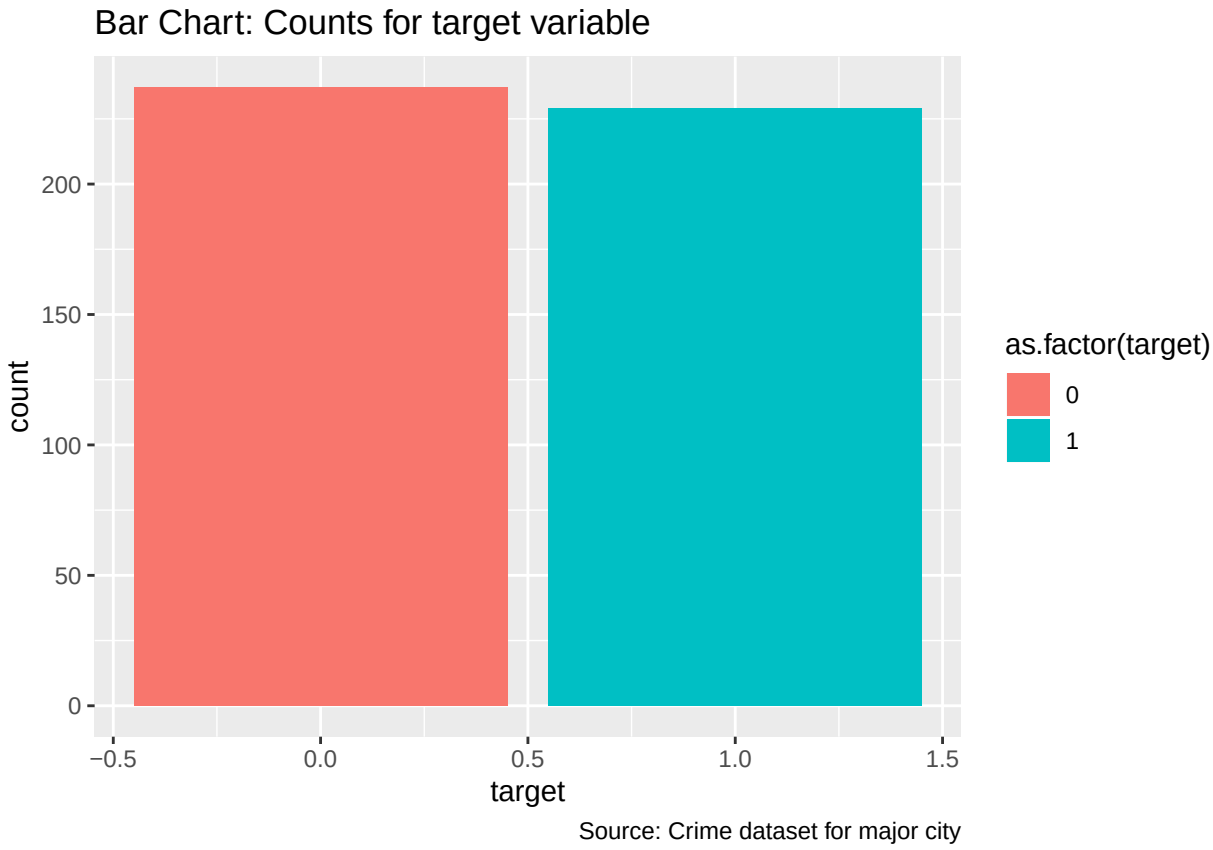
Check missing values

```
## [1] "Number of columns with missing values =  0"
```

Training dataset have no missing values. This is good because we don't have to impute values for any columns

Let's check if the training data is class imbalance. A Dataset is called class imbalance when there are very few obesrvations corresponding to minority class. This is very important in deciding model evaluation metrics.

Check class Imbalance

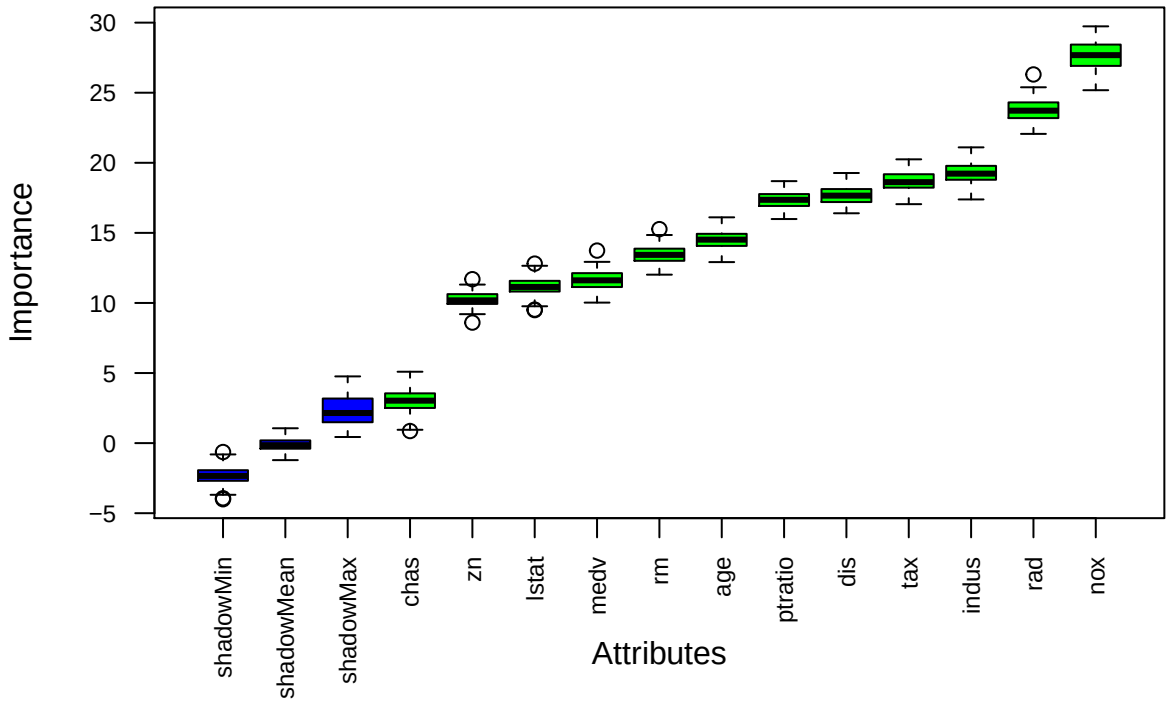


Above barchart shows that we have balanced dataset containing enough observations pertaining to both values of target variable. We can see from the above bar chart that we have 237 observations pertaining to low crime rate and 229 observations pertaining to high crime rate. This makes the dataset class balance and we don't need to worry about minority class here.

The original variable importance is ranked using Boruta function.

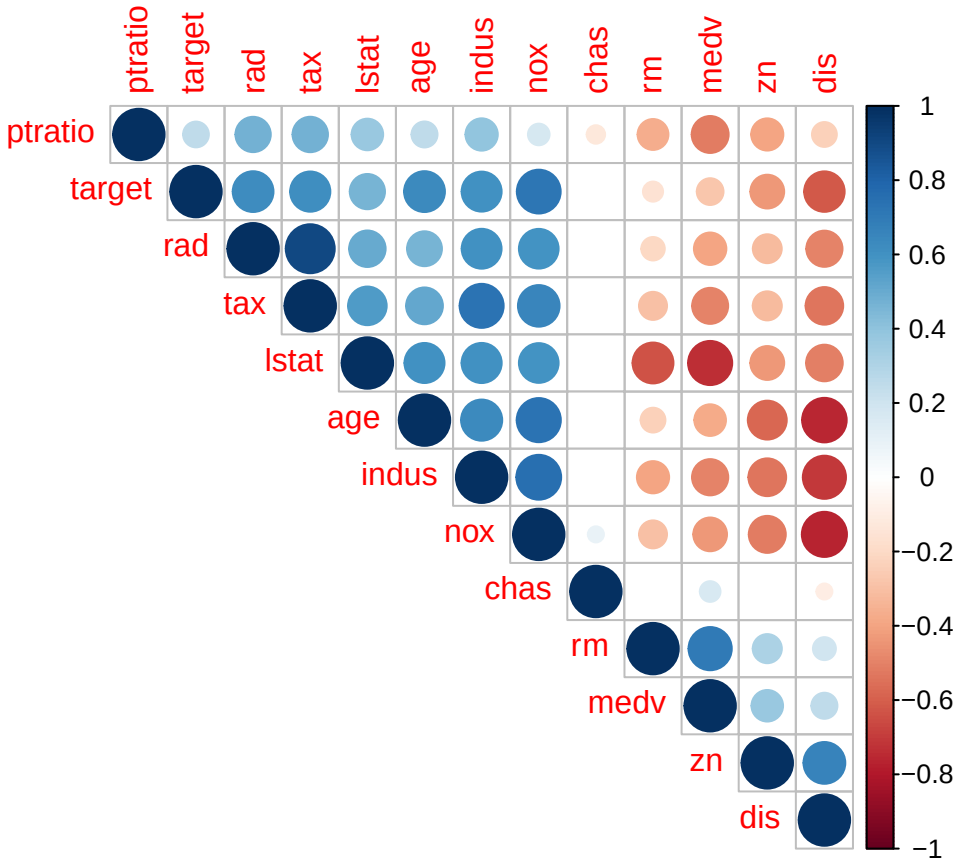
Boruta is one of the feature selection algorithms, uses Random Forest to selection importance variables. It classifies the feature variables into 3 levels based on the p value specified: Confirmed, Rejected or Tentative.

```
## Boruta performed 69 iterations in 14.86549 secs.
##  12 attributes confirmed important: age, chas, dis, indus, lstat
## and 7 more;
##  No attributes deemed unimportant.
```



From the above boruta feature selection chart, all feature variables are confirmed important variables. but chas is clearly the last one, and very close to shadow (random) variables, so its contribution to the model is questionable.

The variables which are highly correlated carry similar information and can affect model accuracy. Highly correlated variables also impacts estimation of model coefficients. Let's figure out which variables in the training datasets are highly correlated to each other



From the above correlation graph we can see that target variable is highly correlated with following variables

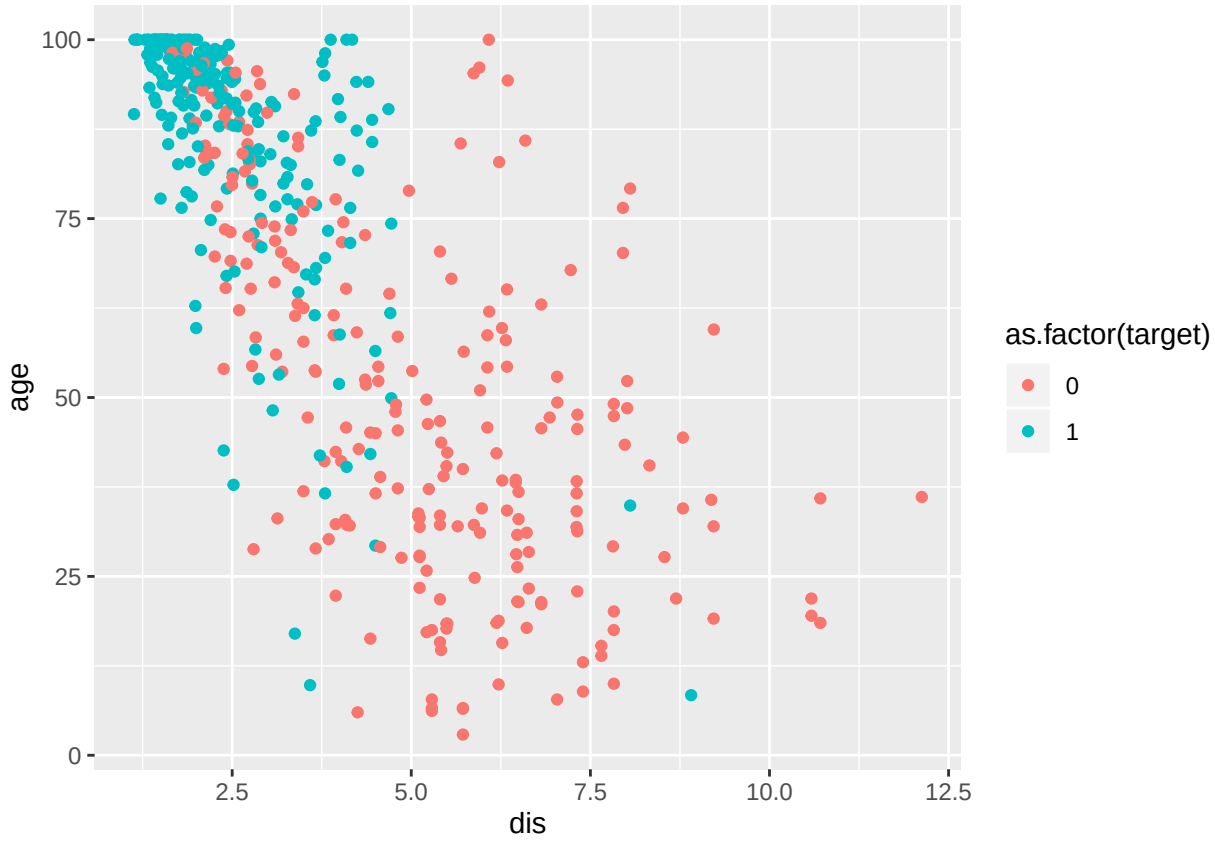
- rad: Index of accessibility to radial highways
- tax: Full-value property-tax rate per $10,000
- lstat: Lower status of the population (percent)
- age: Proportion of owner-occupied units built prior to 1940
- indus: Proportion of non-retail business acres per suburb
- nox: Nitrogen oxides concentration
- zn: Proportion of residential land zoned for large lots
- dis: Weighted mean of distances to five Boston employment centers

Let's focus on four variables listed below. These variables are not only corrlelated with outcome variable target but they also have high negative correlation with variable dis: Weighted mean of distances to five Boston employment centers

- age: Proportion of owner-occupied units built prior to 1940
- indus: Proportion of non-retail business acres per suburb
- nox: Nitrogen oxides concentration
- dis: Weighted mean of distances to five Boston employment centers

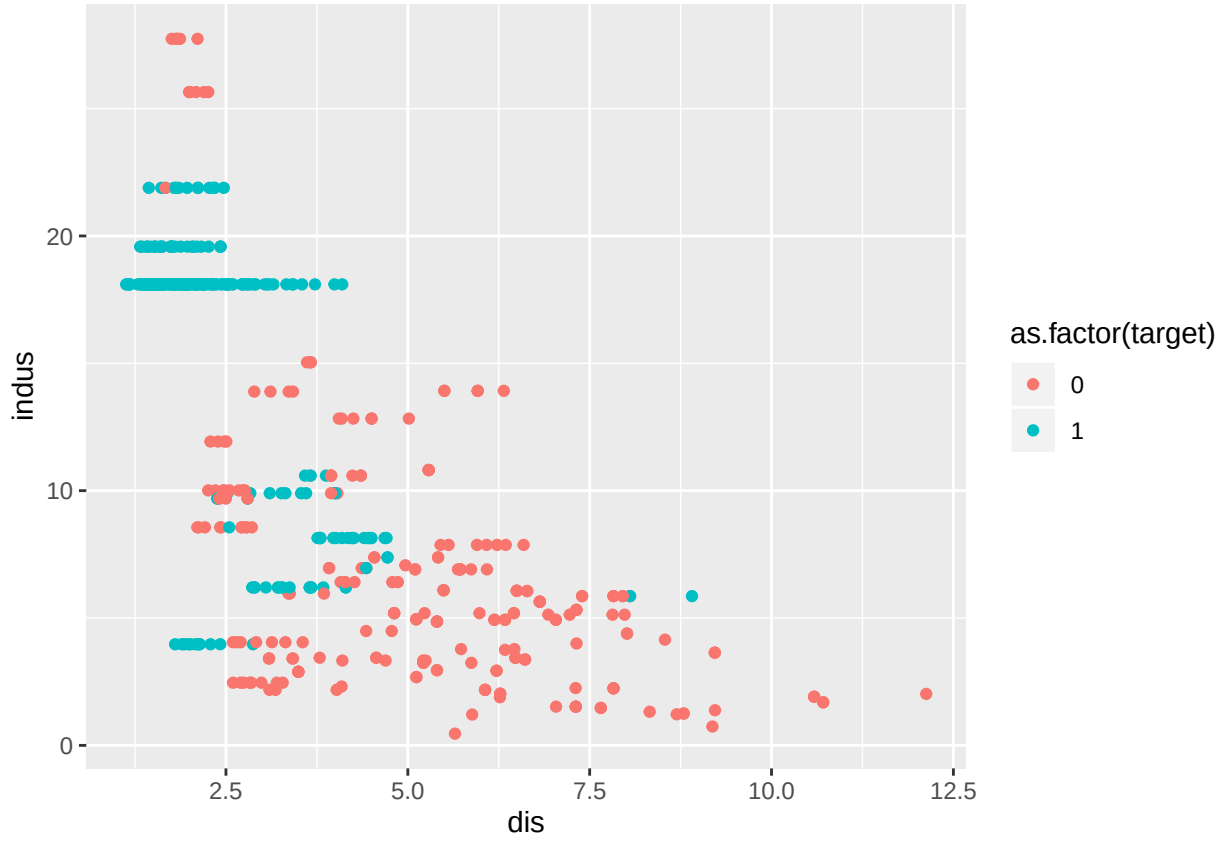
Let's analyze these four variables in more detail so that we can decide if we can add interaction term for these variables

Scatter plot of age and dis



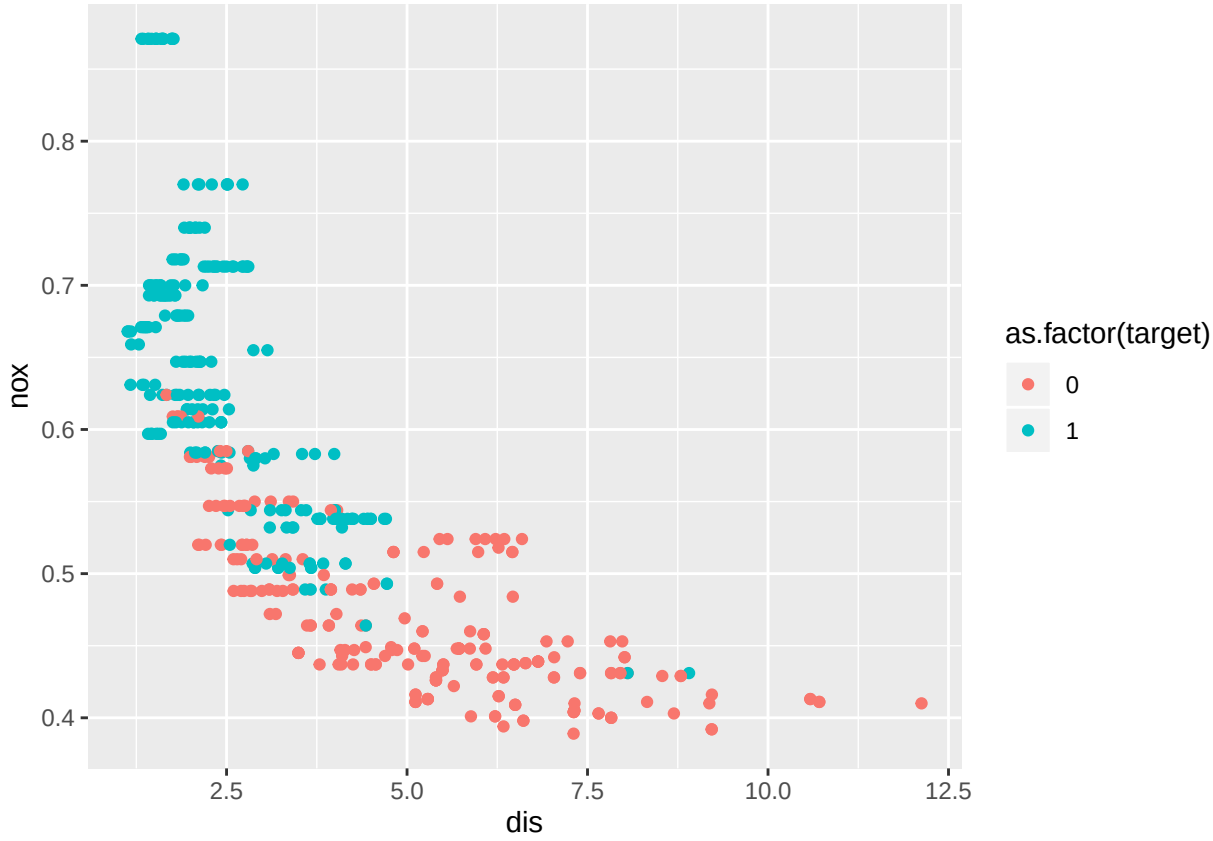
From the above scatter plot, we can conclude that lots of properties which are constructed before 1940 are near to Boston employment centers. Above Scatter plot also shows us that there is a high crime rate near the employment centers where we have lots of older properties (properties built before 1940)

Scatter plot of indus and dis



Above scatter plot shows that proportion of non retail business decreases as we move away from the Boston employment center. There is a decrease in the crime rate also as we move away from the employment center

Scatter plot of nox and dis



Above scatter plot shows that there is very high percentage of Nitrogen Oxide near Bostom employment centers. This can be due to area surrounding employment centers have more industries and that leads to more percentage of Nitorgen Oxide

2. Data Preparation

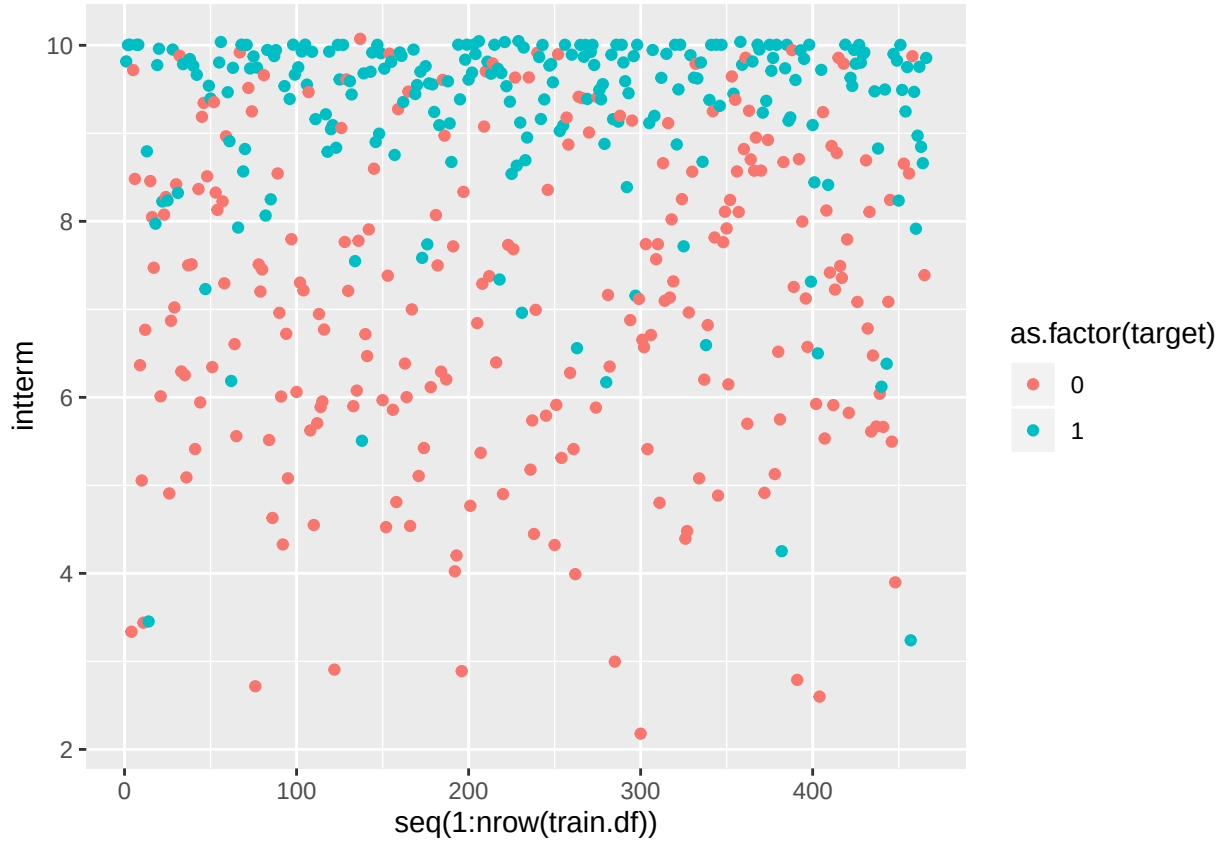
Add Interaction Term

From the above scatter plots we can conclude that old properties exists near to Boston employment centers whihc has high percentage of Nitrogen Oxide and high percentage of non retail business acres. All these four terms are correlated and that can also be seen from the correlation graph above. Let's add an interaction term to our dataset which will account for cumulative effect of all these four input variables

```
train.df$intterm =  ((train.df$dis / (train.df$nox * train.df$indus)) + train.df$age) ^ 0.5
```

Let's see how newly added interaction term is helping us to categories target variable

Scatter plot showing effect of newly added interaction term for categorizing data



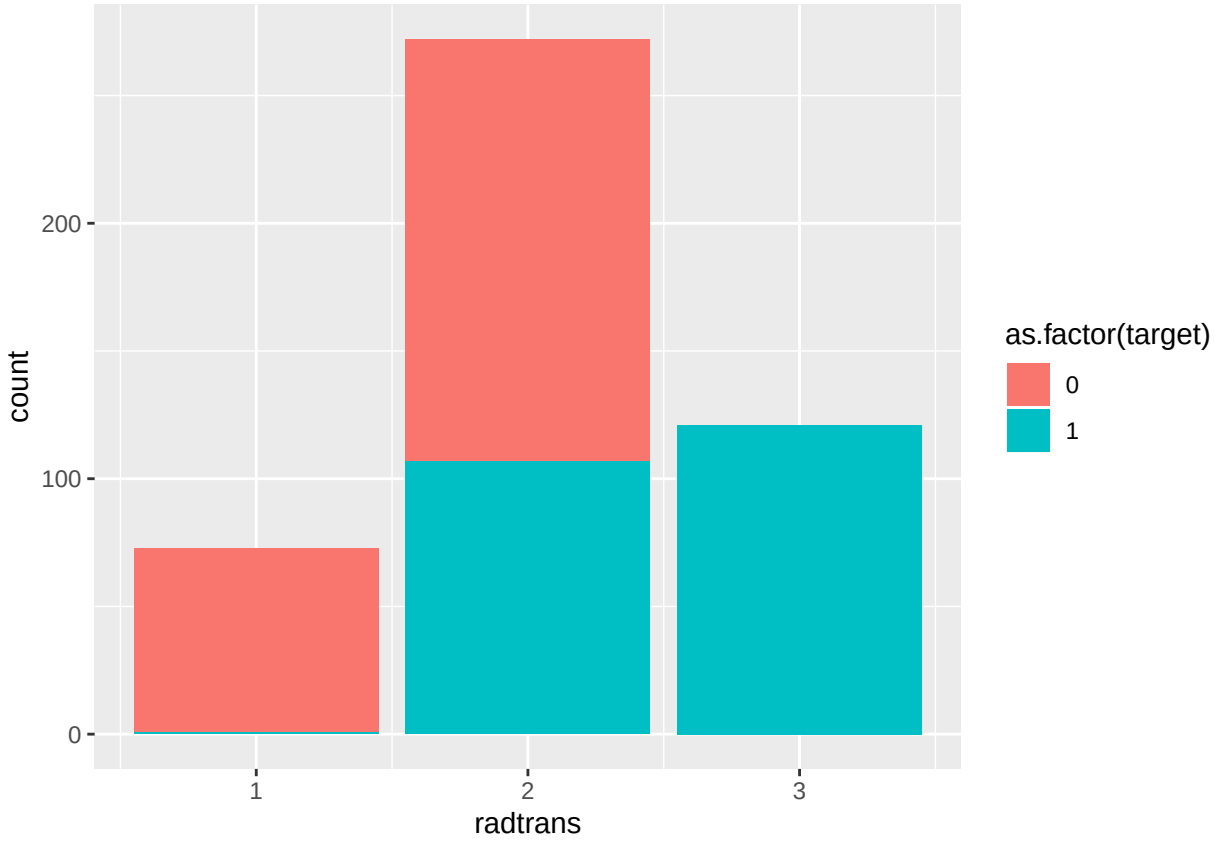
From the above scatter plot we can see that newly added interaction term is helping us to separate data points pertaining to dfferent target variable values nicely.

Add new variable by transforming rad variable into buckets

We can bucketize rad variable. Let's devide data into three buckets of rad variable

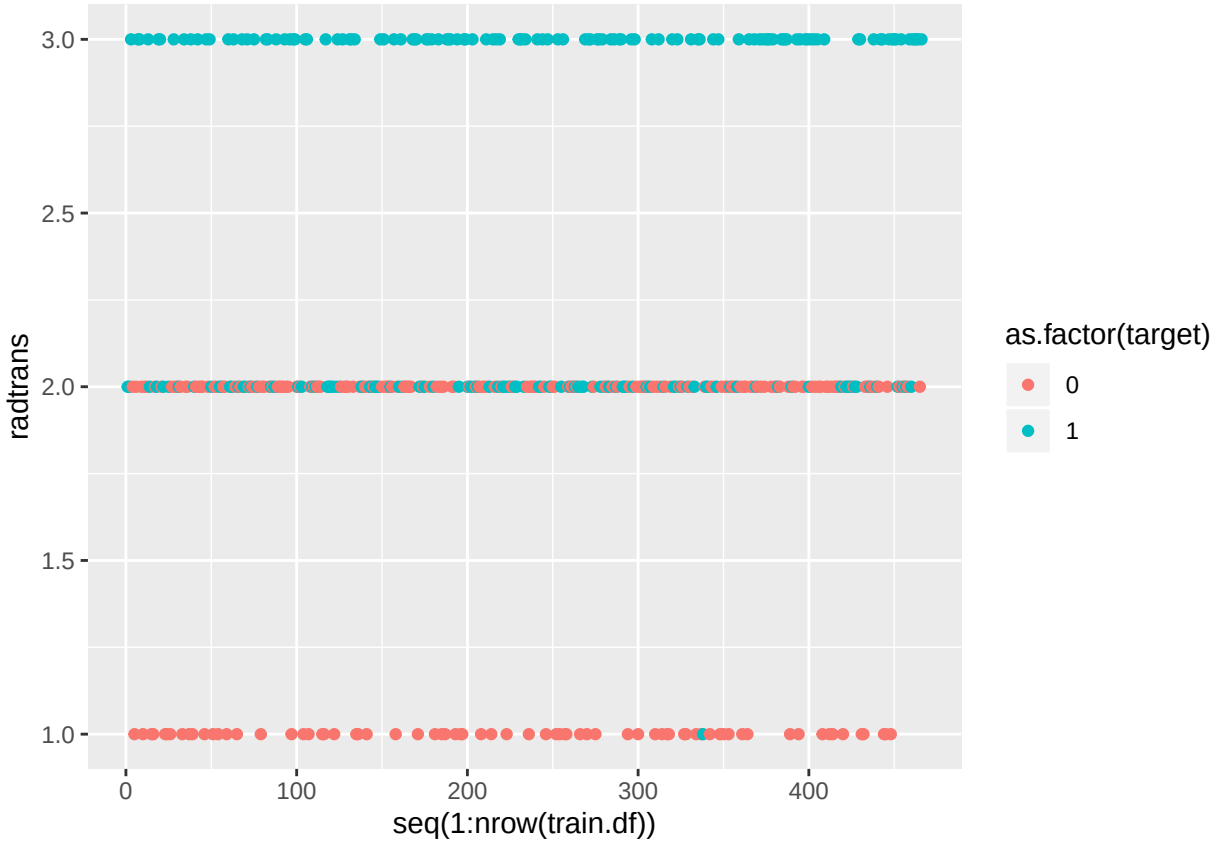
- **Bucket-1: Indicates low accessibility to highways**
- **Bucket-2: Indicates medium accessibility to highways**
- **Bucket-3: Indicates high accessibility to highways**

Let's see how crime rate is devided into these three buckets after we added newly transformed rad variable



From the above bar plot we can see that crime rate is low if there is low accessibility to highways. For medium accessibility crime rate is higher and for areas with highest level of accessibility to highways exhibit highest crime rate.

Let's see how transformed rad variable is helping us to categorize data



Add new variable by transforming tax rate variable into buckets

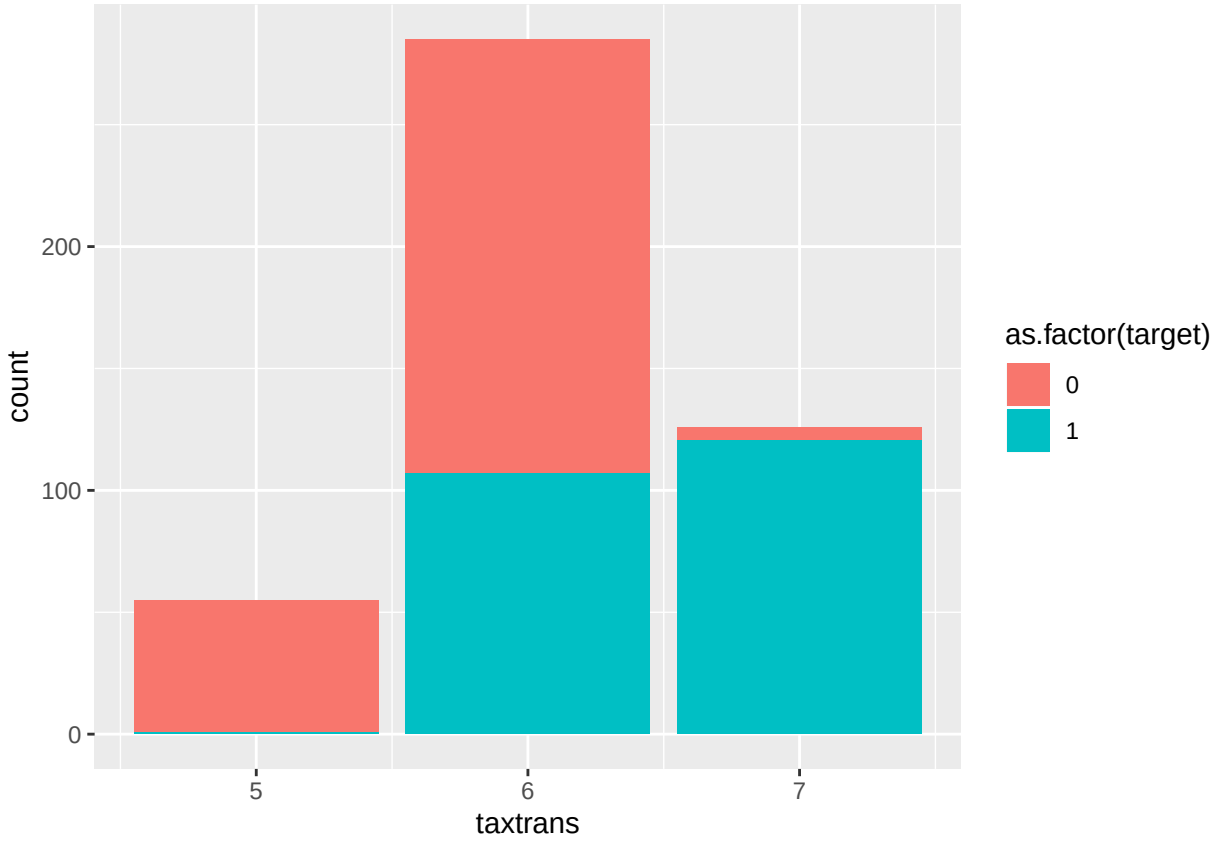
We can bucketize tax rate variable. Let's devide data into three buckets of tax rate variable

- **Bucket-1: Indicates low tax rate**
- **Bucket-2: Indicates medium tax rate**
- **Bucket-3: Indicates high tax rate**

Let's see how crime rate is devided into these three buckets after we transform tax rate variable

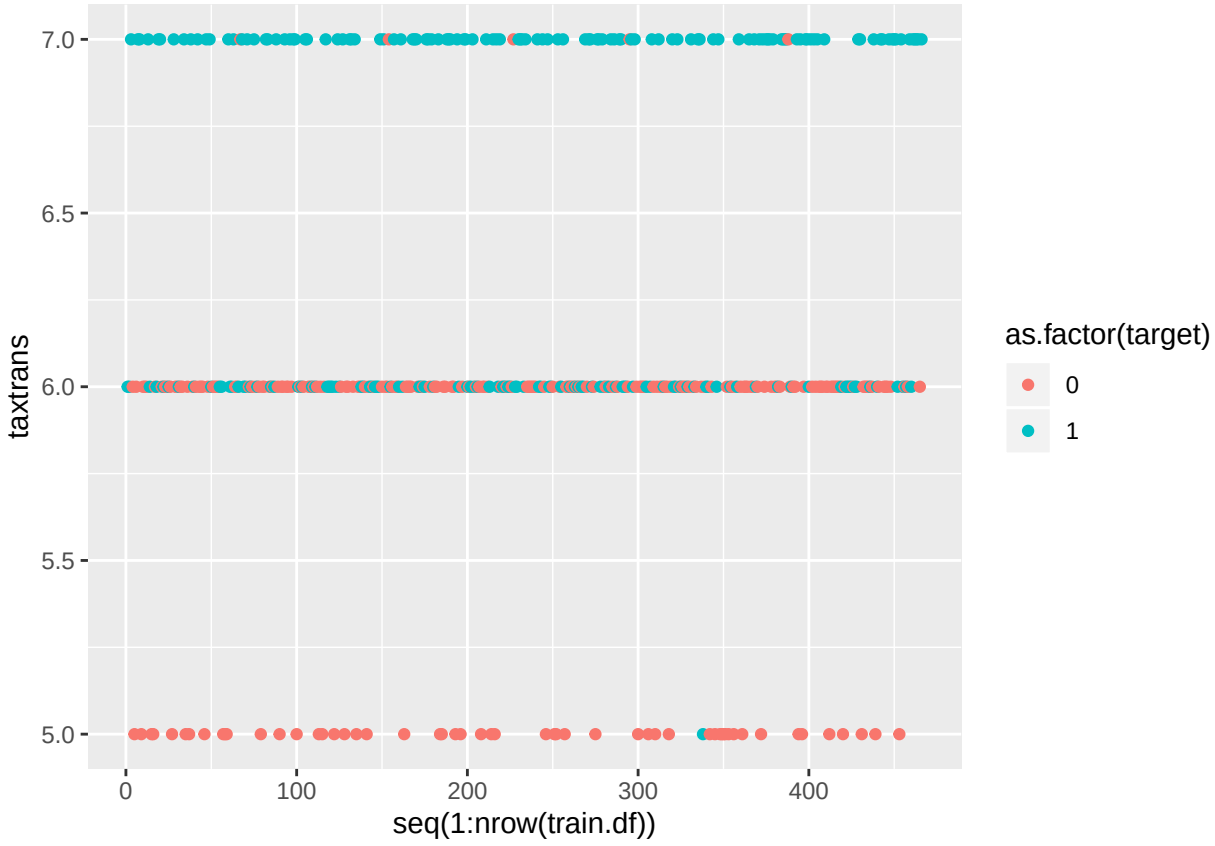
```
train.df$taxtrans = round(log(1+train.df$tax))

ggplot(train.df, aes(x=taxtrans, fill=as.factor(target)))+
geom_bar()
```



From the above bar plot we can see that crime rate is low in the area of low tax rate. For medium tax rate area crime rate is higher and for areas with highest tax rate exhibit highest crime rate. This make sense since area with higher tax rate have high value properties. Crime rate tend to be higher in the area where property values are higher.

Let's see how transformed tax variable is helping us to categorize data



Add new variable by transforming lstat variable

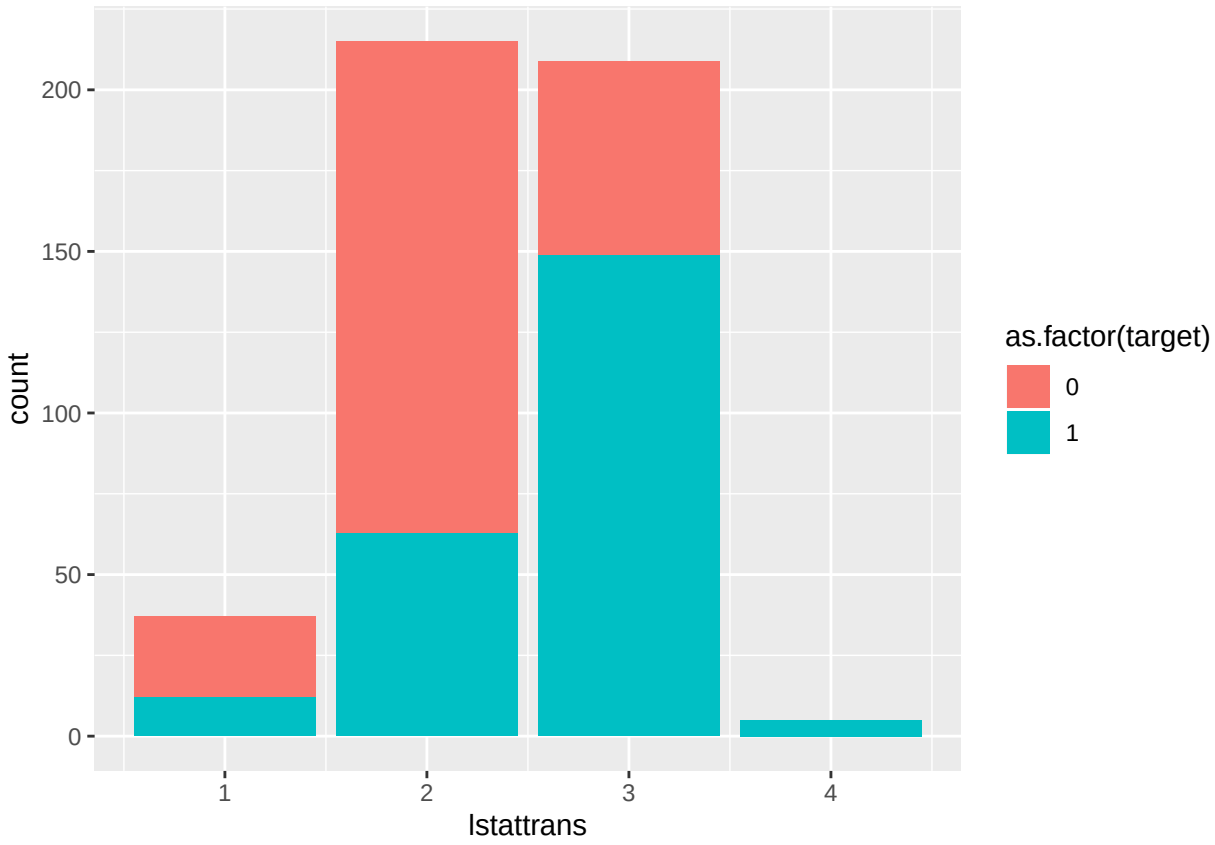
We can bucketize lower status percent variable. Let's devide data into four buckets of lower status percent variable

- **Bucket-1: Indicates very low lower status percent**
- **Bucket-2: Indicates low lower status percent**
- **Bucket-3: Indicates medium lower status percent**
- **Bucket-4: Indicates high lower status percent**

Let's see how crime rate is devided into these three buckets after we transform tax rate variable

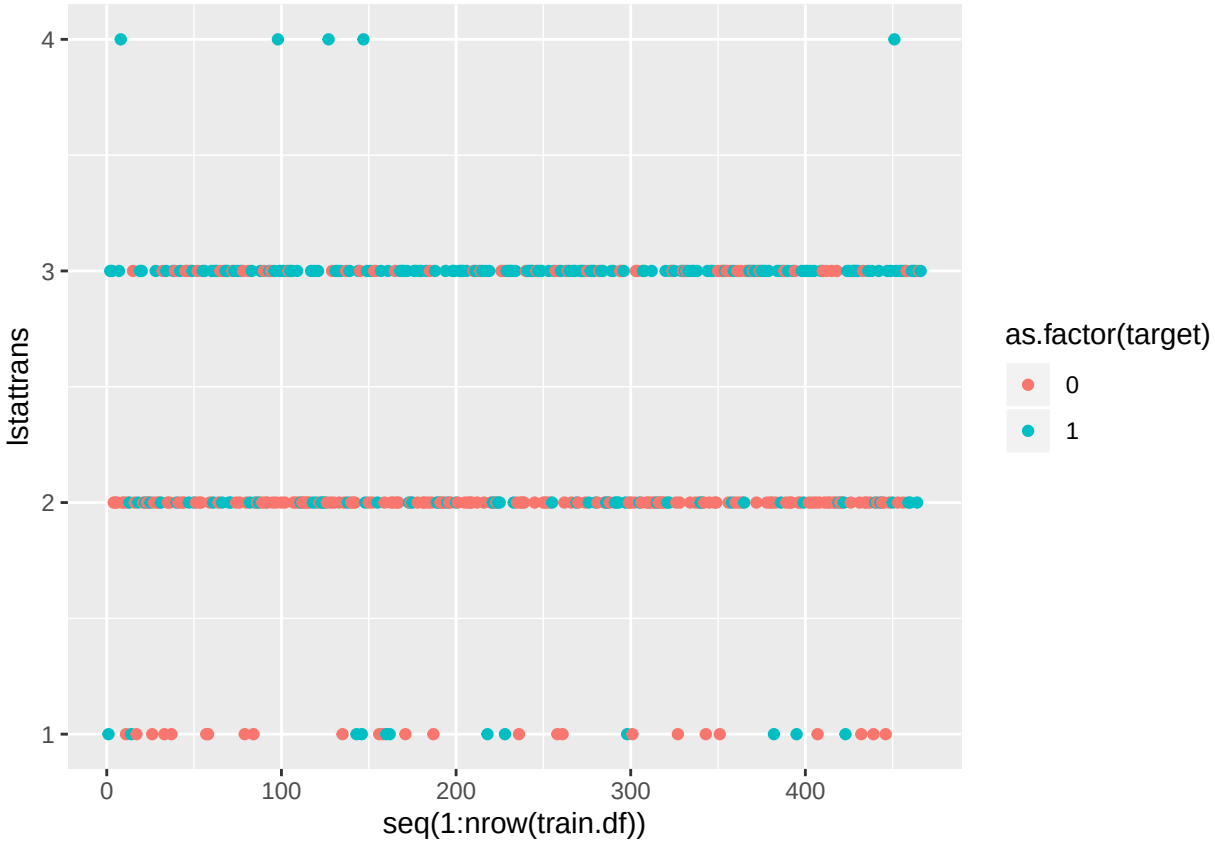
```
train.df$lstattrans = round(log(train.df$lstat))

ggplot(train.df, aes(x=lstattrans, fill=as.factor(target)))+
geom_bar()
```



From the above bar plot we can see that crime rate increases as lower status percent increases. From this we can conclude that areas with high percentage of lower status population exhibits higher crime rate

Let's see how transformed lstat variable is helping us to categorize data



Add new variable by transforming zn variable

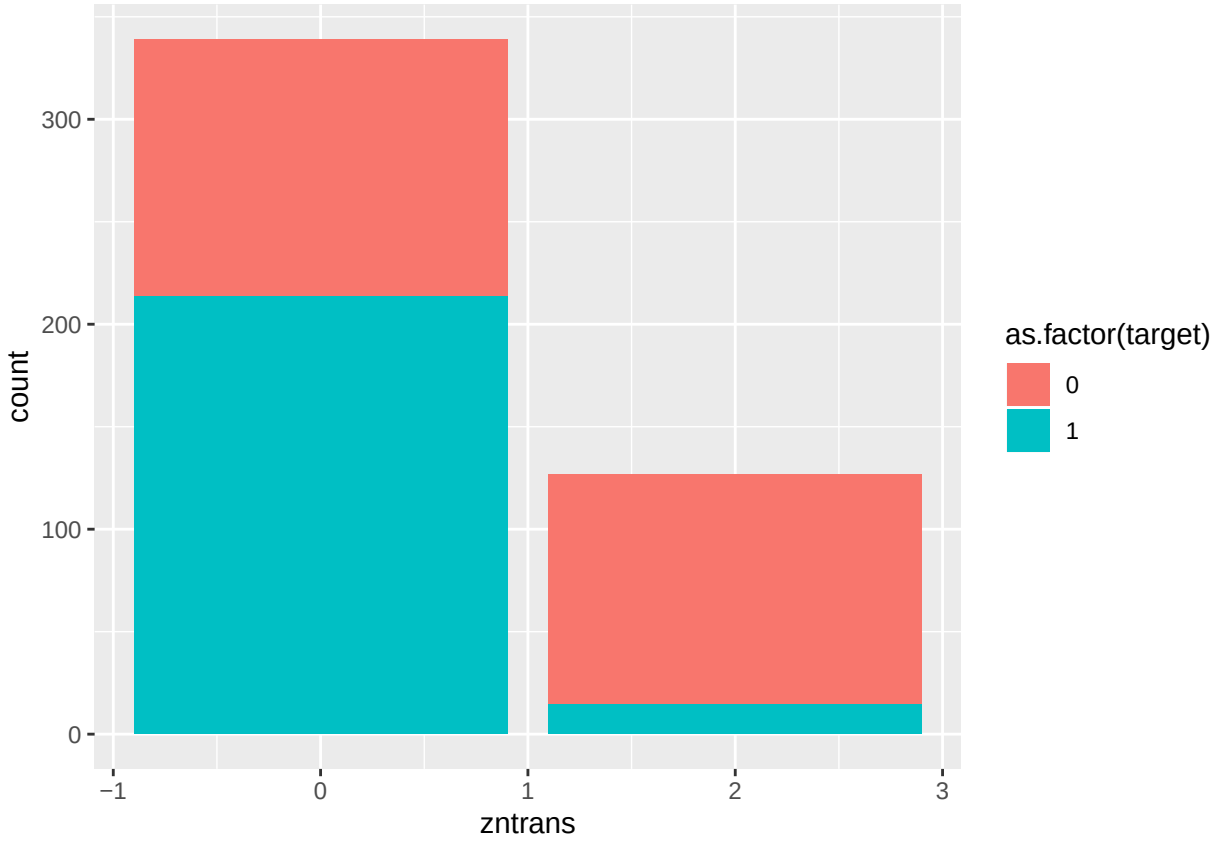
We can bucketize ZN variable. Let's devide data into TWO buckets of ZN variable

- **Bucket-1: Indicates low proportion of residential land zoned for large lots**
- **Bucket-2: Indicates high proportion of residential land zoned for large lots**

Let's see how crime rate is devided into these two buckets after we transform ZN variable

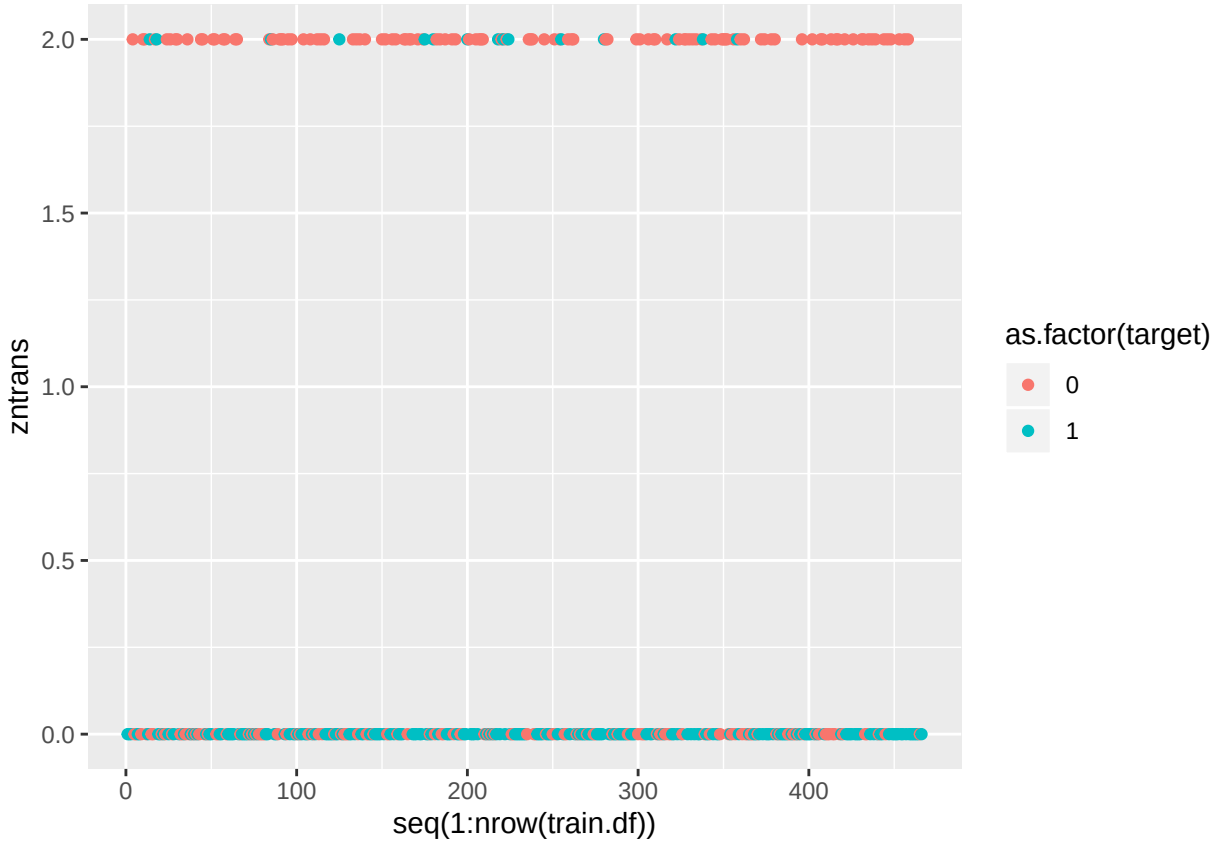
```
train.df$zntrans = round(log(1+train.df$zn^0.5))

ggplot(train.df, aes(x=zntrans, fill=as.factor(target)))+
geom_bar()
```



From the above bar chart we can see that in an area where low proportion of residential land is zoned for larger lots exhibit higher crime rate. Areas where high proportion of residential land is zoned for larger lots exhibit low crime rate

Let's see how transformed zn variable is helping us to categorize data



3. Build Models

Model fitting and evaluation

We will use Cross Validation to fit the model and evaluate it. will leverage 5 fold cross validation technique

Cross Validation

It's a model validation techniques for assessing how the results of a statistical analysis (model) will generalize to an independent data set. It is mainly used in settings where the goal is prediction, and one wants to estimate how accurately a predictive model will perform in practice. The goal of cross-validation is to define a data set to test the model in the training phase (i.e. validation data set) in order to limit problems like overfitting,underfitting and get an insight on how the model will generalize to an independent data set.

Model Evaluation Metrics

We will use following metrics for model evaluation and comparison

- **ROC AUC** : AUC - ROC curve is a performance measurement for classification problem at various thresholds settings. ROC is a probability curve and AUC represents degree or measure of separability. It tells how much model is capable of distinguishing between classes. Higher the AUC, better the model is

- **Model Accuracy** : Accuracy is one metric for evaluating classification models. Informally, accuracy is the fraction of predictions our model got right. Formally, accuracy has the following definition: Accuracy = (TP + TN)/(TP + FP + TN + FN)
- **Model Recall** : Recall is a metrics that focuses on how many true positives are identified from total positive observations in the data set. Formally Recall has following definition: Recall = TP/TP + FN
- **Model Precision** : Model precision is a metric that focuses on how many observations are truly positive out of totally identified positives. Formally Precision has following definition : Precision = TP/TP + FP

Where

- **TP** : Stands for True Positives
- **TN** : Stands for True Negatives
- **FP** : Stands for False Positives
- **FN** : Stands for False Negatives

Now we are clear on our model fitting and evaluation method (Cross Validation) and also have model evaluation metrics which we will use to compare the model effectiveness we are all set to build different models and access it's performance

We will build five different models and compare them using above mentioned model metrics

1. Base Model

Let's build a base model with following eight predictor

- **rad: Index of accessibility to radial highways**
- **tax: Full-value property-tax rate per $10,000**
- **lstat: Lower status of the population (percent)**
- **age: Proportion of owner-occupied units built prior to 1940**
- **indus: Proportion of non-retail business acres per suburb**
- **nox: Nitrogen oxides concentration**
- **zn: Proportion of residential land zoned for large lots**

```
model.metrix = cross.validation(5, "target~ rad+tax+lstat+age+indus+nox+zn+dis", train.df)
print.model.matrix("Base Model", model.metrix)
```

```
## [1] "Printing Metrix for model:  Base Model"
## [1] "AUC : 0.953606020208308"
## [1] "Accuracy : 0.87741935483871"
## [1] "Recall : 0.85940805394676"
## [1] "Precision : 0.884431012250161"
```

Note that above metrics are from a baseline model. These metrics can act like a baseline to compare model effectiveness as we add more variables to the model.

2. Model with Interaction Term

Let's add newly created interaction term to the model and see if it improves performance

```
model.metrix = cross.validation(5, "target~ rad+tax+lstat+age+indus+nox+zn+dis + intterm", train.df)
print.model.matrix("Model with interaction term", model.metrix)
```

```
## [1] "Printing Metrix for model:  Model with interaction term"
## [1] "AUC : 0.957999559963697"
## [1] "Accuracy : 0.898924731182796"
## [1] "Recall : 0.898792043940506"
## [1] "Precision : 0.893309572855849"
```

Above metrics are better than baseline model. This proves that the interaction term we have created is hellping to improve model effectiveness towards predicting crime rate.

3. Model with transformed variables

In addition to interaction term we will also include newly created transformed variables to the model

```
model.metrix = cross.validation(5, "target~ rad+tax+lstat+age+indus+nox+zn+dis + intterm + radtrans + ta
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
print.model.matrix("Model with transformed variable term", model.metrix)
```

```
## [1] "Printing Metrix for model:  Model with transformed variable term"
## [1] "AUC : 0.974811756641251"
## [1] "Accuracy : 0.926881720430108"
## [1] "Recall : 0.937286211490877"
## [1] "Precision : 0.914603635922785"
```

Above metrics are even better and beats the performance of model with interactive term. This proves that the interaction term along with transformed variables we have created is helping to improve model effectiveness towards predicting crime rate.

4. Model with original all variables

Use all original 12 variables in the dataset to fit the model without adding interation and transformed variables.

```
model.metrix = cross.validation(5, "target~ rad+tax+lstat+age+indus+nox+zn+dis + medv+rm+ptratio +chas"
print.model.matrix("Model with all original variables", model.metrix)
```

```
## [1] "Printing Metrix for model:  Model with all original variables"
## [1] "AUC : 0.961738716946576"
## [1] "Accuracy : 0.896774193548387"
## [1] "Recall : 0.880492903173179"
## [1] "Precision : 0.906328407368241"
```

Above metrics are better than 8 variable base model, it is expected as more information was brought in the model by additional important variables, the accuracy is close to the 2nd model above with interaction term.

5. Model with all original variables except chas

Use all original variables in the dataset except chas variable, as boruta confirms importance to all the variables in the train.df dataset but chas, it was tentative, not as important as other

variables.

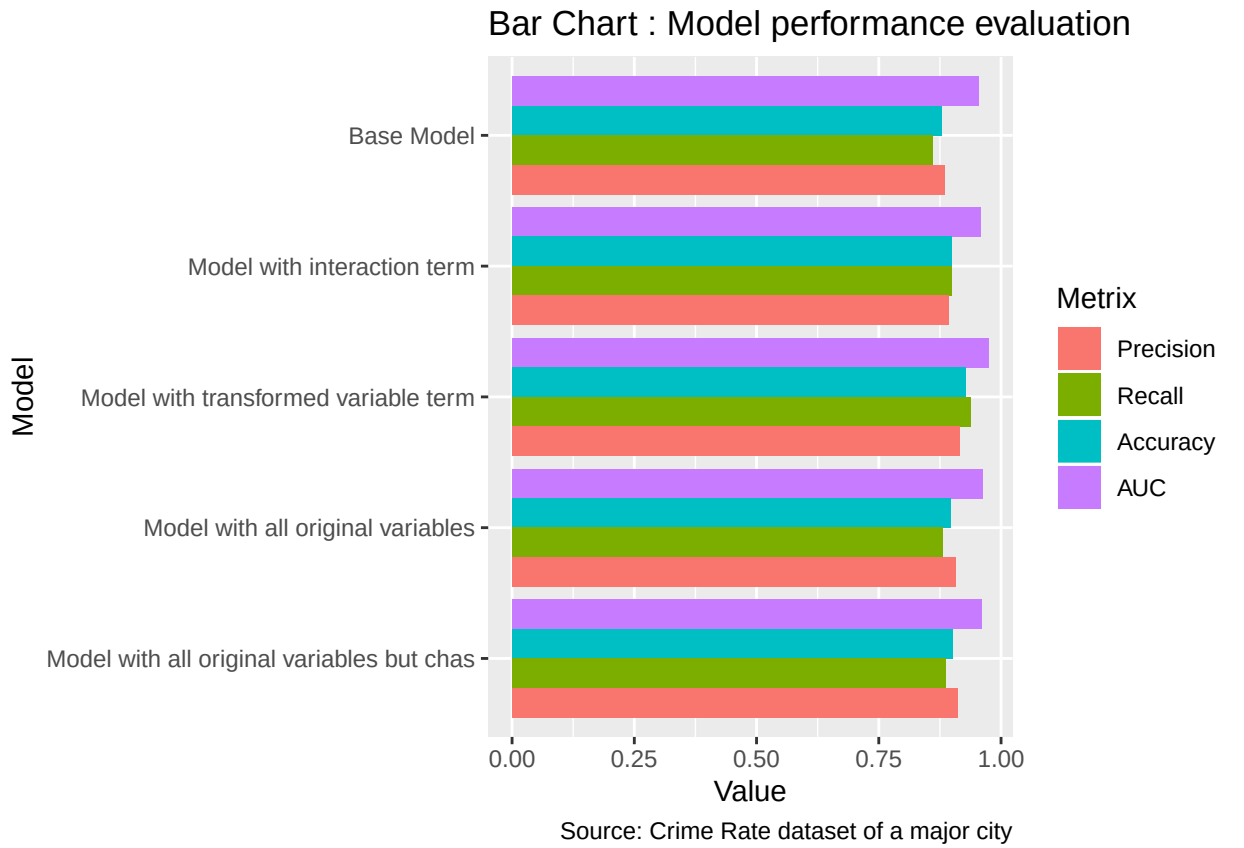
```
model.metrix = cross.validation(5, "target~ rad+tax+lstat+age+indus+nox+zn+dis + medv+rm+ptratio", train
print.model.matrix("Model with all original variables but chas", model.metrix)
```

```
## [1] "Printing Metrix for model:  Model with all original variables but chas"
## [1] "AUC : 0.959861242646322"
## [1] "Accuracy : 0.901075268817204"
## [1] "Recall : 0.885691464774179"
## [1] "Precision : 0.911204960453081"
```

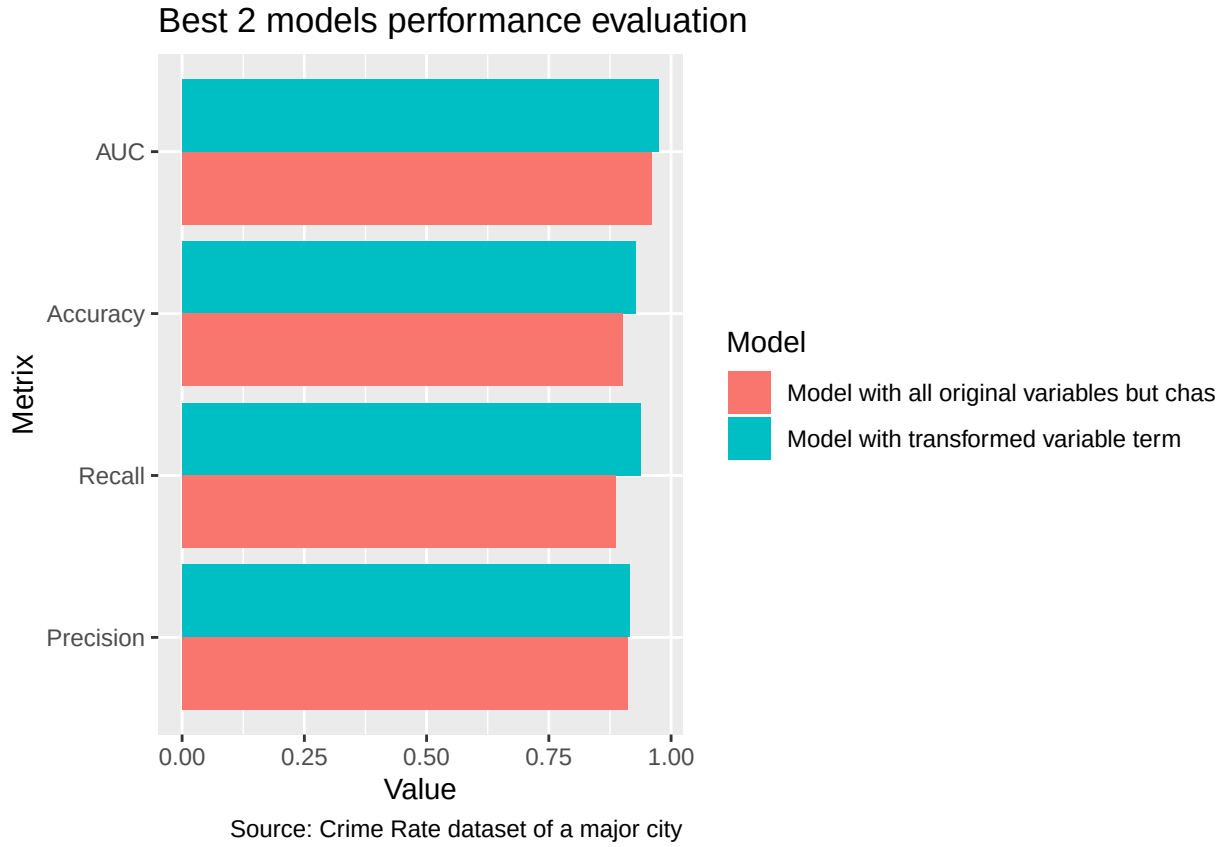
Above metrics show that this model performans better than the 4th model that uses all variables in the model, it proves that removing the less important variable chas improves the model performance; the accuracy and precision both beat the first two models above, but slightly below the model 3, which uses 8 origianl variables plus 3 interaction and transformed variables.

4. Select Models

From the above analysis it is proven that model-3 which has base predictor along with interaction term and transformed variables is the best performing model. Below bar chart shows different models and helps us to compare them on the basis of model metrics



Discussion. **The best two models are compared in the bar chart below. Both models have 11 variables. The interaction term and transformed variables brought higher accuracy, AUC, and Recall to the model, but the precision values are very close. The discussion point here is that the interaction and transformed variables increase the complexity of the model, we should work with domain experts to see if the increased complexity on the model outweights the increased performance; and if the model is easy to explain and implement comparing to the simple model without transformation and interaction term.**



5. Appendix

Fit model and predict on evaluation dataset

```
training = train.df
testing = read.csv("./Data/crime-evaluation-data_modified.csv", stringsAsFactors = FALSE)
testing$intterm =  ((testing$dis / (testing$nox * testing$indus)) + testing$age) ^ 0.5

testing$radtrans = round(log(1+testing$rad))
testing$taxtrans = round(log(1+testing$tax))

model.formula = "target~ rad+tax+lstat+age+indus+nox+zn+dis + intterm + radtrans + taxtrans"


model <- glm(formula = model.formula,
```

```
            family = "binomial", data = training)
predicted <- predict(model, newdata = testing,type="response")
lables = ifelse(predicted > 0.5, 1, 0)

testing$targetproba = round(predicted,1)
testing$target = lables
testing = data.frame(testing)
write.table(testing, "./PredictedOutcome.csv", row.names = FALSE, sep=",")
```