Text

Description automatically generated

Text

Description automatically generated

1. We will import the Boston Housing Dataset from the ISLR package. After importing, we will do some preliminary Exploratory Data Analysis (EDA) by checking for things like -> presence of blank or ‘NA’ values, going through the summary of the entire dataset to visually single out values with a huge range or presence of outliers. We will also see the metadata of the dataset.

boston <- Boston

# View(boston)

contents(boston) # View the metadata

names(boston) # View the column names

str(boston) # View the structure of the data frame

sum(is.na(boston))

head(boston)

Text

Description automatically generated

We see that the dataset is composed of 506 observations within 14 fields/variables. Also, no blanks or NA values are present.

We will now see the summary statistics of the dataset for further analysis.

summary(boston)

A black screen with white text

Description automatically generated with low confidence

We see that the fields – zn, crim, black, and rm have a huge range along with significant differences in their mean and median. This indicates the presence of outliers. We can verify the same by looking at the boxplots of these variables below.

par(mfrow = c(1,4))

boxplot(boston$crim, main = "crim")

boxplot(boston$zn, main = "zn")

boxplot(boston$rm, main = "rm")

boxplot(boston$black, main = "black")

Chart, box and whisker chart

Description automatically generated

As expected, there are a lot of outliers in these four variables, especially in crim, zn, and black. We will now see the linear correlations between variables. This will give us a fair bit of hint regarding the presence of multi-collinearity. To get a better idea for multi-collinearity we can also perform the Farrar-Glauber Test.

corrplot(cor(boston))

A picture containing chart

Description automatically generated

We see that there are some high degrees of correlation between the fields. Example – ‘rad’ and ‘tax’ have a high positive correlation which makes sense as properties on or near the highways may have higher sale value owing to ease of access. Conversely, there seems to be a high negative correlation between the variables ‘nox’ and ‘dis’ which is again logical as the concentration of pollutants tends to be higher in urban work-centers rather than sub-urban areas.

1. In this dataset – Y (Dependent Variable) will be ‘medv’ i.e the median price/property value of the house that needs to be predicted.

X(Independent Variables) will be the remaining variables. However, the number of independent variables will vary across models and may have different interactions in each model.

i.) **Multiple Linear Regression** -> I will be pasting only the parsimonious model’s code here for ease.

# We will now include the following interactions -> rm\*lstat, rm\*rad, and lstat\*rad to see if

# a parsimonious model is possible. We will drop crim, zn, and black altogether owing to huge

# variances present internally amongst the variables.

multiple\_lm\_4 <- lm(medv~crim+chas+nox+rm+dis+rad+tax+ptratio+lstat+rm\*lstat+rm\*rad+lstat\*rad, data = train)

summary(multiple\_lm\_4)

residuals <- data.frame('Residuals' = multiple\_lm\_4$residuals)

res\_hist <- ggplot(residuals, aes(x=Residuals)) + geom\_histogram(color='black', fill='red') + ggtitle('Histogram of Residuals')

res\_hist

par(mfrow=c(2,2))

plot(multiple\_lm\_4)

glance(multiple\_lm\_4)

mlm4\_mse <- mean(residuals(multiple\_lm\_4)^2)

mlm4\_mse

mlm4\_rmse <- sqrt(mlm4\_mse)

mlm4\_rmse

mlm4\_rss <- sum(residuals(multiple\_lm\_4)^2)

mlm4\_rss

mlm4\_rse <- sqrt(mlm4\_rss/341)

mlm4\_rse

# As the fourth model is the better of the four models, we'll consider it as the parsimonious model and

# use it to predict the testing data set

fit\_predict <- predict(multiple\_lm\_4,test)

summary(fit\_predict)

fit\_ssl <- sum((test$medv-fit\_predict)^2)

sprintf("SSL/SSR/SSE: %f", fit\_ssl)

fit\_test\_mse <- fit\_ssl/nrow(test)

sprintf("MSE: %f", fit\_test\_mse)

fit\_test\_rmse <- sqrt(fit\_test\_mse)

sprintf("RMSE: %f", fit\_test\_mse)

# Let us create a new column to store the predicted prices/property values in the testing data set

test$pred\_price <- fit\_predict

pred\_plot <- test %>% ggplot(aes(medv,pred\_price)) + geom\_point(alpha = 0.75) + geom\_smooth(method = "loess") + stat\_smooth(aes(color = "black")) + xlab("Actual Property Value") + ylab("Predicted Property Value")

pred\_plot

The parsimonious model has a lower value for all the relevant metrics – RMSE, AIC, and BIC. It also has a higher R2 and F-statistic value.

The interaction terms were used basis the correlation plot created above which showed –

1. Rm and medv have a high positive correlation and as we have seen rm also has a lot of outliers. Taking it into an interaction term can also help us deal with heteroscedasticity.
2. Rm and lstat have high negative correlation which makes sense given that the lower the population status of the neighborhood, the smaller will be the house size and thus, the lesser number of rooms.
3. Rm and rad have a low but noticeable negative correlation. Given the previous correlation, it is likely that small houses belong to poorer sections of the population who might have a relatively difficult time accessing infrastructure and thus have lower property values.
4. Lstat and rad have a noticeably high positive correlation which runs counter to the above explanation. This means that poorer residents tend to be living closer to radial highways and other infrastructure which may contribute to things like noise etc. and thus reduce the property value owing to a lower standard of living. Including this could help capture the variance in the dataset to a higher degree.

Now, we could have also removed rad and tax variables from the model but doing that led to a negligible increase in the R2  value.

Chart, histogram

Description automatically generated

Chart, scatter chart

Description automatically generated

Chart, scatter chart

Description automatically generated

By looking at the above plot we can say that our model has done a pretty good job of predicting the property values.

Mathematically,

**Medv = -26.88 + (-0.115) \* crim + 2.73 \* chas + (-13.20) \* nox + 11.85 \* rm + (-0.74) \* dis + 2.71 \* rad + (-0.008) \* tax + (-0.625) \* ptratio + 1.83 \* lstat + (-0.327) \* (rm:lstat) + (-0.314) \* (rm:rad) + (-0.035) \* (rad:lstat)**

Before carrying out both Ridge and Lasso Regression we’d need to do some preparatory steps to separate the dependent and independent variables. We’ll be doing them below:

grid <- 10 ^ seq(6, -3, length = 10)

# Independent/Action Variables

x <- model.matrix(medv~., boston)[,-1] #-1 is to remove the Intercept column which auto-creates

# upon running the model for the first time

head(x)

#Dependent/Response Variable

y <- boston$medv

**ii.) Ridge Regression** ->

# Perform the first ridge regression with a random lambda function obtained from the grid

ridge\_mod <- glmnet(scale(x), y, alpha = 0, lambda = grid, thresh = 1e-2, standardize = TRUE)

#### P.S. - I need help in understanding the output of the below two lines

coef(ridge\_mod)

plot\_glmnet(ridge\_mod, xvar = "lambda", label = 4)

Chart

Description automatically generated

# plot\_glmnet(ridge\_mod, xvar = "lambda", label = 2) # Optional Step. Only minor changes observed by changing the value of the label argument

cv\_ridge <- cv.glmnet(scale(x), y, alpha = 0, nfolds = 10)

cv\_ridge

plot(cv\_ridge)

Chart, histogram

Description automatically generated

best\_lambda\_ridge <- cv\_ridge$lambda.1se

best\_lambda\_ridge

ridge\_mod\_final <- glmnet(scale(x), y, alpha = 0, lambda = best\_lambda\_ridge, thresh = 1e-2, standardize = TRUE)

predict(ridge\_mod\_final, type = "coefficients", s = best\_lambda\_ridge)

ridge\_pred <- predict(ridge\_mod\_final, s=best\_lambda\_ridge, newx = scale(x))

sprintf("MSE: %f", mean((ridge\_pred - y)^2))

sprintf("RMSE: %f", sqrt(mean((ridge\_pred - y)^2)))

Text

Description automatically generated

Mathematically,

**Medv = 22.53 + (-0.42) \* crim + 0.26 \* zn + (-0.379) \* indus + 0.702 \* chas + (-0.693) \* nox + 2.636 \* rm + (-0.406) \* age + (-0.966) \* dis + 0.153 \* rad + (-0.512) \* tax + (-1.476) \* ptratio + 0.718 \* black + (-2.307) \* lstat**

**iii.) Lasso Regression ->**

lasso\_mod <- glmnet(scale(x), y, alpha = 1, lambda = grid, thresh = 1e-2, standardize = TRUE)

plot\_glmnet(lasso\_mod, xvar = "lambda", label = 4)

Chart

Description automatically generated

lasso\_cv <- cv.glmnet(scale(x), y, alpha = 1, nfolds = 10)

lasso\_cv

plot(lasso\_cv)

A picture containing graphical user interface

Description automatically generated

best\_lambda\_lasso <- lasso\_cv$lambda.1se

best\_lambda\_lasso

lasso\_mod\_final <- glmnet(scale(x), y, alpha = 0, lambda = best\_lambda\_lasso,thresh = 1e-2, standardize = TRUE)

predict(lasso\_mod, type = "coefficients", s = best\_lambda\_lasso)

lasso\_pred <- predict(lasso\_mod\_final,s = best\_lambda\_lasso, newx = scale(x))

sprintf("MSE: %f", mean((lasso\_pred - y)^2))

sprintf("RMSE: %f", sqrt(mean((lasso\_pred - y)^2)))

Text

Description automatically generated

**Mathematically,**

**Medv = 22.53 + (-0.295) \* crim + 0.15 \* zn + (-0.122) \* indus + 0.489 \* chas + (-0.190) \* nox + 3.01 \* (rm) + 0.003 \* age + (-0.917) \* dis + (-0.167) \* tax + (-1.68) \* ptratio + 0.642 \* black + (-3.678) \* lstat**

1. **Summary Tables ->**
2. **Ridge & Lasso Regression** ->

|  |  |  |
| --- | --- | --- |
|  | Ridge Regression (α = 0) | Lasso Regression (α = 1) |
| min λ | 0.678 | 0.0212 |
| 1se λ (optimal lambda, **λ\***)) | 4.356725 | 0.345263 |
| MSE (**at λ\***) | 26.301543 | 22.769636 |
| RMSE (**at λ\*** ) | 5.128503 | 4.771754 |

1. **Multiple Linear Regression** ->

|  |  |  |  |
| --- | --- | --- | --- |
| Statistic | Multiple Regression | | |
| Train | Test |
| MSE | 12.02844 | 16.946752 |
| RMSE | 3.468205 | 4.116643 |
| RSS | 4258.069 | - |
| RSE | 3.533696 | - |
| R^2 | 0.8545 | - |
| Adj. R^2 | 0.8494 | - |
| F-Statistic | 166.9 | - |

1. **CV in Lasso Regression** -> The Cross-Validation (CV) approach applied here is called “Leave -One-Out- Cross-Validation” where a single observation is used to validate the training set containing the remaining variables. This approach gives an unbiased approximation of the test error but suffers from high variability owing to it being trained on a single variable.

While the model is trained (n-1) times it must be fit ‘n’ times. This can be very time-consuming if ‘n’ is large regardless of the n-folds value. The n-folds value is analogous to running a sample of size ‘n’ from the original sample ‘n-folds’ times. For large ‘n’ this can be very tedious and time-consuming as mentioned above.

1. **References** ->
2. Introduction to Statis tical Learning
3. R-bloggers -> [(i)](https://www.r-bloggers.com/2020/05/quick-tutorial-on-lasso-regression-with-example/), [(ii)](https://www.r-bloggers.com/2017/09/multicollinearity-in-r/), and [(iii)](https://www.r-bloggers.com/2021/10/model-selection-in-r-aic-vs-bic/)
4. [Cross-Validated](https://stats.stackexchange.com/questions/69549/lasso-cross-validation)