Gurjus Singh

MSDS 422 – Practical Machine Learning

October 11th, 2020

# Assignment #4 Random Forests and Gradient Boosting PART A

# Data preparation, exploration, visualization

In the Boston dataset our goal was to see the effect of nox on the median value of houses, I also wanted to see which features in general had a great effect on the median value. I first started out by loading the dataset “Boston.csv”. I first wanted to get an initial look and feel for the data set. So, I looked at the size of the data set which was (512, 14). This means there are 512 rows and 14 columns mainly features and one variable we want to predict. After getting a feel for the size of the dataset I wanted to check out the head of data frame and types for every column. I saw non-numeric row, which we do not want to use in linear regression, so I dropped this feature column. Most of the columns were mainly floats, except for two columns which contained integer values. I used **.describe()** value and .**isnull.sum()** to find out whether the initial data had any typos, or missing value. From looking at min-max category in describe table I did not see anything obviously wrong, so I decided to continue. I also did not see any columns with NA values which is what **isnull.sum()** does. This counts up all the values which evaluate to True in a column.

Graphical user interface, application

Description automatically generated

*Boxplot of Columns (Plot 1-1)*

I then decided to go to exploring through data visualizations of the data set. I first tried to boxplot as depicted in Plot 1-1. In the plot I could see many of our columns had an abundance of outliers such as our target variable **MV, lstat, dis, rooms, chas, zn and crim**. I wanted to play around more and look closely at some of the variables, so I decided to see distributions through histograms.

Chart, histogram

Description automatically generated

*Histogram Plot 1-2*

In plot 1-2, it is the median value of the houses, which we will be predicting on this using the independent variables. This looks like a less skewed distribution in comparison to doing the boxplot. I also wanted to look at features I thought were more important to predicting this dataset such as **crim**, which is the per capita crime rate per town, **ptratio** which is pupil to teacher ratio by town and **lstat** which is percent of lower status in an area.

Chart, histogram

Description automatically generatedGraphical user interface, application

Description automatically generated

*Histogram Plot 1-3 Histogram 1-4*

Chart, histogram

Description automatically generated

*Histogram 1-5*

The **lstat** variable in 1-3 seems to be skewed to the right where 10% in a given neighborhood or region are mostly of lower status. In 1-4 the histogram is right skewed as well with most of the crime happening near 0 in the areas in Boston where the data is collected. In Histogram 1-5, we see for **ptratio**, is near 20 which mean the difference in teacher to student ratio is high, this data is more skewed to the left. Lastly, I wanted to make some plots to see any initial correlation between MV, and these variables. I also wanted to see if age was in any way correlated.

Chart, scatter chart

Description automatically generatedChart, scatter chart

Description automatically generated

*Dot plot of Age variable compared to MV Plot 1-6 Figure 1-7 of highly linear ‘rooms’ variable*

In plot 1-6, I do not see any linear shape compared to age, which suggests there is no relationship in MV, and the **age** variable. I also looked at crim variable which did not seem to have a linear shape as well. During my initial findings I also saw that rooms and **lstat** variables seemed to have a linear shape which means they were highly positive or negatively correlated as seen in figure 1-7.

After examining linear shaped data and nonlinear data, I then wanted to scale the features between 0 and 1 as it helps speed up the learning process. In order to make the data more linear I had to use the lambda function which helps to map the function to all the data in one line of code. I did not want any zeros in my data, so I added .01 to every data point. I then used “boxcox” which is great for making sure my data is more normalized and more linear to use linear models on it. I had to hand pick the columns, so I chose the features shown below because I felt they were nonlinear by looking at their scatterplots.



Chart

Description automatically generatedBefore modeling using the linear regression method and tree methods, I first wanted to create a correlation heatmap to get an overall picture of which features in the data set might be more useful to predict median value price of a neighborhood of houses. This heatmap showed exactly what might initial findings confirmed in that lstat was negatively correlated, and rooms was highly positively correlated as shown in figure 1-8 below. It also shows which columns we would like to drop that have the multicollinear property.

*Figure 1-8 Heatmap*

# Review research design and modeling methods

After we choose, the features I am going to use 3 modeling methods which are mainly tree learning algorithms but have some variations and one regression algorithm. The three types are Regular Linear Regression, Random Forest, and Extra Randomized Trees. The differences between the three is that Linear Regression does not add regularization term which is important in most cases to prevent overfitting [1]. Linear Regression assumes linearity so that is why it is important to transform your features. Linear Regression is also important because to use when doing predictions on numerical response values [1]. Random Forest is another special algorithm which involves decision trees [1]. Decision trees are trees that use feature cutoffs to decide on the final value of the prediction [1]. With Random Forests it involves an ensembled method which involves different trees which all aggregate together to form a predicted result [1]. Each tree uses a method to collect training data which is known as “Bagging”. “Bagging” involves sampling which collects n data points from the training set to train on. Thirdly another Tree learning algorithm used is Extra Randomized Trees [1]. This is similar to Random Forests in a sense because they both pick a random number of features to train on, but this also involves random thresholds to use as a cutoff for each feature chosen [1]. I think these algorithms are important because Linear Regression as mentioned before is mostly used to predict numerical values. Trees are kind of different in that one can see how the algorithm works and it is not a “black-box” like Linear Regression is. Trees do not assume linearity like Regression does and does not need transformation and scaling [1].

Before we start training the algorithm, I had to save the target variable in its own data frame to separate it from the features. In order to use the regression methods, it is important to split up the data into train data, and test data. I did the 80-20 split on the data. Our goal was to make sure that the data trained and predicted well on the test set.

# Review Results, and Evaluate Model

# After implementing the model, I took at the results at both the Training Set and Test Set. I also wanted to test the models by dropping features with insignificant P-values as shown in 1-9. I first looked at the results from when no columns were dropped. It looks like the data was overfitting by looking at the Training Data Results from 1-9. What I noticed for was Extra Trees was the highest with an R^2 of 0.97. I then looked at the Results of the Test Set in 1-10 and I saw Extra Trees was still in the lead with an R^2 of 0.8848. I then dropping features with high P-values mainly age, zn, and crim first seen in 1-12. What I noticed is that Extra Trees was still in the lead with an R^2 performing slightly lower, but not that lower compared to Linear Regression with a R^2 of 0.8846. I then dropped other columns to only leave P-Values of 0 shown in 1-13. What I noticed is Extra Trees slightly went down, but still came out on top with an R^2 of 0.878.

# A picture containing text Description automatically generated

# *Results without columns dropped on Training Set 1-9*

# A close up of text on a white background Description automatically generated

# 

# *Test Data Results W/O Columns Dropped 1-10 P Value Table 1-11*

# *A picture containing graphical user interface Description automatically generated*

# *Dropped Features with High P-Values 1-12 Left P-Values of 0 in 1-13.*

# After looking at the results I wanted to see the feature importance of the Extra Trees and Linear Regression for when they performed the best, which was when no features were dropped. I looked particularly at the Train Data/Test Data feature coefficients. For Linear Regression I got the following Train /Test Data formula:

# 45.6 + 0.49\*crim + 0.30\*zn -2.93\*indus + 2.06\*chas – 8.38\*nox + 12.42\*rooms + 2.42\*age -14.72\*dis + 3.59\*rad -5.96\*tax – 4.64\*ptratio – 29.74\*lstat

# Train Data Formula 1-14

# Text Description automatically generated

# *Train Data Coefficients 1-15 Train Data Feature Importance 1-16*

# 38.8 + 11.21\*crim + 1.33\*zn -1.89\*indus + 3.71\*chas – 11.04\*nox + 23.94\*rooms - 2.34\*age -18.44\*dis + 3.41\*rad -9.04\*tax – 7.68\*ptratio - 21.9\*lstat

# *Test Data Formula 1-17*

# Text Description automatically generated

# *Coefficients and Intercept for Linear Regression Test Data 1-18 Feature importance Test Data for Extra Trees 1-19*

# The Linear Regression formula for Training data tells us that lstat has the biggest negative influence on median house values and tells us the target value falls by -29.74 with every increase in lstat. The other big negative weight I see is with dis which shows that with every increase in this there is a -14.72 decrease in the target variable. I also see rooms which seems to have the biggest positive weight. What this says is that with every increase in a room the median value increases by 12.42.

# The Linear Regression formula for test data tells us that lstat has the biggest negative influence on median house values and tells us the target value falls by -21.9 with every increase in lstat. The other big negative weight I see is with dis which shows that with every increase in this there is a -18.44 decrease in the target variable. I also see rooms which seems to have the biggest positive weight. What this says is that with every increase in a room the median value increases by 23.94.

# For interpreting the extra trees matrix, you have to see the largest number to find the most important feature. It seems from the matrix that the most important features are rooms and lstats. Rooms has an importance of 28 percent while lstat has one of 43.6 percent. The least important ones are zn with importance of 0.7 percent and chas with 0.7 percent.

# Implementation and Programming

# Before implementation it is important to import packages such as sklearn, pandas, matplotlib and stats. After initial importing we start our data prep phase, which will involve using .describe() method from pandas package. This is important to get basic statistics of each column of Pandas Dataframe. It is also important to look at the .dtypes to look at variable types for each column to get a sense of the data. To see the data without doing anything on it one can use .head() method on the dataframe to get initial rows and columns of the data. Using isnull.sum() on the dataset allows one to find null values for imputation purposes. After seeing the initial data, we can visualize each feature by using matplotlib. This is useful to use different visualizations such as boxplot() method to get a sense of how skewed the data is. One can also use matplotlib.plot() function to get scatterplots and line plots. .Hist() function is another good visual to plot a histogram. When one wants to see relationships of features to the response variable used to predict on they can use a for loop, and make subplots using the subplot() and scatter() functions. It is also important to use boxcox() on select features that do not have linear relationships with the target variable. This is why the stats package is used. It is also important to scale features to compare features. In order to scale features the following code can be used: boston\_df=boston\_df.transform(lambda x: (x - x.min()) / (x.max() - x.min())). Drop function is also used to drop unnecessary features before training such as variables that a multicollinear when seeing the heatmap depicted in 1-8 . When its time to start modeling it is important to save your response variable using this format: y = “response column”. Using train\_test.split() one can split data into training set and test set. From here one can instantiate models such as LinearRegression(), RandomForestRegressor(), and ExtraTreeRegressor(). Once instantiated they can call attributes such as .coef for coefficients for Linear Regression or functions such as .fit() or .score() for R^2.

# Text Description automatically generatedText Description automatically generated

*Packages to Import 1-20 Scatterplot matrix for finding relationships. 1-21*

# Exposition, problem description and management recommendations

# After examining the Boston dataset, it is clear that Nox does not have a significant influence on median value of a house. Two ways one can see this is by looking at the p value for Nox which is at 0.20 so it is nonsignificant since it above 0.05. Another way is to look at Extra Trees feature importance which marks Nox as 3.4% important. Therefore, Nox is not a good indicator of median value. For this Boston Dataset, the best model was Extra Trees algorithm. It was the best do to it’s R^2 score which stayed relatively the same after dropping columns from the dataset. I also tried to use OOB score to see if Random Forest Regressor performed well, but it actually was overfitting in the training dataset as the OOB Score got significantly worse on the test set. For example, with no columns dropped the OOB score was around 0.8511 for fitting the training data, but for the test data the score was only 0.769. Therefore, it showed how much worse it was doing using the test data. In conclusion, I recommend Extra Trees to management as it did well on the R^2 metric in all three trials which involved dropped features while the linear regression did significantly worst, and Random Trees did not perform well on OOB score.

References

# [1] Géron, A. *Hands-On Machine Learning with Scikit-Learn & TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems.* 2d Edition. Sebastopol, Calif.: O'Reilly. [ISBN 9781492032649], 2019.