Binary Logistic Regression

Data 621: Homework 3 - Group 4

Sachid Deshmukh
Michael Yampol
Vishal Arora
Ann Liu-Ferrara

Oct. 30, 2019

Contents

---

1. Data Exploration

Data Description

Predictor variables:

The dataset consists of information concerning crime in various neighborhoods of a major city (Boston.)
The available predictor variables include:

Num	Variable	Description
1.	`zn`	proportion of residential land zoned for large lots (over 25000 square feet)
2.	`indus`	proportion of non-retail business acres per suburb
3.	`chas`	a dummy var. for whether the suburb borders the Charles River (1) or not (0)
4.	`nox`	nitrogen oxides concentration (parts per 10 million)
5.	`rm`	average number of rooms per dwelling
6.	`age`	proportion of owner-occupied units built prior to 1940
7.	`dis`	weighted mean of distances to five Boston employment centers
8.	`rad`	index of accessibility to radial highways
9.	`tax`	full-value property-tax rate per $10,000
10.	`ptratio`	pupil-teacher ratio by town
11.	`lstat`	lower status of the population (percent)
12.	`medv`	median value of owner-occupied homes in $1000s

The project description indicated that one more predictor variable should have been included:

13. `black` : $1000 \cdot (Bk - 0.63)^2$, where Bk is the proportion of blacks by town

However, this feature was not found in the dataset, so it had apparently been removed before the dataset was made available to us.

Target Variable:

The `target` variable is binary, set to 0 or 1, indicating whether the crime rate in the neighborhood is above the median crime rate (1 = High Crime Rate) or below (0 = Low Crime Rate).

Load Data

Let's load the training dataset and preview it:

Here are the first 10 observations from the training dataset:

	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
7	0	18.10	0	0.693	5.453	100.0	1.4896	24	666	20.2	30.59	5.0	1
8	0	18.10	0	0.693	4.519	100.0	1.6582	24	666	20.2	36.98	7.0	1
9	0	5.19	0	0.515	6.316	38.1	6.4584	5	224	20.2	5.68	22.2	0
10	80	3.64	0	0.392	5.876	19.1	9.2203	1	315	16.4	9.25	20.9	0

```
## [1] "Number of columns (features) =  13"
```

```
## [1] "Number of rows (observations) =  466"
```

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

Analyze Datatypes

Let's analyze the datatypes of each column:

```
## '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, which suggests that all the variables are quantitative.

However, we note that the target variable is categorical, as it is set to "1" to represent a high-crime area, and "0" to represent a low-crime area.

Let's make this into a factor (which we will keep outside of the dataframe) to facilitate plot legends:

```
##  Factor w/ 2 levels "LowCrime","HighCrime": 2 2 2 1 1 1 2 2 1 1 ...
```

```
##  LowCrime HighCrime
##       237       229
```

Check missing values

Let's check whether any variables have missing values, e.g., values which are NULL or NA:

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

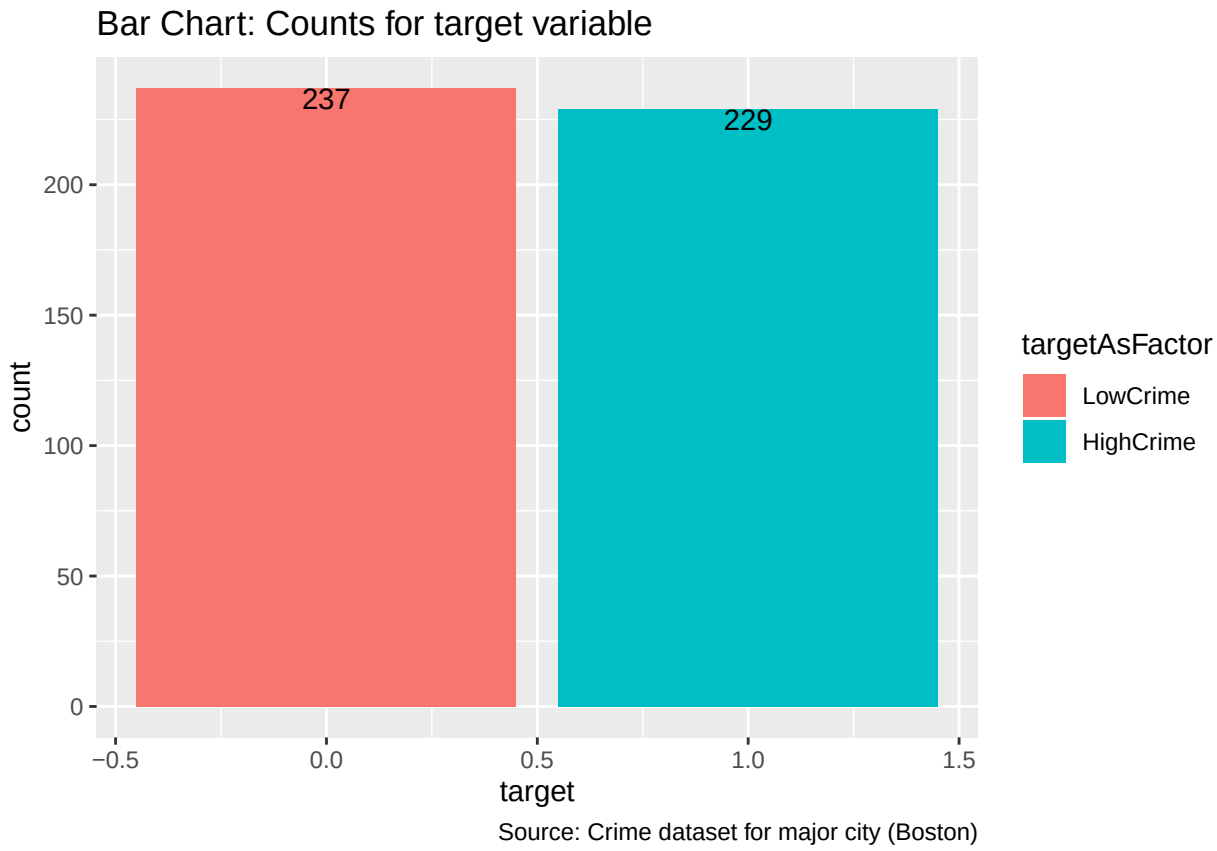
The training dataset has no missing values. This is good because we don't have to impute values for any columns.

Check Class Imbalance

Let's check whether the training data suffers from *class imbalance*.

A Dataset is called *class-imbalanced* when there are very few observations corresponding to a minority class.

This is very important in selecting *model evaluation metrics*.



The above barchart shows that we have a *balanced* dataset containing sufficient observations pertaining to both values of the `target` variable.

We can see from the above barchart that we have

237 observations pertaining to 0: `low crime rate` and

229 observations pertaining to 1: `high crime rate`.

This makes the dataset class-balanced, so we don't need to worry about the minority class here.

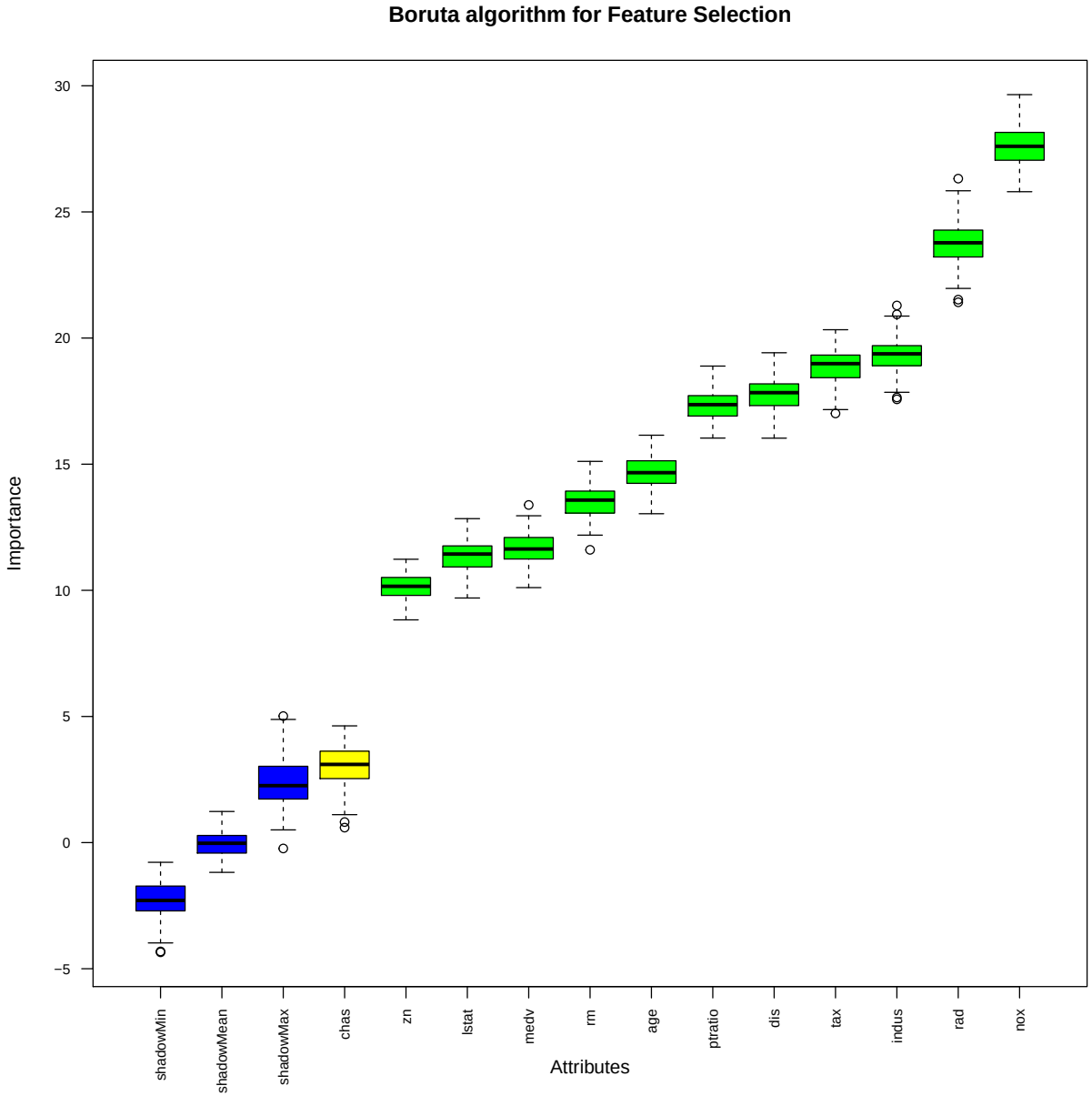
Boruta: Variable Importance

The importance of each original variable is ranked using the `Boruta` function.

Boruta is a feature selection algorithm, using Random Forest to select "important" variables.

It classifies the feature variables into 3 levels based on the p-value specified: Confirmed, Tentative, or Rejected.

```
## Boruta performed 99 iterations in 42.42 secs.
##  11 attributes confirmed important: age, dis, indus, lstat, medv and 6 more;
##  No attributes deemed unimportant.
##  1 tentative attributes left: chas;
```


Boruta algorithm for Feature Selection
Importance
30
25
20
15
10
5
0
−5
shadowMin
shadowMean
shadowMax
chas
zn
lstat
medv
rm
age
ptratio
dis
tax
indus
rad
nox
Attributes

From the above Boruta feature selection chart, eleven (11) feature variables are confirmed to be "important" variables, but chas (in yellow) is tagged "tentative".

As `chas` is very close to shadow (random) variables, its contribution to the model is questionable.

(Later we will confirm that its inclusion detracts from the model; its omission improves it.)

References on Boruta algorithm and R package:

Kursa, Jankowski, Rudnicki: "Boruta – A System for Feature Selection", in Fundamenta Informaticae 101 (2010) 271–285, available at https://pdfs.semanticscholar.org/85a8/b1d9c52f9f795fda7e12376e751526953f38.pdf

Kursa, Rudnicki: "Feature Selection with the Boruta Package", in Journal of Statistical Software September 2010, Volume 36, Issue 11, available at https://www.jstatsoft.org/index.php/jss/article/view/v036i11/v36i11.pdf

Kursa: "Boruta for those in a hurry", CRAN vignette available at https://cran.r-project.org/web/packages/Boruta/vignettes/inahurry.pdf

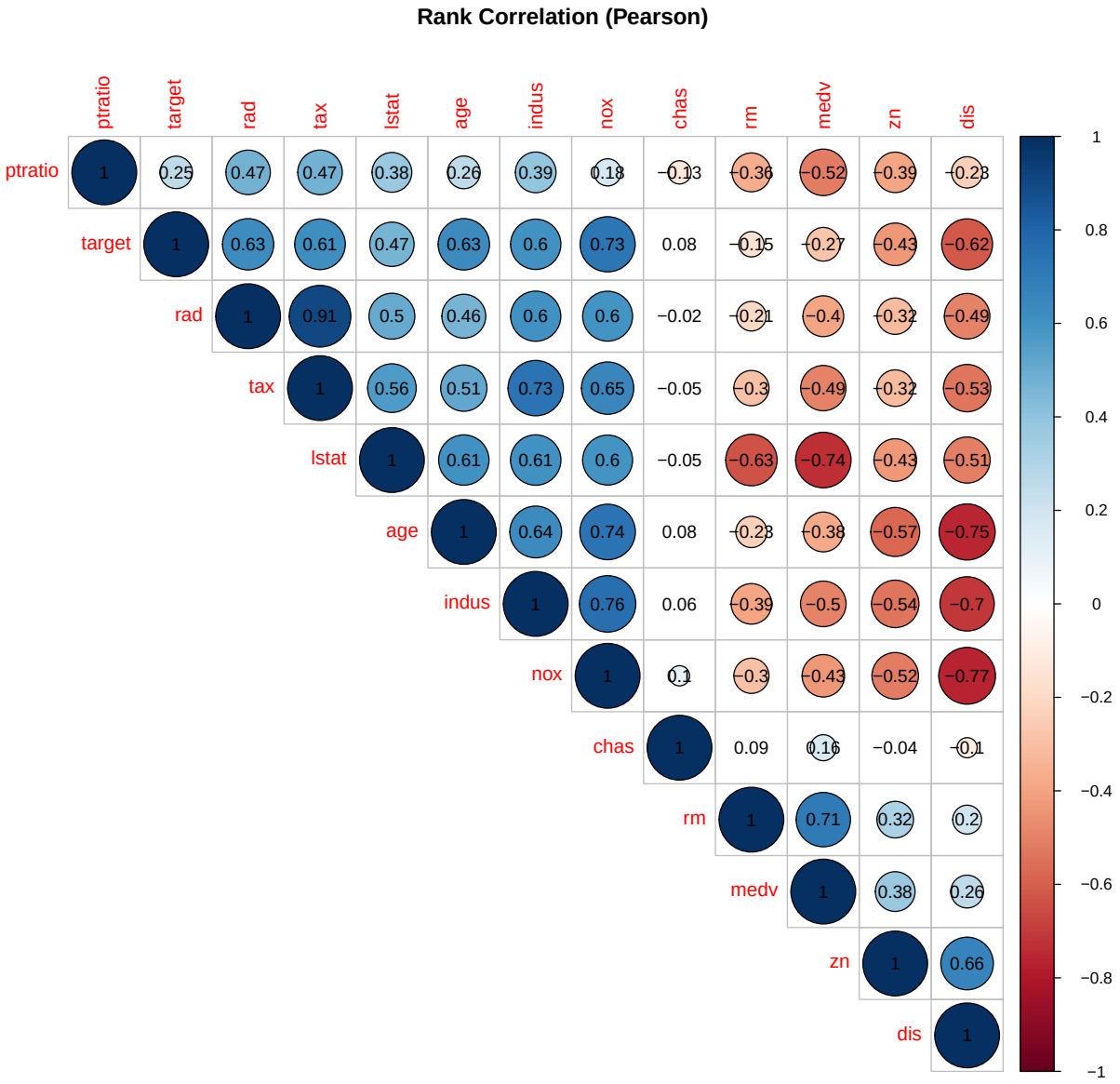
Check variable correlation

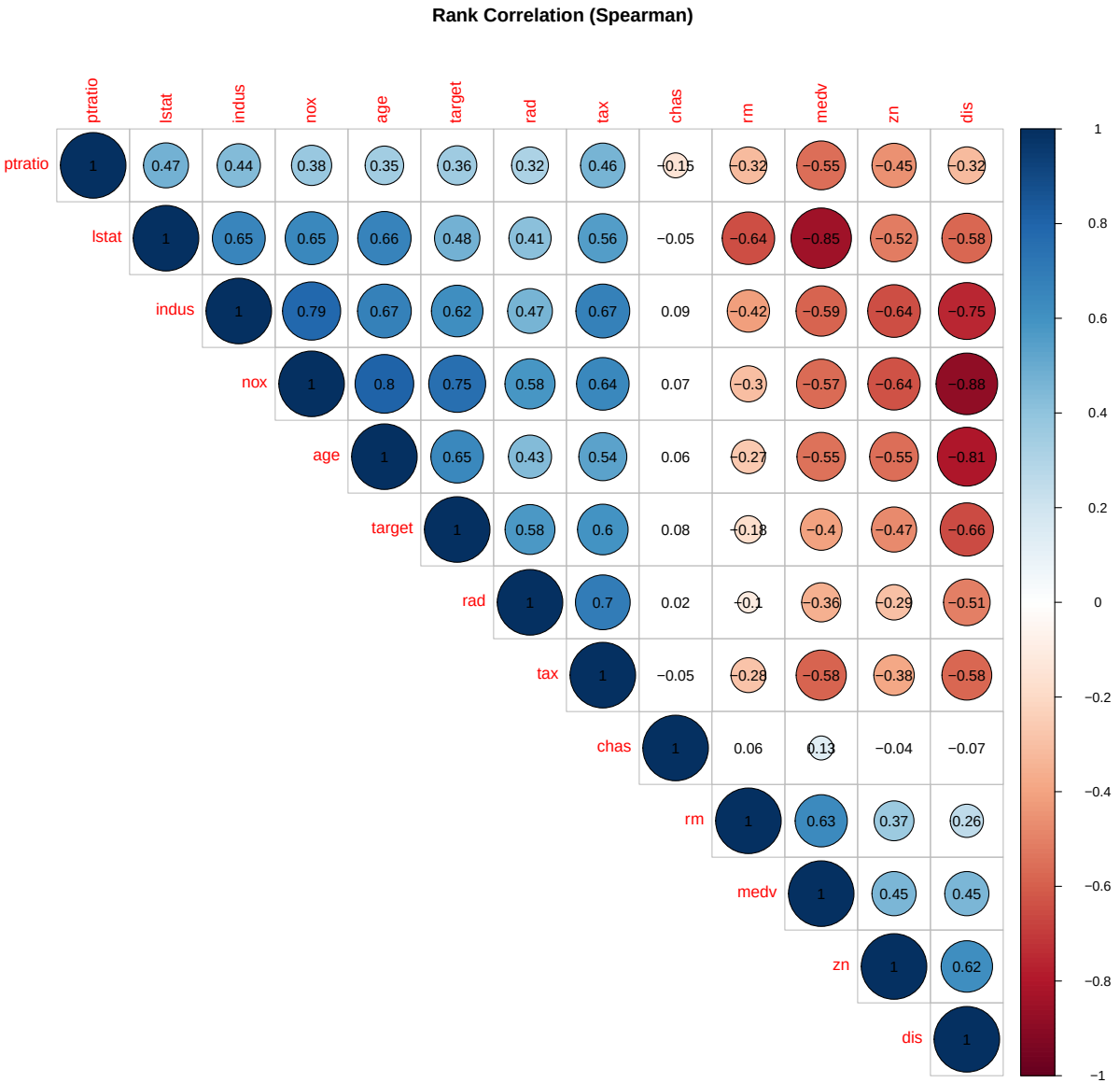
Variables which are *highly correlated* carry similar information and can affect model accuracy.

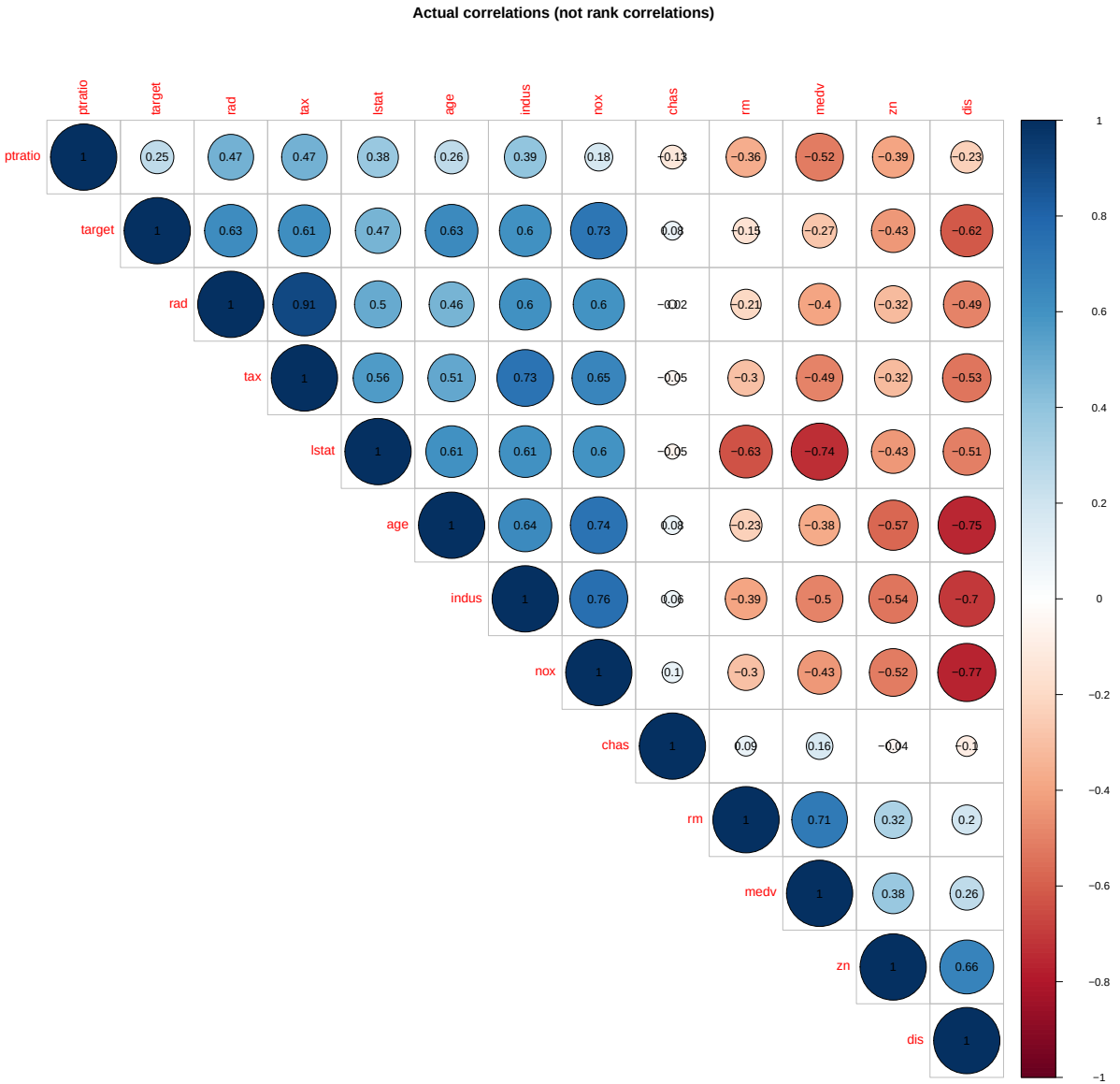
Highly correlated variables also impact estimation of model coefficients.

Let's determine which variables in the training datasets are highly correlated to each other.

Below are Pearson (Rank), Spearman (Rank), and ordinary correlations between variables:


Rank Correlation (Pearson)
ptratio
target
rad
tax
lstat
age
indus
nox
chas
rm
medv
zn
dis
1
0.8
0.6
0.4
0.2
0
−0.2
−0.4
−0.6
−0.8
−1


Rank Correlation (Spearman)
ptratio
lstat
indus
nox
age
target
rad
tax
chas
rm
medv
zn
dis
1
0.8
0.6
0.4
0.2
0
−0.2
−0.4
−0.6
−0.8
−1


Actual correlations (not rank correlations)
ptratio
target
rad
tax
lstat
age
indus
nox
chas
rm
medv
zn
dis
1
0.8
0.6
0.4
0.2
0
−0.2
−0.4
−0.6
−0.8
−1

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 **(highly negative correlation)**

Look for interactions

Let's focus on four variables listed below.

Not only are these variables are highly correlated with outcome variable `target`, but also the first 3 have high negative correlation with the fourth 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

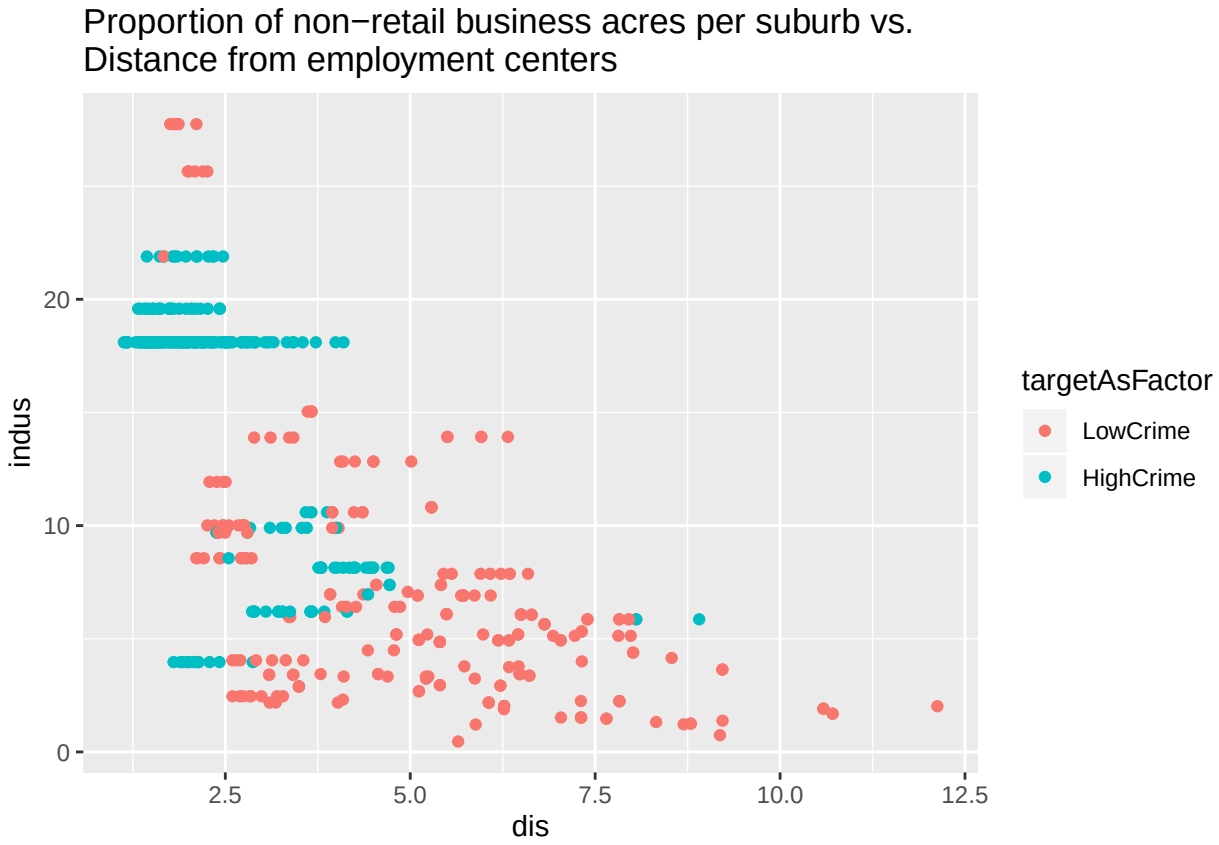
Let's analyze these four variables in more detail so that we can decide if we should add an interaction term for these variables.

Scatter plot of `age` vs. `dis`

From the above scatter plot, we can conclude that lots of properties which are constructed before 1940 are near to Boston employment centers.

The 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).

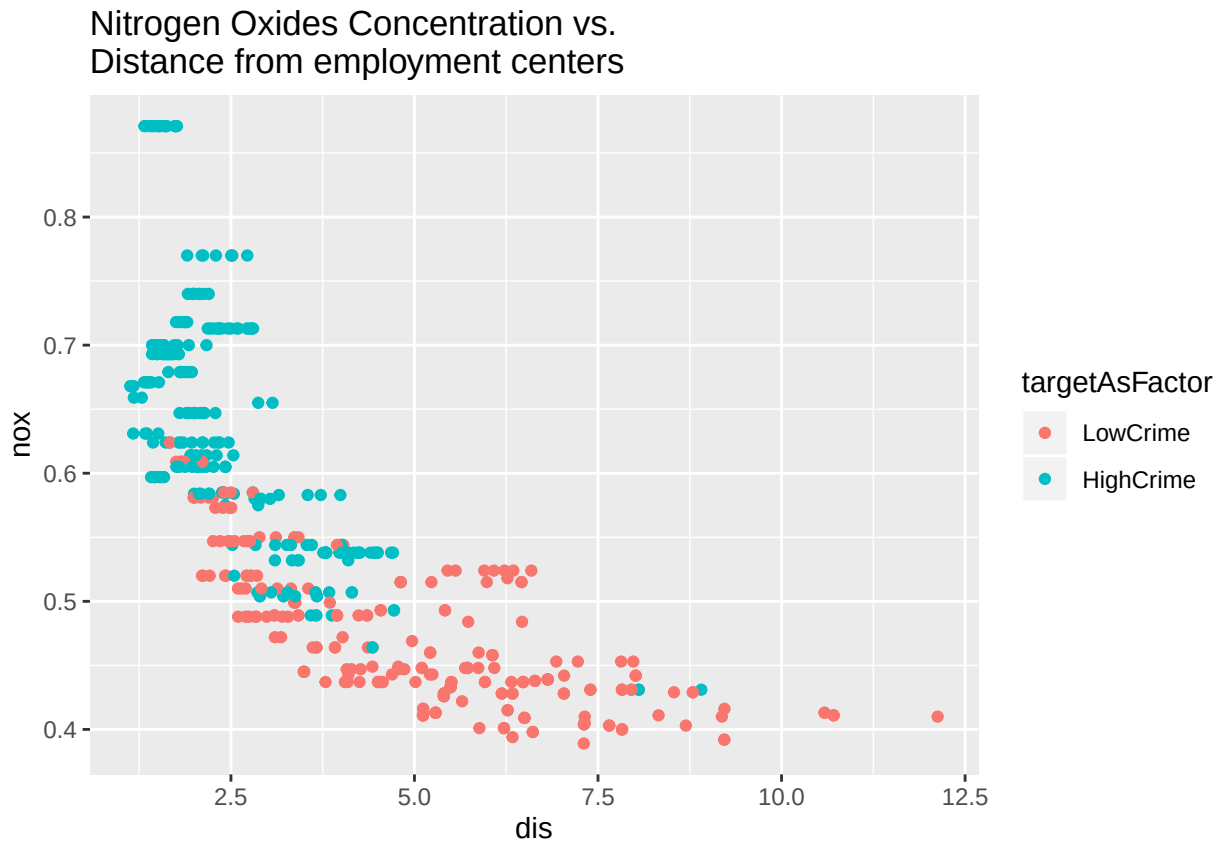
Scatter plot of `indus` vs. `dis`



The above scatter plot shows that proportion of non retail business acres *decreases* as we move away from the Boston employment centers.

From the colors, we also note that there is a decrease in the crime rate as we move away from the employment centers.

Scatter plot of `nox` vs. `dis`



The above scatter plot shows that there is very high percentage of Nitrogen Oxide (`nox`) near Boston employment centers.

This could be due to the area surrounding employment centers having more industry, leading to a greater percentage of Nitrogen Oxide. Additionally, we observe higher crime in such areas.

---

2. Data Preparation

Add Interaction Term

From the above scatter plots we can conclude that:

- ***old properties*** exist
- ***near to Boston employment centers***, which have
- a high percentage of ***Nitrogen Oxide***, and
- a high percentage of ***non-retail business acres***.

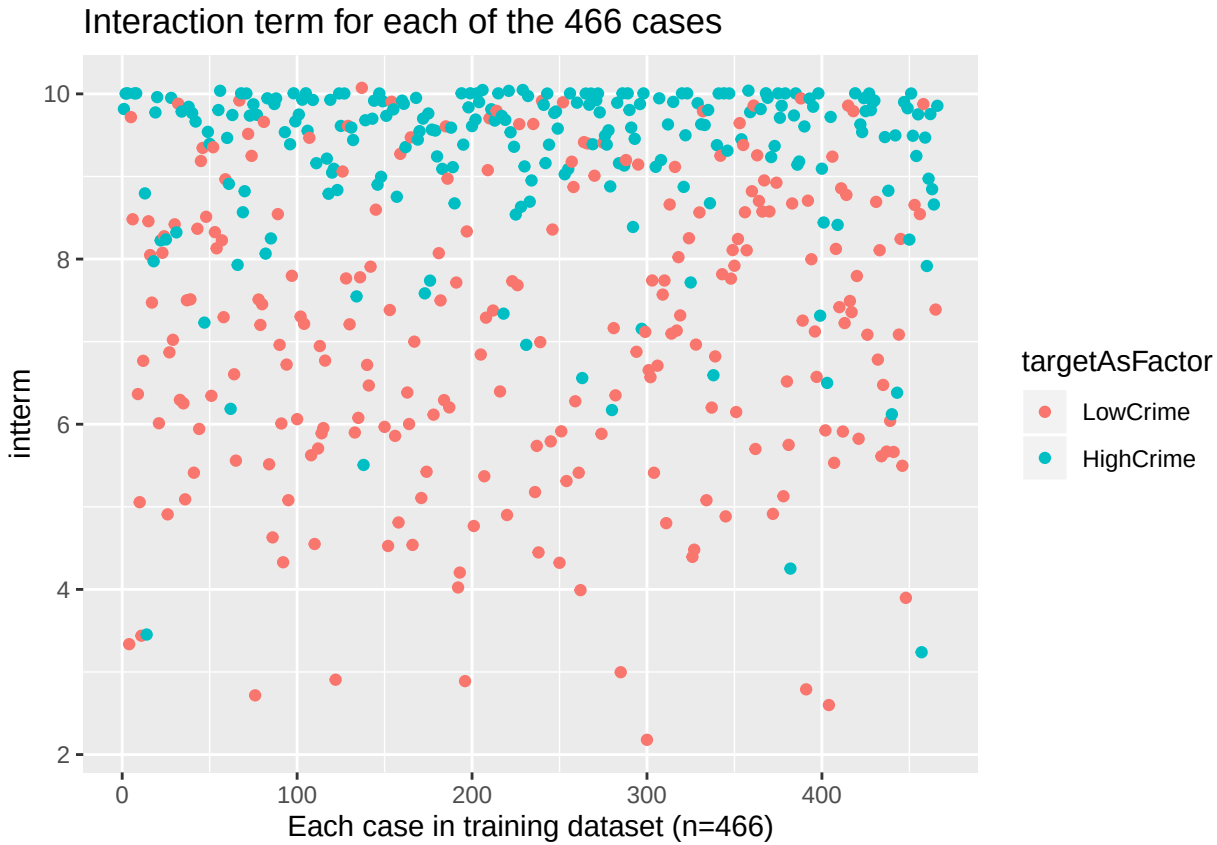
All these four terms are correlated (raw distance is negatively correlated, thus "near" is inverse). That can also be seen from the correlation graph above.

Let's add an interaction term to our dataset which will account for cumulative effect of all these four input variables.

We will define the interaction term using the following formula: $intterm = \sqrt{\frac{dis}{nox*indus} + age}$.

Let's see how newly added interaction term is helping us to categorize the 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 different target variable values nicely.

Specifically, high values of `intterm` generally correspond to high crime values

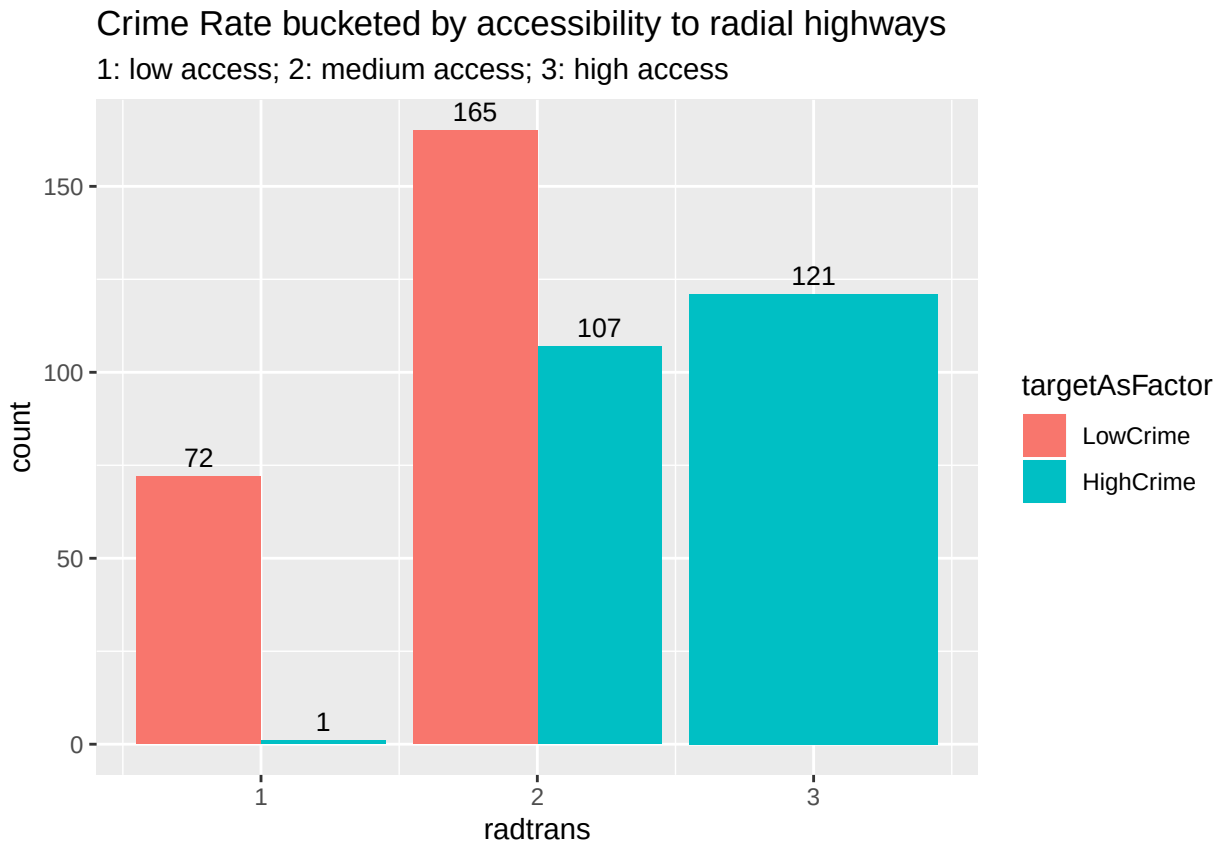
Bucketize `rad` variable

Add new variable by transforming `rad` variable into buckets

Let's divide the 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 divided into these three buckets after we add the newly transformed `rad` variable

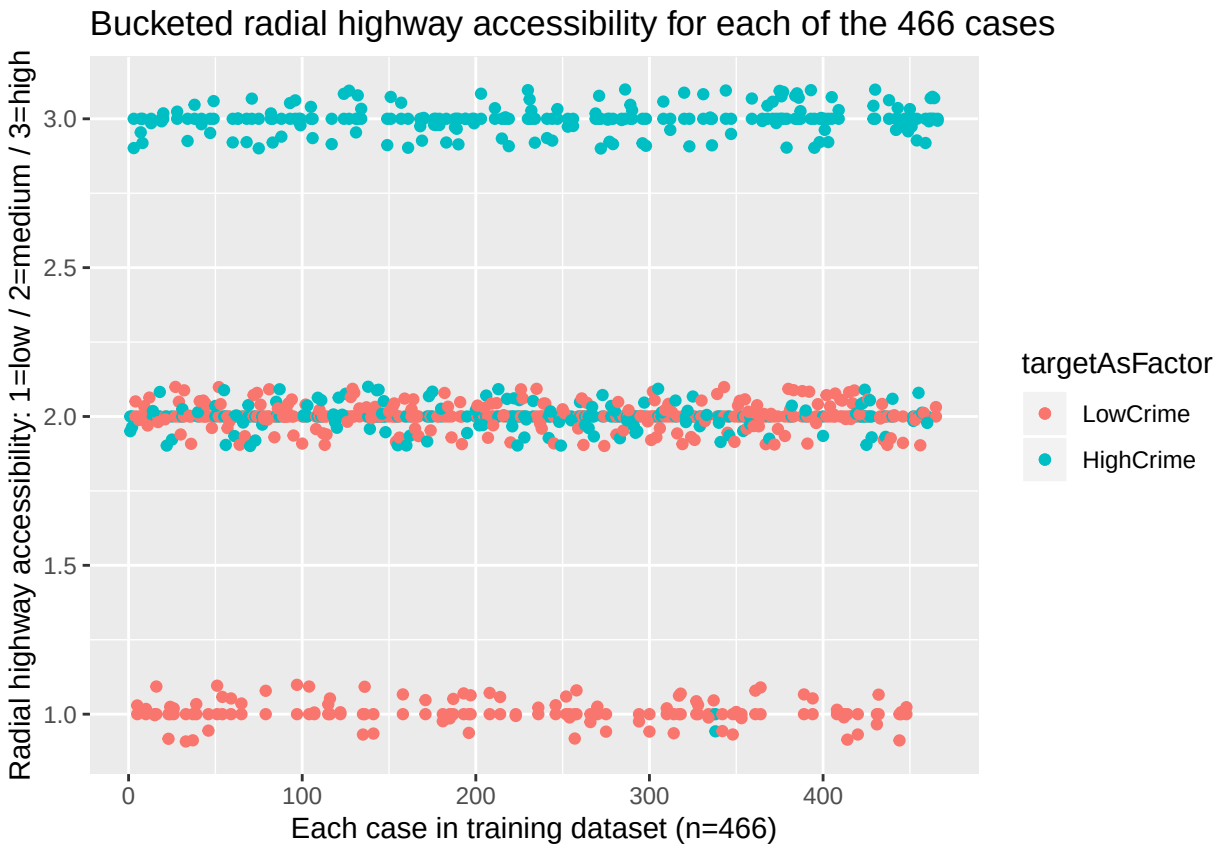


From the above bar plot we can see that

- ***crime rate is low*** if there is ***low accessibility*** to highways;
- for medium accessibility, crime rate is higher, and
- areas with ***highest level of accessibility*** to radial highways exhibit the ***highest crime rate***.

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

(Y-values of 1, 2, 3 have been "jittered" for display purposes)



Bucketize tax rate variable

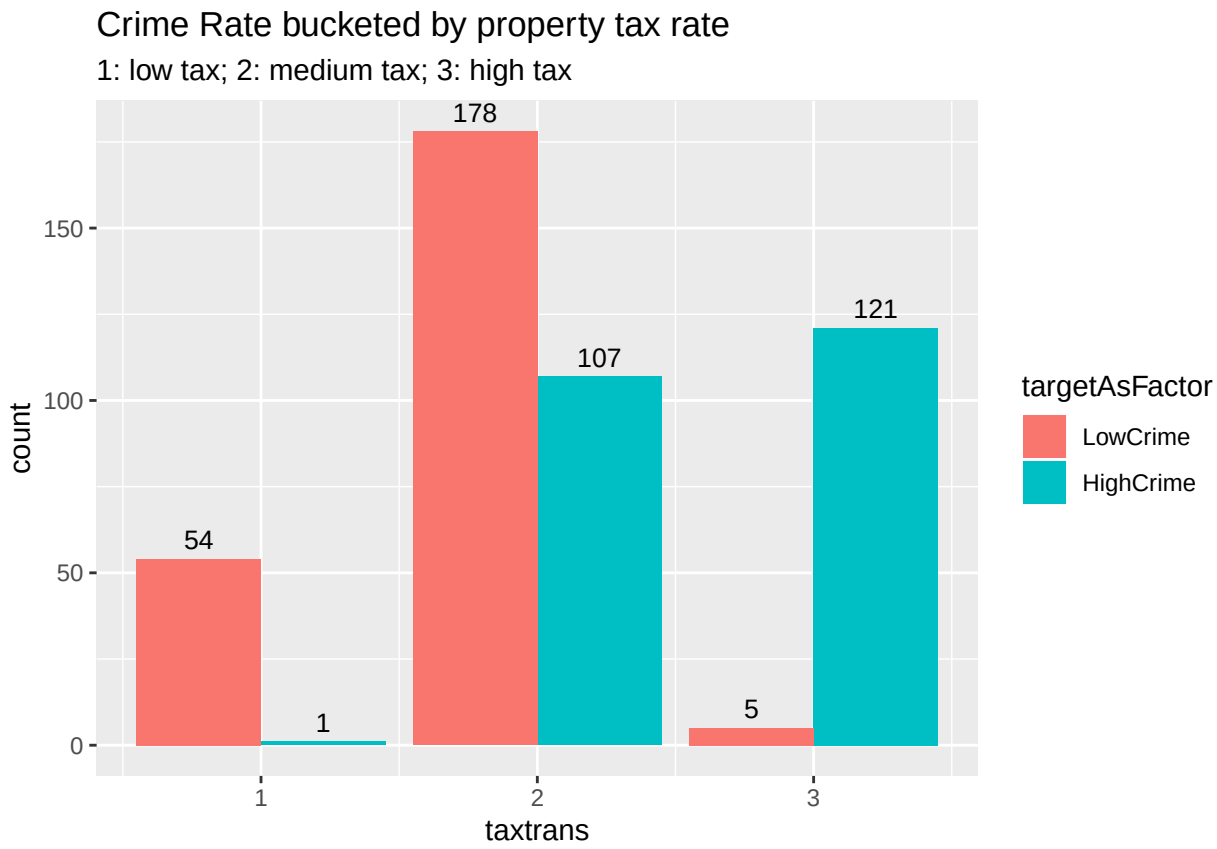
Add new variable by transforming tax rate variable into buckets

Let's divide data into three buckets of tax rate variable, using this formula: $round(\log(1 + tax)) - 4$

- **Bucket-1: Indicates low tax rate:** $tax \in [187, 243]$
- **Bucket-2: Indicates medium tax rate:** $tax \in [244, 469]$
- **Bucket-3: Indicates high tax rate:** $tax \in [665, 711]$

There are no observations between 470 and 664, but if there were, they would be bucketed into the medium category.

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



From the above bar plot we can see that

- the crime rate is low in the area of low tax rate,
- for medium tax rate areas, the crime rate is higher, and
- areas with highest tax rate exhibit the highest crime rate.

This make sense since areas with higher tax rate have high value properties. The data indicates that the crime rate tends to be higher in the areas where the property values are higher.

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

(Y-values of 1, 2, 3 have been "jittered" for display purposes)

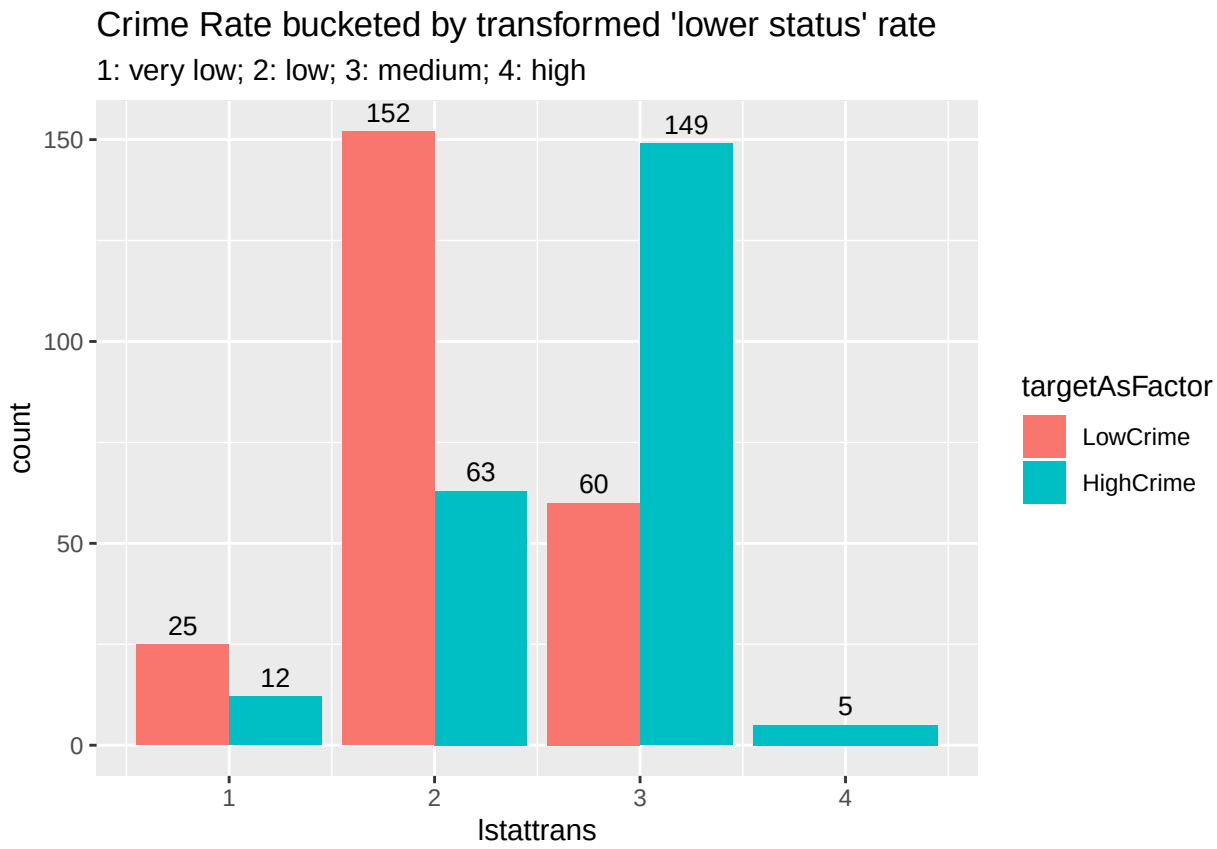
Bucketize "lstat" (lower status percent) variable

Add new variable by transforming `lstat` variable

Let's divide data into four buckets of lower status percent variable, using this formula: $round(\log(1 + lstat))$

- **Bucket#1**: Indicates **very low** lower status percent: **(lstat $<$ 4.5)**
- **Bucket#2**: Indicates **low** lower status percent: [**4.5,12.2**]
- **Bucket#3**: Indicates **medium** lower status percent: [**12.2,33.2**]
- **Bucket#4**: Indicates **high** lower status percent: **(lstat $>$ 33.2)**

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

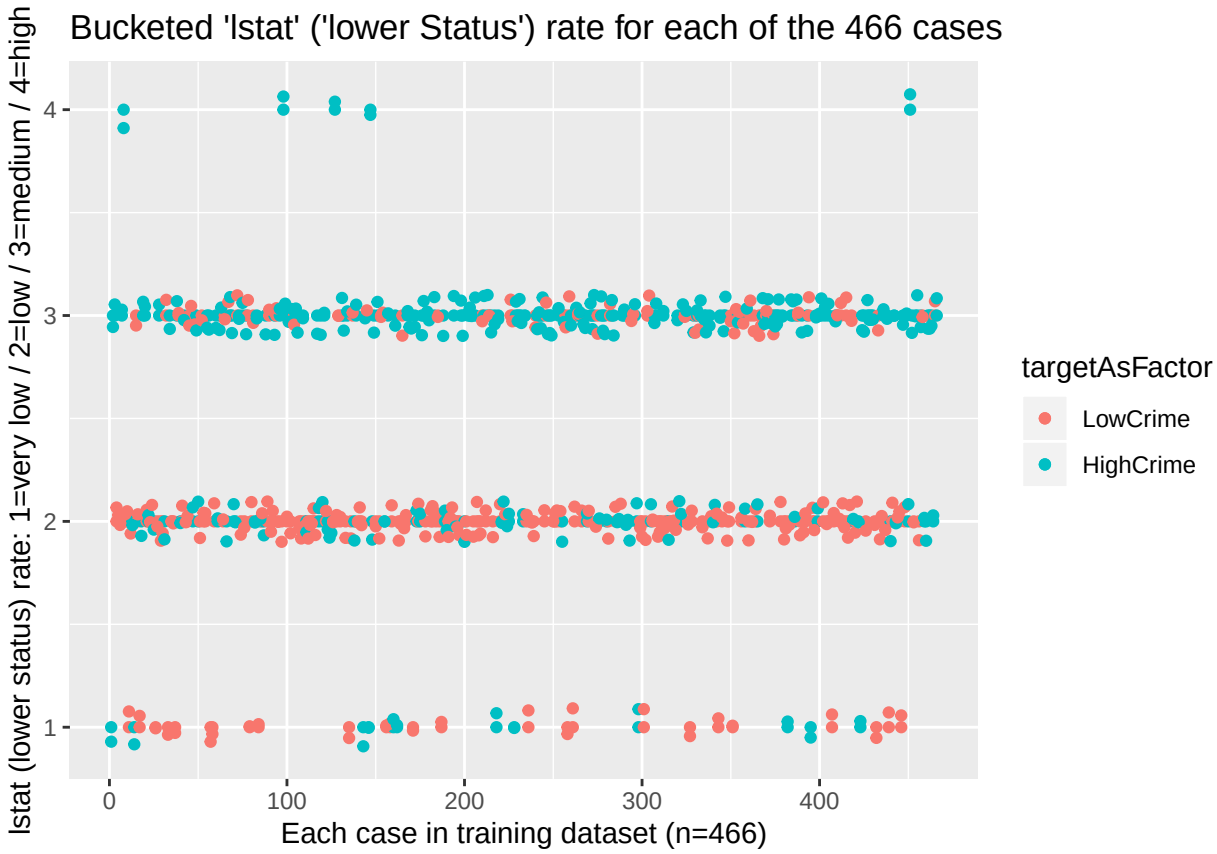


From the above bar plot we can see that crime rate increases as "lower status percent" increases.

From this we can conclude that areas with a high percentage of "lower status population" exhibit a higher crime rate.

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

(Y-values of 1, 2, 3, 4 have been "jittered" for display purposes)



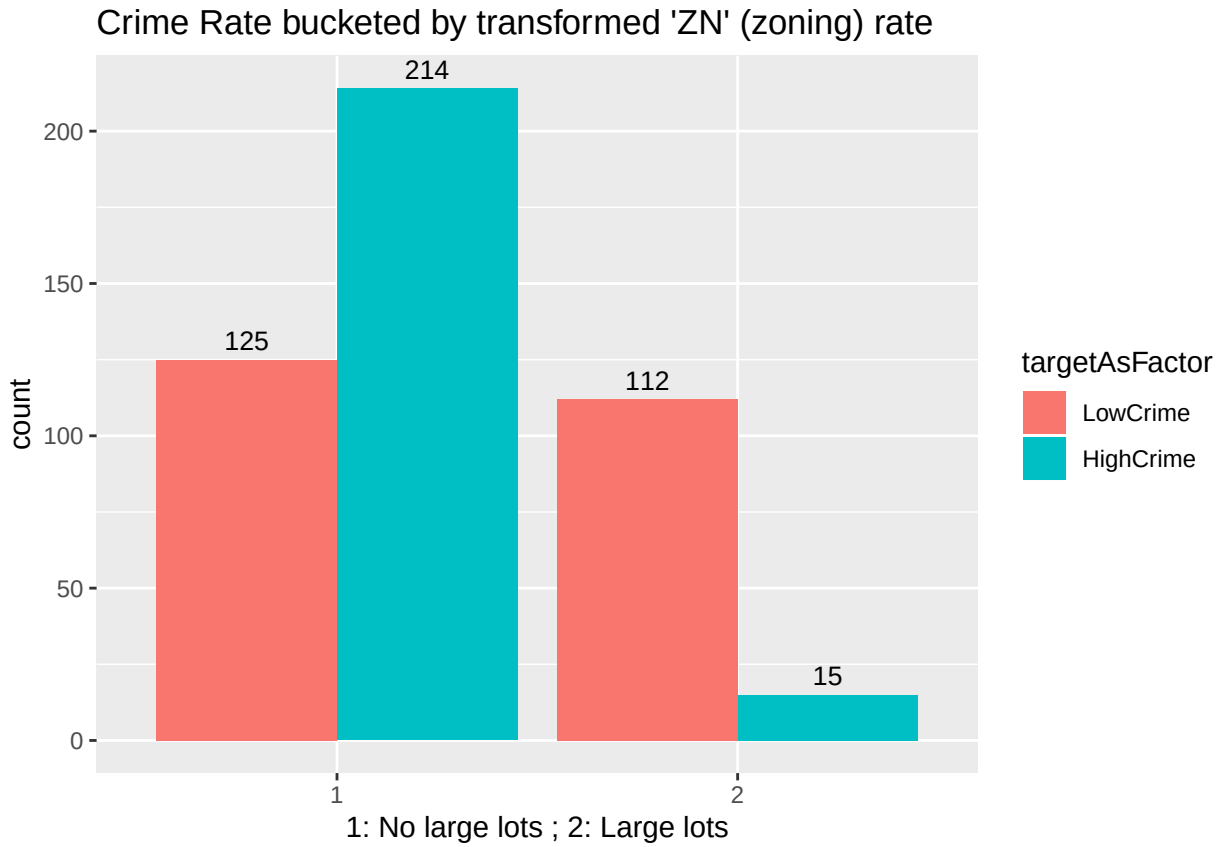
Bucketize ZN variable (zoning for large lots)

Add new variable by transforming zn variable

Let's divide 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**

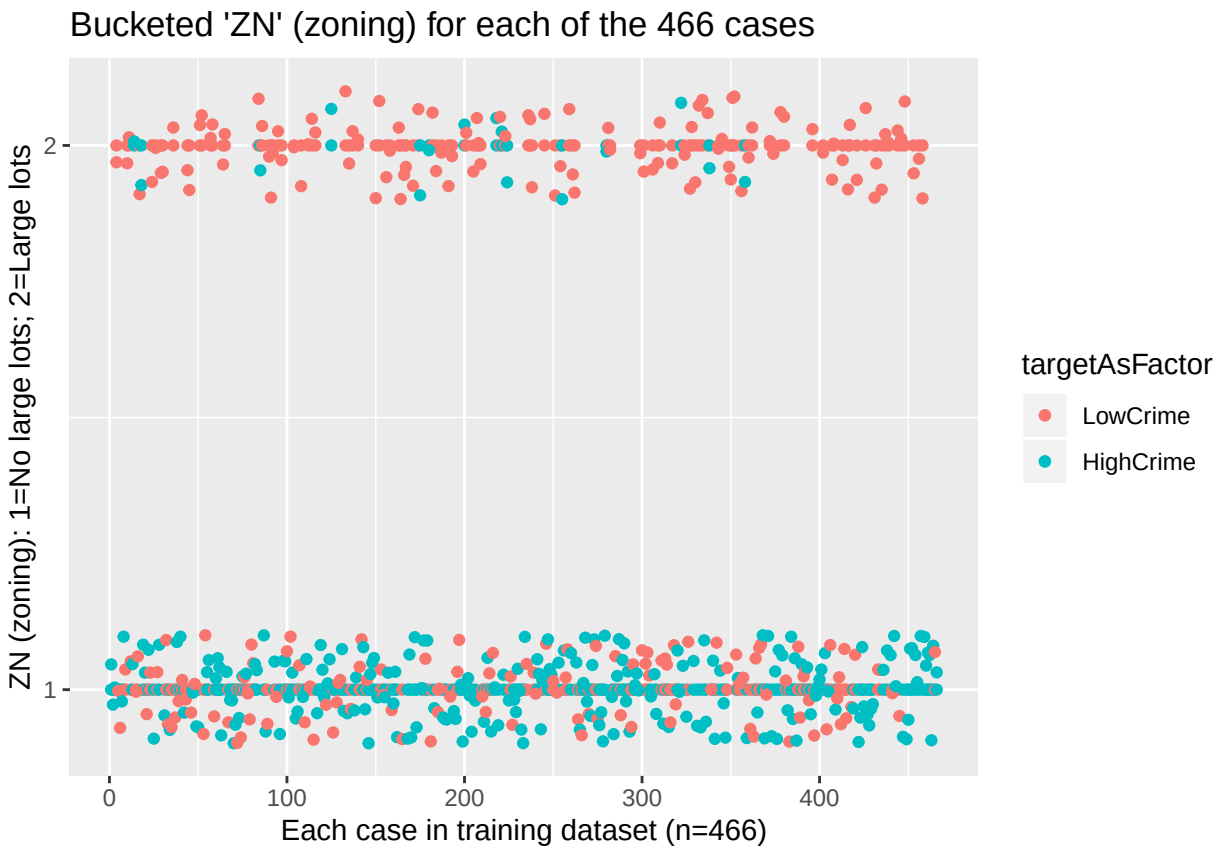
Let's see how the crime rate is divided into these two buckets after we transform ZN variable:



From the above bar chart we can see that

- areas where no residential land is zoned for larger lots exhibit a higher crime rate, while
- areas where a proportion of residential land is zoned for larger lots exhibit a lower crime rate.

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



The above illustrates that the higher crime rate occurs in areas which are not zoned for large residential lots.

---

3. Build Models

Model fitting and evaluation

Create binary logistic regression models, using different combinations of (transformed) variables.

We will use Cross-Validation to fit the model and evaluate it.

We will leverage the 5-fold cross validation technique.

Create functions to perform cross-validation of each model, and to assess model metrics.

Cross Validation

Cross-validation is a model validation technique 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 obtain an insight on how the model will generalize to an independent data set.

Model Evaluation Metrics

Definitions:

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

We will use following metrics for model evaluation and comparison:

- **ROC AUC** : AUC - ROC curve is a performance measurement for the classification problem at various thresholds. ROC is a probability curve, and AUC ("Area Under the Curve") represents the degree or measure of separability.
  - It tells how much the model is capable of distinguishing between classes.
  - The higher the AUC, the better the model.
- **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: $Accuracy = \frac{TP+TN}{TP+FP+TN+FN}$
- **Model Recall** : Recall is a metric that focuses on ***how many true positives*** are identified from ***total positive observations*** in the data set.
  - False Negatives (FN) are positive observations which our model failed to identify, and reduce the recall.
  - Formally, recall has following definition: $Recall = \frac{TP}{TP+FN}$.
- **Model Precision** : Precision is a metric that focuses on ***how many observations are truly positive*** out of the total number of cases which the model ***identified as positive***.
  - False Positives (FP) are negative observations which our model misidentified as positive, and reduce the precision.
  - Formally, precision has following definition : $Precision = \frac{TP}{TP+FP}$.

Now we are clear on our model fitting and evaluation method (Cross Validation).

Additionally, we have model evaluation metrics which we will use to compare the model effectiveness.

We are now ready to build different models and assess their 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 predictors:

- **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

```
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 Metrics 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 as a baseline to compare model effectiveness as we add more variables to, or otherwise transform, the model.

2. Model with Interaction Term

Let's add the 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 Metrics 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 confirms that the interaction term we have created is helping improve model effectiveness towards predicting the crime rate.

3. Model with transformed variables

In addition to the interaction term, we will also incorporate newly-created transformed variables in this model

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

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

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

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

The above metrics are even better, beating the performance of the model with interactive term (Model 2).

This confirms that the interaction term, along with our transformed variables, are further helping improve model effectiveness towards predicting the crime rate.

4. Model including all original variables

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

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

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

The above metrics are better than the 8-variable base model, which is as expected since more information was brought into the model by inclusion of additional important variables.

The accuracy is close to that of 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 of all other variables in the `train.df` dataset which the exception of `chas`, which was "tentative", i.e., not as important as other variables.

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

```
## [1] "Printing Metrics 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"
```

The above metrics indicate that this model performs better than the 4th model, which includes all 12 variables.

Removing the less important variable `chas` improves the model performance:

the accuracy and precision both beat the first two models above,

but are slightly below the results for model #3, which uses 8 original variables plus 3 interaction and transformed variables.

---

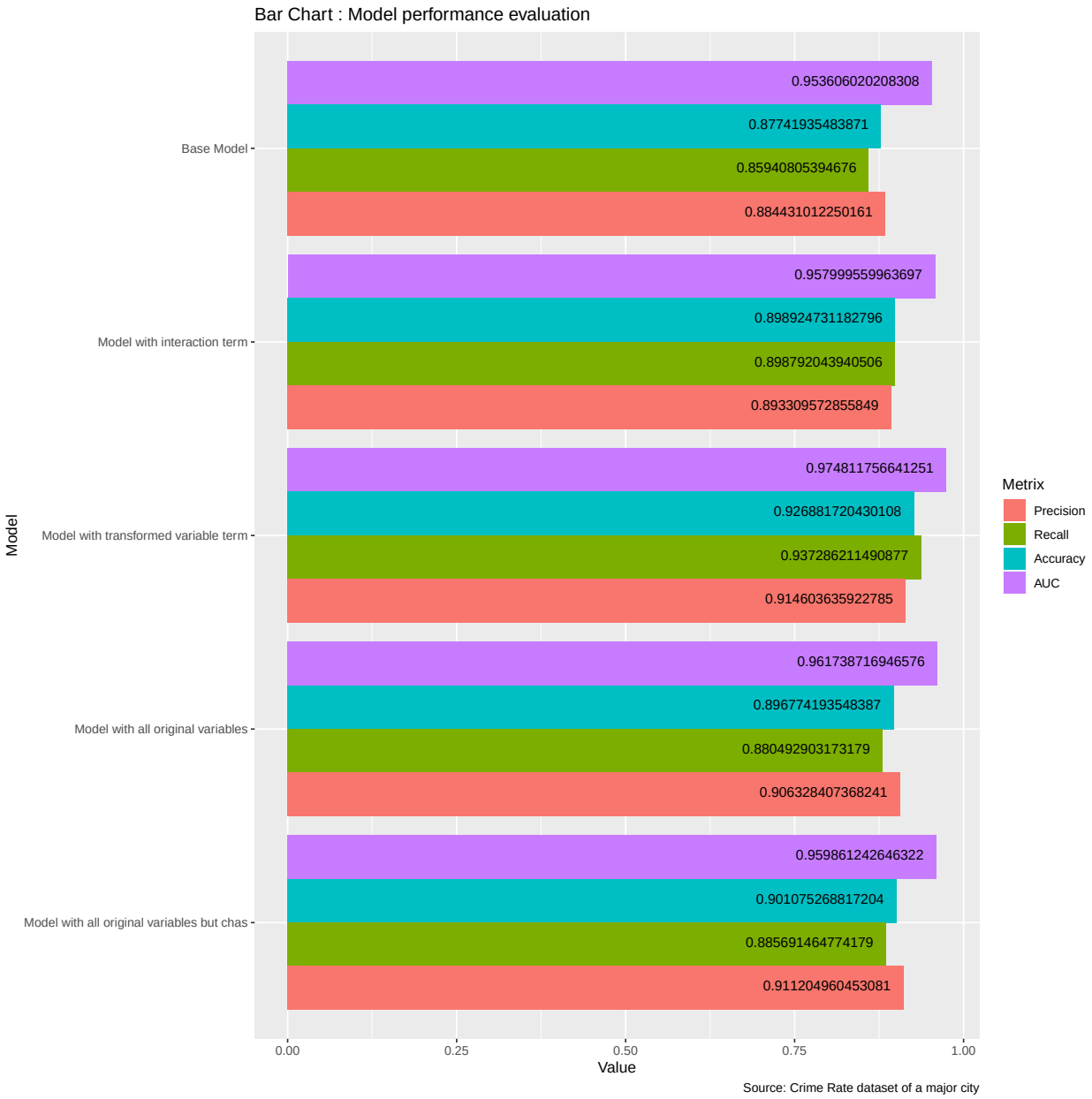
4. Select Models

Model #3 is best

From the above analysis it is demonstrated that the best performing model is model #3, which incorporates

- 8 base predictors, along with
- interaction term, and
- transformed variables.

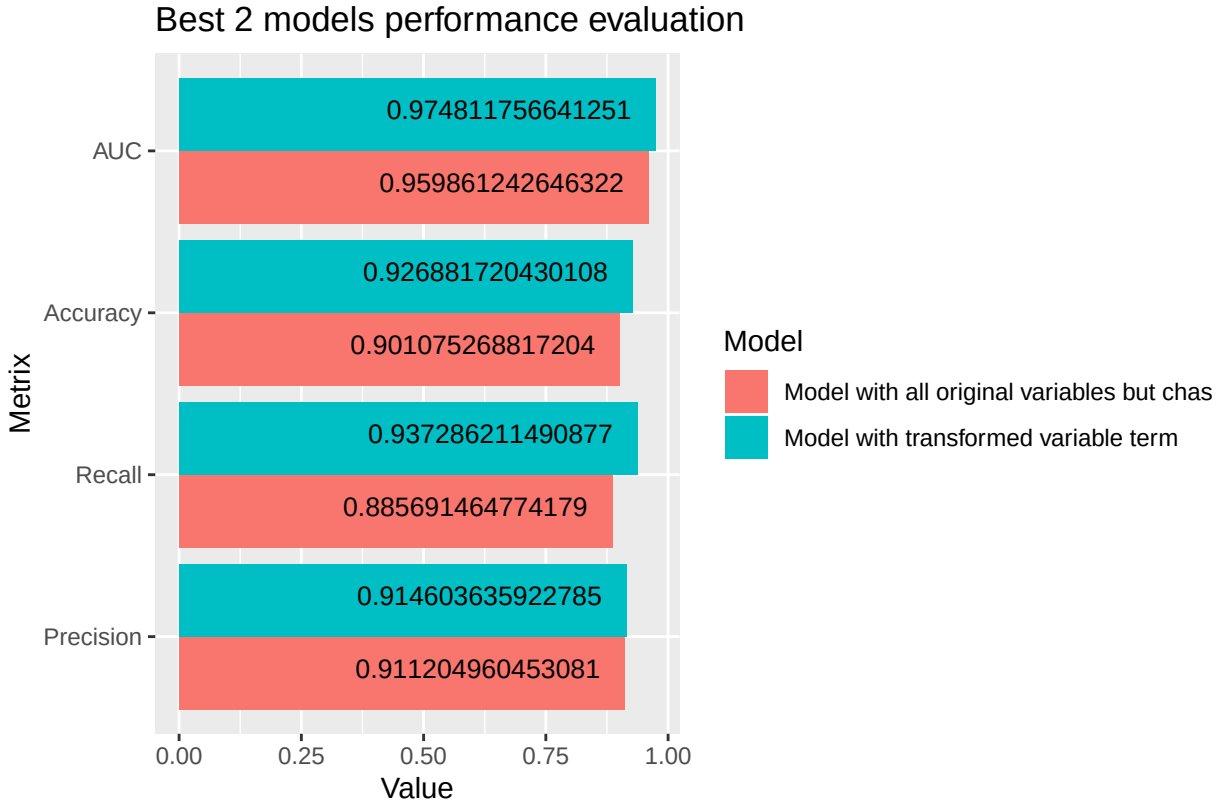
The below barchart illustrates the metrics associated with the various different models:


Bar Chart : Model performance evaluation
Base Model
Model with interaction term
Model with transformed variable term
Model with all original variables
Model with all original variables but chas
Model
0.953606020208308
0.87741935483871
0.85940805394676
0.884431012250161
0.957999559963697
0.898924731182796
0.898792043940506
0.893309572855849
0.974811756641251
0.926881720430108
0.937286211490877
0.914603635922785
0.961738716946576
0.896774193548387
0.880492903173179
0.906328407368241
0.959861242646322
0.901075268817204
0.885691464774179
0.911204960453081
0.00
0.25
0.50
0.75
1.00
Value
Source: Crime Rate dataset of a major city
Metrix
Precision
Recall
Accuracy
AUC

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, while the ***precision*** values remain very close.
The discussion point here is that since the ***interaction term*** and ***transformed variables*** increase the complexity of the model, we should work with domain experts to determine whether

- such increased complexity on the model outweighs the increased performance, and
- the model is easy to explain and implement compared vs. the simple model (without variable transformation and interaction term.)



5. Predictions

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))-4

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_v2.csv", row.names = FALSE, sep=",")
```

---

6. Source code

```
options(scipen = 999, digits=6, width=150)

library(dplyr)
library(ggplot2)
library(corrplot)
library(Hmisc)
library(ROCR)
library(parallel)
library(Boruta)
library(kableExtra)

# Set working directory and load the training data

mydir = "C:/Users/Michael/Dropbox/priv/CUNY/MSDS/201909-Fall/DATA621_Nasrin/20191030_HW3_Crime/HW-3"
setwd(mydir)
train.df = read.csv("./Data/crime-training-data_modified.csv")


# 1. Data Exploration

## **Load Data**


numhead = 10
head(train.df,n = numhead) %>%
  kable(row.names = 1:numhead) %>%
  kable_styling(c("striped", "bordered"))

print(paste("Number of columns (features) = ", ncol(train.df)))
print(paste("Number of rows (observations) = ", nrow(train.df)))


#### **Let's analyze the datatypes of each column:**

str(train.df)
targetAsFactor <- factor(train.df$target, labels=c("LowCrime","HighCrime"))
```

```
str(targetAsFactor)
summary(targetAsFactor)

## **Check missing values**


miss.cols = apply(train.df, 2, function(x) any(is.na(x)))
print(paste("Number of columns with missing values = ", length(colnames(miss.cols))))

## **Check Class Imbalance**


ggplot(train.df, aes(target, fill = targetAsFactor)) +
  geom_bar() +
  geom_text(stat='count', aes(label=..count..), vjust=1) +
  labs(title="Bar Chart: Counts for target variable",
       caption="Source: Crime dataset for major city (Boston)")

## **Boruta: Variable Importance**


set.seed(1)

Boruta(target ~ .  , data=train.df)->Bor.hvo
print(Bor.hvo)
plot(Bor.hvo,  cex.axis=0.75, las=2, main="Boruta algorithm for Feature Selection")

## **Check variable correlation**



  res2<-rcorr(as.matrix(train.df))
  respearson=rcorr(as.matrix(train.df),type = "pearson")
  resspearman=rcorr(as.matrix(train.df),type = "spearman")
  res3 <- cor(as.matrix(train.df))

#### Pearson rank correlation
  corrplot(corr = respearson$r, type = "upper", outline = T, order="hclust",
```

```
          p.mat = respearson$P, sig.level = 0.05, insig = "blank", addCoef.col = "black",
          title = "\nRank Correlation (Pearson)",
          number.cex = 0.9, number.font = 1, number.digits = 2 )

#### Spearman rank correlation
  corrplot(corr = resspearman$r, type = "upper", outline = T,  order="hclust",
          p.mat = resspearman$P, sig.level = 0.05, insig = "blank", addCoef.col = "black",
          title = "\nRank Correlation (Spearman)",
          number.cex = 0.9, number.font = 1, number.digits = 2)

#### actual correlations (not rank correlations)
  corrplot(corr = res3, type = "upper", outline = T,  order="hclust",
          sig.level = 0.05, insig = "blank", addCoef.col = "black",
          title = "\nActual correlations (not rank correlations)",
          number.cex = 0.9, number.font = 1, number.digits = 2 )

## **Look for interactions**

### **Scatter plot of age vs. dis**


ggplot(train.df, aes(x = dis, y=age, color=targetAsFactor)) +
geom_point() + ggtitle("Percentage of aged housing vs. Distance from employment centers")


### **Scatter plot of indus vs. dis**

ggplot(train.df, aes(x = dis, y=indus, color=targetAsFactor)) +
geom_point() + ggtitle("Proportion of non-retail business acres per suburb vs. \nDistance from employment centers")

### **Scatter plot of nox vs. dis**

ggplot(train.df, aes(x = dis, y=nox, color=targetAsFactor)) +
geom_point() + ggtitle("Nitrogen Oxides Concentration vs. \nDistance from employment centers")


# 2. Data Preparation
```

```
## **Add Interaction Term**

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

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

ggplot(train.df, aes(x = seq(1:nrow(train.df)), y=intterm, color=targetAsFactor)) +
geom_point() + ggtitle("Interaction term for each of the 466 cases") + labs(x = "Each case in training dataset (n=466)")

## **Bucketize rad variable**

train.df$radtrans = round(log(1+train.df$rad))

ggplot(train.df, aes(x=radtrans, fill=targetAsFactor))+
geom_bar(position=position_dodge()) +
geom_text(stat='count', aes(label=..count..), vjust=-0.5, color="black",
    position = position_dodge(0.9), size=3.5) +
ggtitle(label="Crime Rate bucketed by accessibility to radial highways",
        subtitle="1: low access; 2: medium access; 3: high access")

ggplot(train.df, aes(x = seq(1:nrow(train.df)), y=radtrans, color=targetAsFactor)) +
geom_point() +
  geom_jitter(width = 0, height = 0.1) +
  ggtitle("Bucketed radial highway accessibility for each of the 466 cases") +
  labs(x = "Each case in training dataset (n=466)",
       y = "Radial highway accessibility: 1=low / 2=medium / 3=high")

## **Bucketize tax rate variable**


# round(log(1+tax)) gives results in the set {5,6,7}
# So, subtracting 4 from the above makes the results {1,2,3} which makes more sense.
train.df$taxtrans = round(log(1+train.df$tax))-4

ggplot(train.df, aes(x=taxtrans, fill=targetAsFactor))+
geom_bar(position=position_dodge()) +
geom_text(stat='count', aes(label=..count..), vjust=-0.5, color="black",
    position = position_dodge(0.9), size=3.5) +
```

```
ggtitle(label="Crime Rate bucketed by property tax rate",
        subtitle="1: low tax; 2: medium tax; 3: high tax")


ggplot(train.df, aes(x = seq(1:nrow(train.df)), y=taxtrans, color=targetAsFactor)) +
geom_point() +
  geom_jitter(width = 0, height = 0.1) +
  ggtitle("Bucketed tax rate for each of the 466 cases") +
  labs(x = "Each case in training dataset (n=466)",
       y = "tax rate: 1=low / 2=medium / 3=high")

## **Bucketize "lstat" (lower status percent) variable**

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

ggplot(train.df, aes(x=lstattrans, fill=targetAsFactor))+
geom_bar(position=position_dodge()) +
geom_text(stat='count', aes(label=..count..), vjust=-0.5, color="black",
    position = position_dodge(0.9), size=3.5) +
ggtitle(label="Crime Rate bucketed by transformed 'lower status' rate",
        subtitle="1: very low; 2: low; 3: medium; 4: high")


ggplot(train.df, aes(x = seq(1:nrow(train.df)), y=lstattrans, color=targetAsFactor)) +
geom_point() +
  geom_jitter(width = 0, height = 0.1) +
  ggtitle("Bucketed 'lstat' ('lower Status') rate for each of the 466 cases") +
  labs(x = "Each case in training dataset (n=466)",
       y = "lstat (lower status) rate: 1=very low / 2=low / 3=medium / 4=high")

## **Bucketize ZN variable** (zoning for large lots)

train.df$zntrans = round(log(1+train.df$zn^0.5))/2+1

ggplot(train.df, aes(x=as.factor(train.df$zntrans), fill=targetAsFactor))+
  geom_bar(position=position_dodge()) +
  geom_text(stat='count', aes(label=..count..), vjust=-0.5, color="black",
    position = position_dodge(0.9), size=3.5) +
```

```
ggtitle(label="Crime Rate bucketed by transformed 'ZN' (zoning) rate") +
xlab("1: No large lots ; 2: Large lots")

ggplot(train.df, aes(x = seq(1:nrow(train.df)), y=zntrans, color=targetAsFactor)) +
geom_point() +
  geom_jitter(width = 0, height = 0.1) +
  scale_y_continuous(name="ZN (zoning): 1=No large lots; 2=Large lots", breaks=c(1,2)) +

  ggtitle("Bucketed 'ZN' (zoning) for each of the 466 cases") +
  labs(x = "Each case in training dataset (n=466)")

# 3. Build Models

## **Model fitting and evaluation**

cross.validation <- function(K, model.formula, LCdata) {
  auclist = NULL
  accuracylist = NULL
  recalllist = NULL
  precisionlist = NULL
  fold.size <- nrow(LCdata) / K

  for(k in 1:K){


    testing.index <- (k-1)*fold.size + 1:fold.size

    training <- LCdata[-testing.index,]
    testing.org <- LCdata[testing.index,]
    testing = testing.org[1:nrow(testing.org), names(testing.org)[names(testing.org) != 'target']]

    model <- glm(formula = model.formula,
                 family = "binomial", data = training)
    predicted <- predict(model, newdata = testing,type="response")
    pred <- prediction(predicted, testing.org$target)
    ind = which.max(round(slot(pred, 'cutoffs')[[1]],1) == 0.5)
    perf <- performance(pred, measure = "tpr", x.measure = "fpr")
```

```
    auc.tmp <- performance(pred,"auc");
    auc <- as.numeric(auc.tmp@y.values)
    auclist[k] = auc

    acc.perf = performance(pred, measure = "acc")
    acc = slot(acc.perf, "y.values")[[1]][ind]
    accuracylist[k] = acc

    prec.perf = performance(pred, measure = "prec")
    prec = slot(prec.perf, "y.values")[[1]][ind]
    precisionlist[k] = prec

    recall.perf = performance(pred, measure = "tpr")
    recall = slot(recall.perf, "y.values")[[1]][ind]
    recalllist[k] = recall
  }
  return(list("AUC" = mean(auclist), "Accuracy" = mean(accuracylist),  "Recall" = mean(recalllist), "Precision" = mean(precisionlist)))
}

df.metrix <<- NULL
print.model.matrix = function(model.name, matrixobj)
{

  print(paste("Printing Metrics for model: ", model.name))
  for(i in 1 : length(matrixobj))
  {
    df = data.frame("Model" = model.name, "Metrix"=names(matrixobj)[[i]], "Value" = matrixobj[[i]])
    df.metrix <<- rbind(df, df.metrix)
    print(paste(names(matrixobj)[[i]], ":", matrixobj[[i]]))
  }

}



## **Cross Validation**

## **1. Base Model**
```

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

## **2. Model with Interaction Term**

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)

## **3. Model with transformed variables**

model.metrix = cross.validation(5, "target~ rad+tax+lstat+age+indus+nox+zn+dis + intterm + radtrans + taxtrans", train.df)
print.model.matrix("Model with transformed variable term", model.metrix)
## **4. Model including all original variables**

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

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

# 4. Select Models

## Model #3 is best

ggplot(df.metrix, aes(x = Model, y=Value, fill=Metrix)) +
  geom_bar(stat='identity', position=position_dodge()) +
  labs(title="Bar Chart : Model performance evaluation",
            caption="Source: Crime Rate dataset of a major city") +
geom_text(aes(label=Value), vjust=0.25, hjust=1.1, color="black",
    position = position_dodge(0.9), size=3.5) +
coord_flip()


### **Discussion:**


selected.metrix <- df.metrix[df.metrix$Model %in% c('Model with all original variables but chas',
```

```
                                              'Model with transformed variable term'), ]
ggplot(selected.metrix, aes(x = Metrix, y=Value, fill=Model)) +
  geom_bar(stat='identity', position=position_dodge()) +
  labs(title="Best 2 models performance evaluation",
            caption="Source: Crime Rate dataset of a major city") +
  geom_text(aes(label=Value), vjust=0.25, hjust=1.1, color="black",
    position = position_dodge(0.9), size=3.5) +
  coord_flip()


# 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))-4

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_V2.csv", row.names = FALSE, sep=",")
```