Combined Cycle Power

Hitesh Gupta

08/06/2019

Objective

The objective of this data science project is to build a machine learning (ML) algorithm that will predict Energy Output of a power plant based on factors such as Temperature, Ambient Pressure, Relative Humidity and Exhaust Vacuum.

Evaluation Criteria

To evalaute and compare multiple ML approaches, we will use the principal of Least Squared Error (LSE). LSE can be defined as

$$LSE = \frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2$$

where $\hat{y}_i$= output predicted by selected approach and y_i= Actual output of the observation

Data Set

For this project we are using a free data set provided by University of Irvine Machine Learning Repository. More details on the dataset can be seen in this link

The dataset has 5 types of data (columns) for each energy output reading (row)

Input Variables

- **T**: Temperature of surrounding (range 1.81°C and 37.11°C)
- **AP**: Ambient Pressure (range 992.89-1033.30 milibar)
- **RH**: Relative Humidity in air (range 25.56% to 100.16%)
- **V**: Pressure inside Exhaust Vacuum (range 25.36-81.56 cm Hg)

Output Variables

- **PE**: Hourly Energy Output (range 420.26-495.76 MW)

Note: Original data has 5 shuffled copies of data which was used by authors for other baseline studies and other tests. Only the first copy of the data was used in this study

Methods & Analysis

Approach

Machine learning algorithms can be classified into 2 types

1. **Supervised Algorithms** Supervised algorithms are used in cases where we already have outcome or output values for our sample data and the machine then tries to learn from a sub-set of this data (training data set) to predict the outcome of another mutually exclusive subset (test data set) or entirely new data set. An example of this could be - Given data for 1 million patients tested for a given medical condition, predicting the probability of a patients testing positive for that condition
2. **Un-supervised Algorithms** These type of algorithms do not have any prior data or prior outputs to learn from. These algorithms try to find a pattern, structure, similarity etc within given data set and then try to classify the given data into different clusters or categories
3. **Classification Algorithms** Classification algorithms are useful when the predicted value is binary, categorical or ordinal in nature. Some examples would be QDA, LDA, Classification Tree etc. For example predicting whether or not a user has disease or not?
4. **Regression & Hybrid Algorithms** Regression (Linear and Logit regression) along with other algos (called hybrid algorithms) such as KNN, Random Forest, Regression Trees etc can work both for continous and categorical data

For this project, we will be using supervised regression algorithms and supervised hybrid algorithms

Training and Test Sets

We have randomly selected 50% of our data as the training set and stored in *train_set* variable. The rest of the data will used for validation and is stored in the *test_set* variable.

```
set.seed(1)
test_index<-createDataPartition(dataset$PE,times=1,p=0.5,list=FALSE)
test_set<-dataset[test_index,]
train_set<-dataset[-test_index,]
```

We also select 4% of our training data set to visually showcase how our algorithms are working and how close they are to actual values

```
set.seed(1)
mini_index<-createDataPartition(train_set$PE,times=1,p=0.04,list=FALSE)
mini_set<-dataset[mini_index,]
```

Data Wrangling

It's a set of techniques to process the raw data in usable or tidy format which can then easily be used for further processing. We take the following steps to check the data wrangling measures required on our data (if any)

Converting to tidy format - data frame

The class of data is already "data frame" and data is alredy present in tidy format

Checking for empty (NA) values

Each column in the data was checked with **is.na()** and no empty values were found in each of the columns

Checking for Null

Each column in the data was checked with **is.null()** and no empty values were found in each of the columns

Checking for NaN

Each column in the data was checked with **is.nan()** and no empty values were found in each of the columns

Structure of the data table

The structure of the data table was analyzed using **str()** and the data types used for each column were found as expected. No factor variables or string found

```
str(dataset)
```

```
## 'data.frame':    9568 obs. of  5 variables:
##  $ AT: num  14.96 25.18 5.11 20.86 10.82 ...
##  $ V : num  41.8 63 39.4 57.3 37.5 ...
##  $ AP: num  1024 1020 1012 1010 1009 ...
##  $ RH: num  73.2 59.1 92.1 76.6 96.6 ...
##  $ PE: num  463 444 489 446 474 ...
```

Data Exploration & Insights

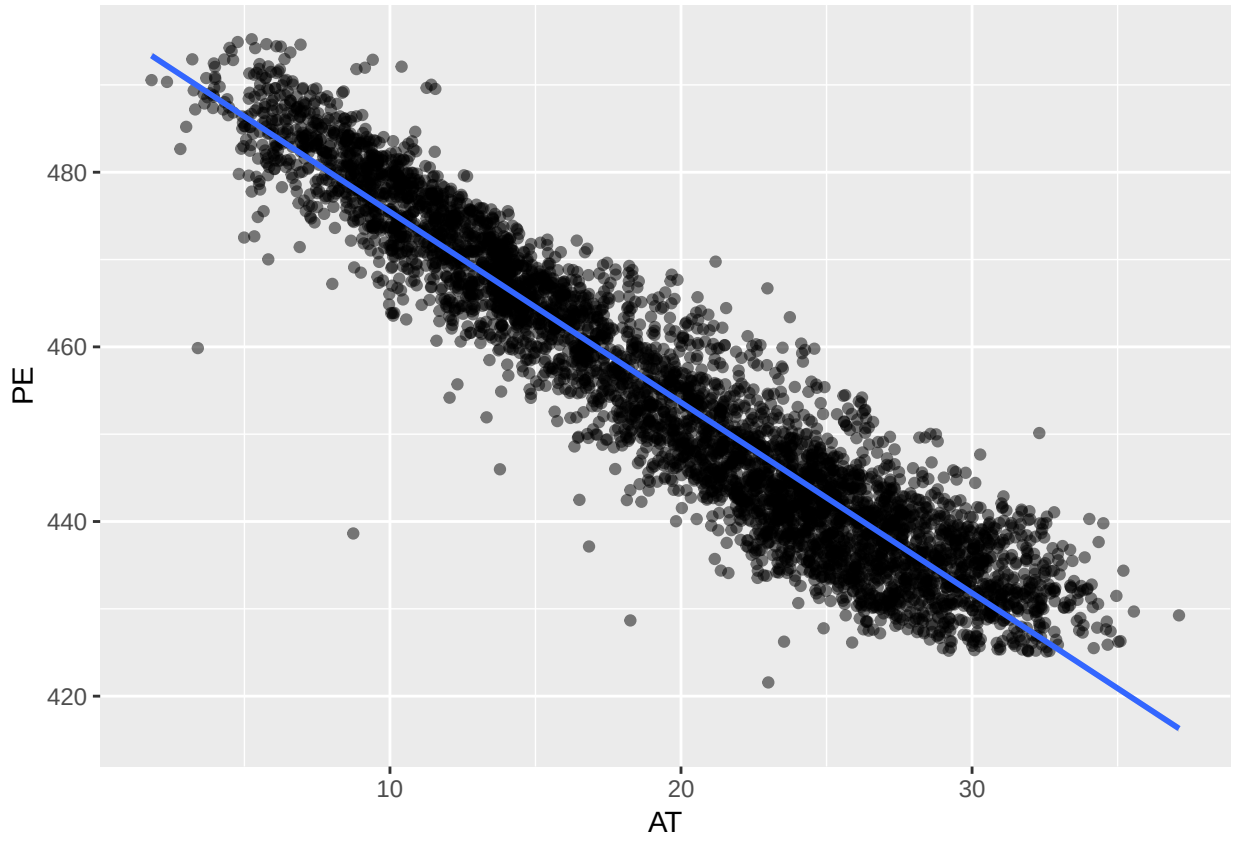
Exploring the training data set (train_set) gives us some key insights.

- There are 9568 entries in the dataset

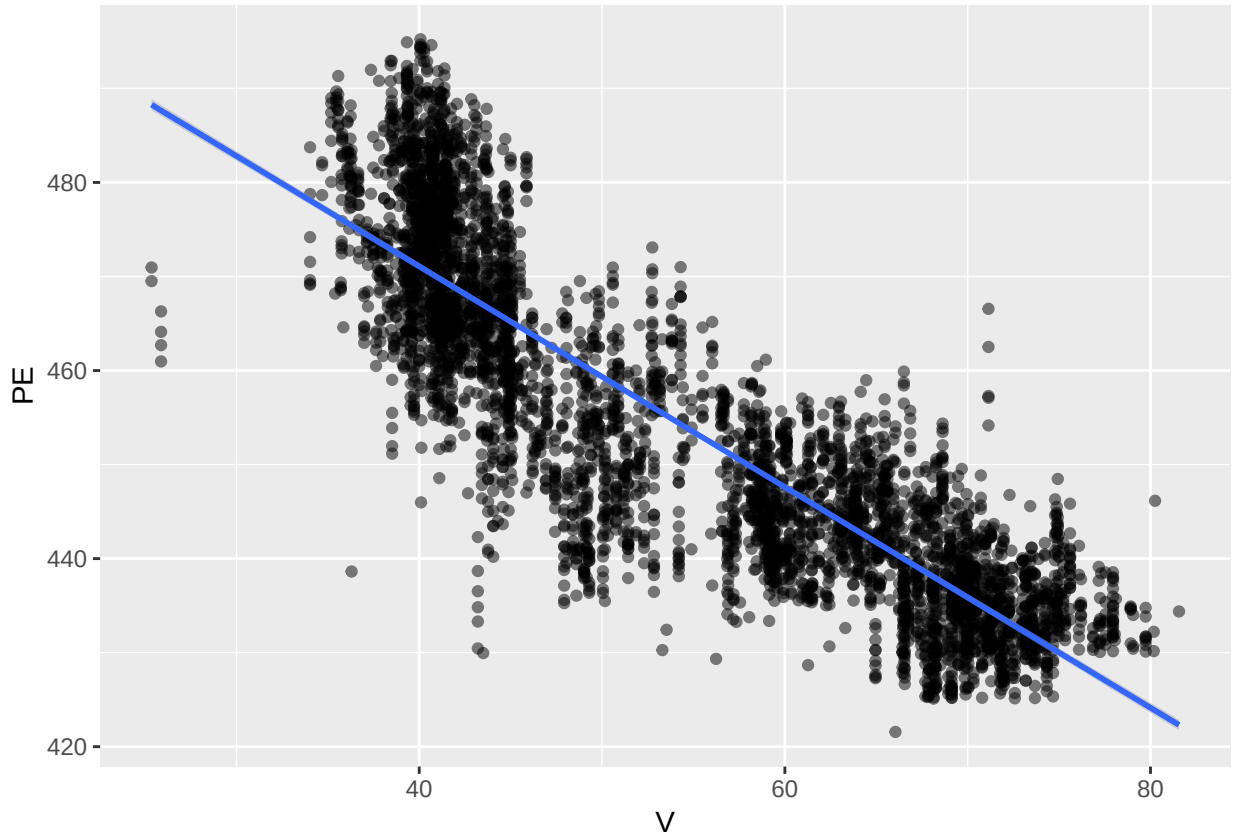
Checking Relationship for input variables**

To check the relationship of PE against all input variables, we see the scatter plot for all 4 input variables

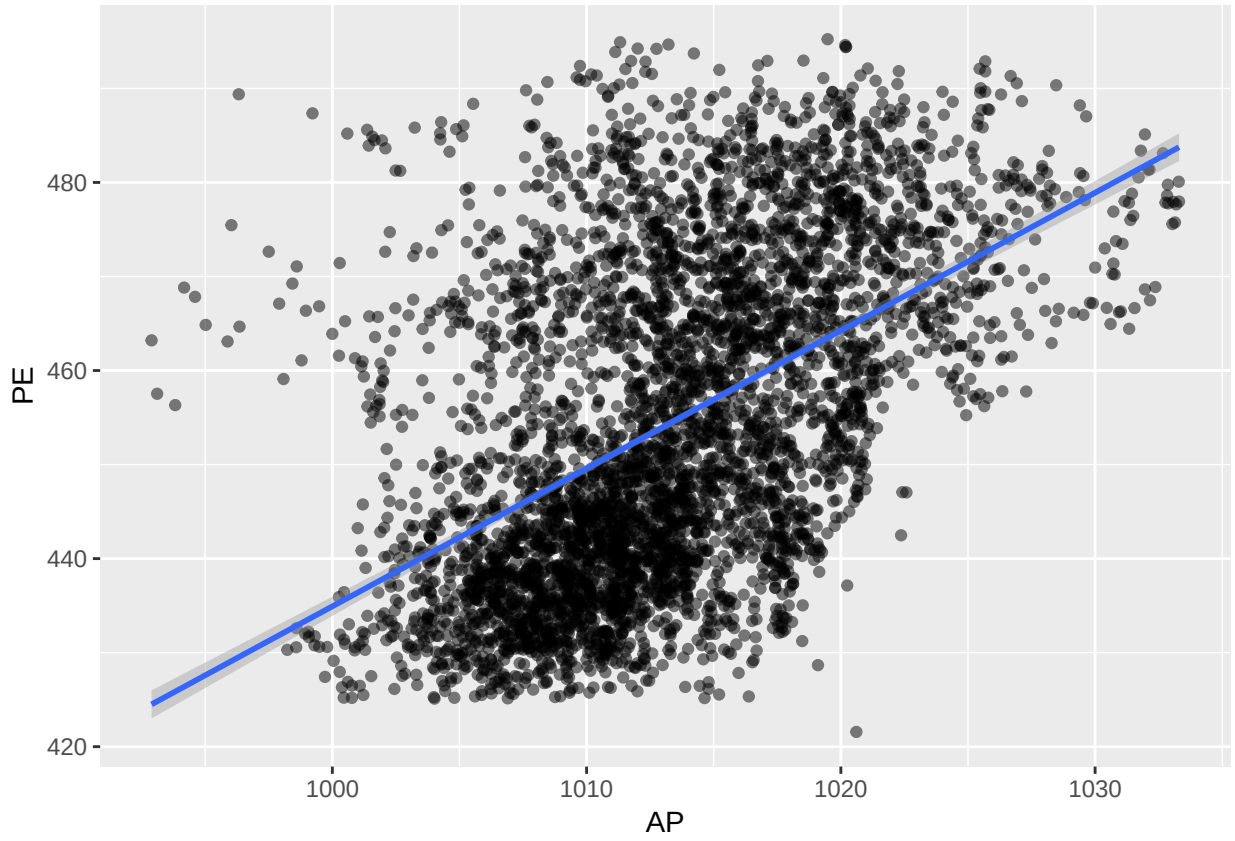
```
train_set %>%
  ggplot(aes(AT, PE)) +
  geom_point(alpha = 0.5) +
  geom_smooth(method = "lm")
```



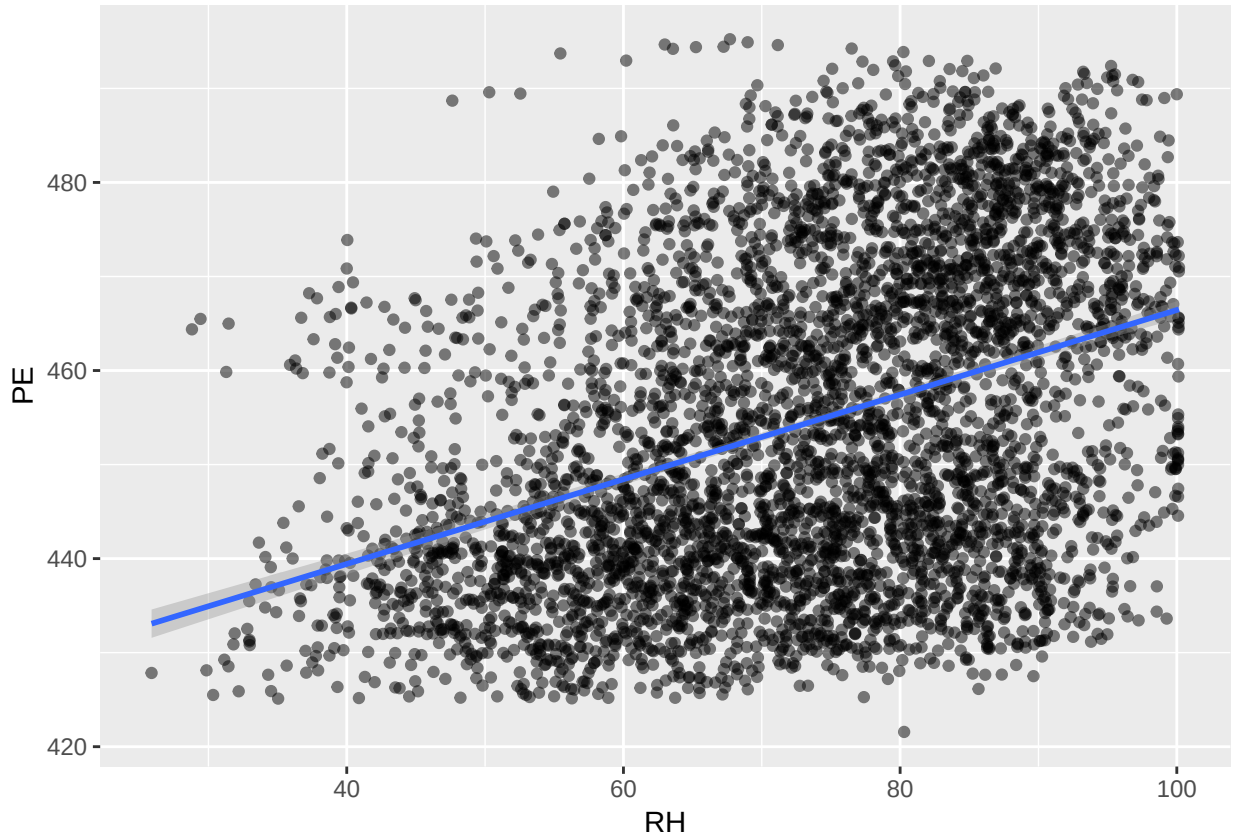

```
train_set %>%
  ggplot(aes(V, PE)) +
  geom_point(alpha = 0.5) +
  geom_smooth(method = "lm")
```




```
train_set %>%
  ggplot(aes(AP, PE)) +
  geom_point(alpha = 0.5) +
  geom_smooth(method = "lm")
```



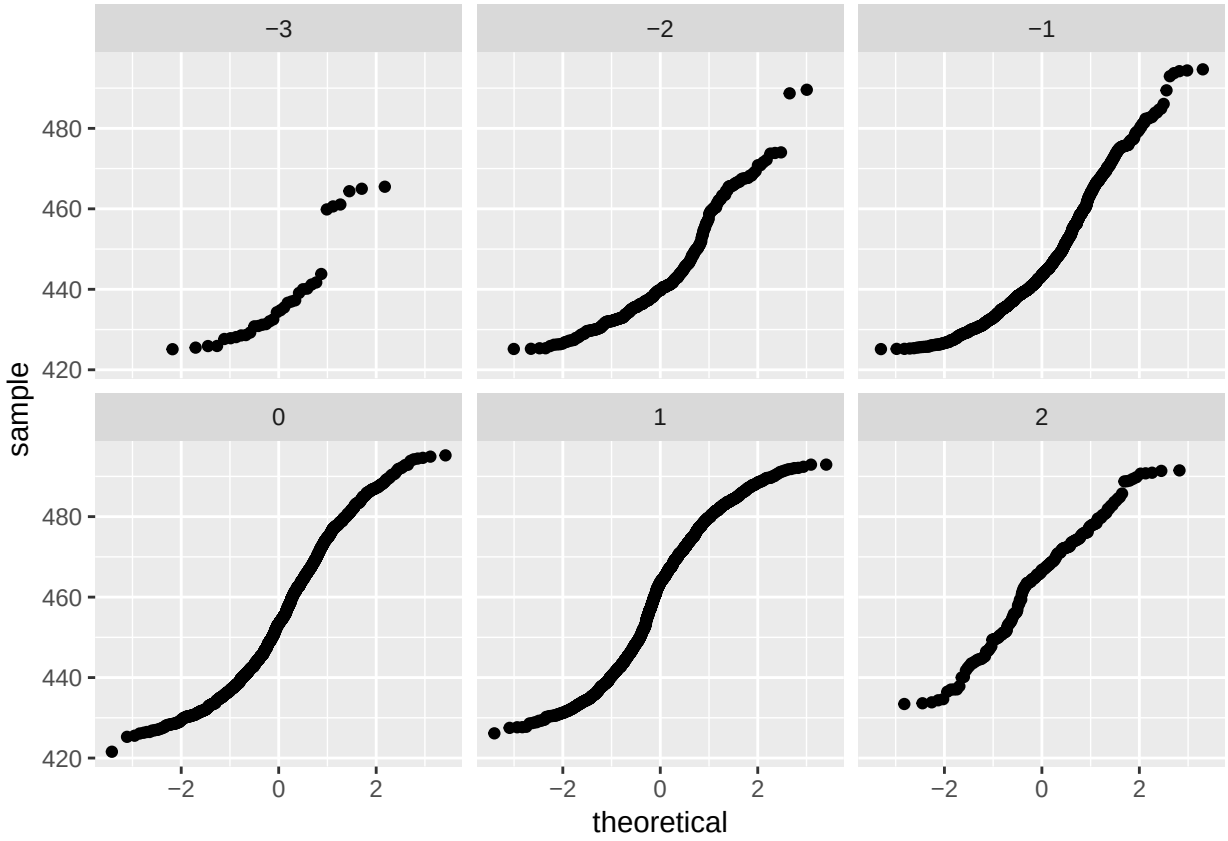

```
train_set %>%
  ggplot(aes(RH, PE)) +
  geom_point(alpha = 0.5) +
  geom_smooth(method = "lm")
```



From all graphs above we can infer that AT & V have strong correlation with PE, while AP is somewhat weakly correlated. One cannot infer much for RH since no pattern emerges from it's scatter plot.

To dive deeper for RH, we check if the pair (RH,PE) is bivariate normal or not

```
train_set %>%
  mutate(z_RH = round((RH - mean(RH)) / sd(RH))) %>%
  filter(z_RH %in% -5:5) %>%
  ggplot() +
  stat_qq(aes(sample = PE)) +
  facet_wrap( ~ z_RH)
```



By looking at graphs above it is clear that our data can be approximated as bivariate normal and best prediction of PE with a given value of RH will be given by a regression line

Checking confounding between input variables

In all the input variables, there could be a relationship between all environment related variables - temperature(AT), pressure(AP) and Humidity (RH). Vaccum pressure is a human controllable factor and is not assumed to have any causal effect on any other variable. To find the correlation between these 3 variables we use the following code

```
train_set %>%
  summarize(cor(AT, AP), cor(AP, RH), cor(AT,RH))
```

```
##   cor(AT, AP) cor(AP, RH) cor(AT, RH)
## 1  -0.4991258  0.09002693  -0.5383619
```

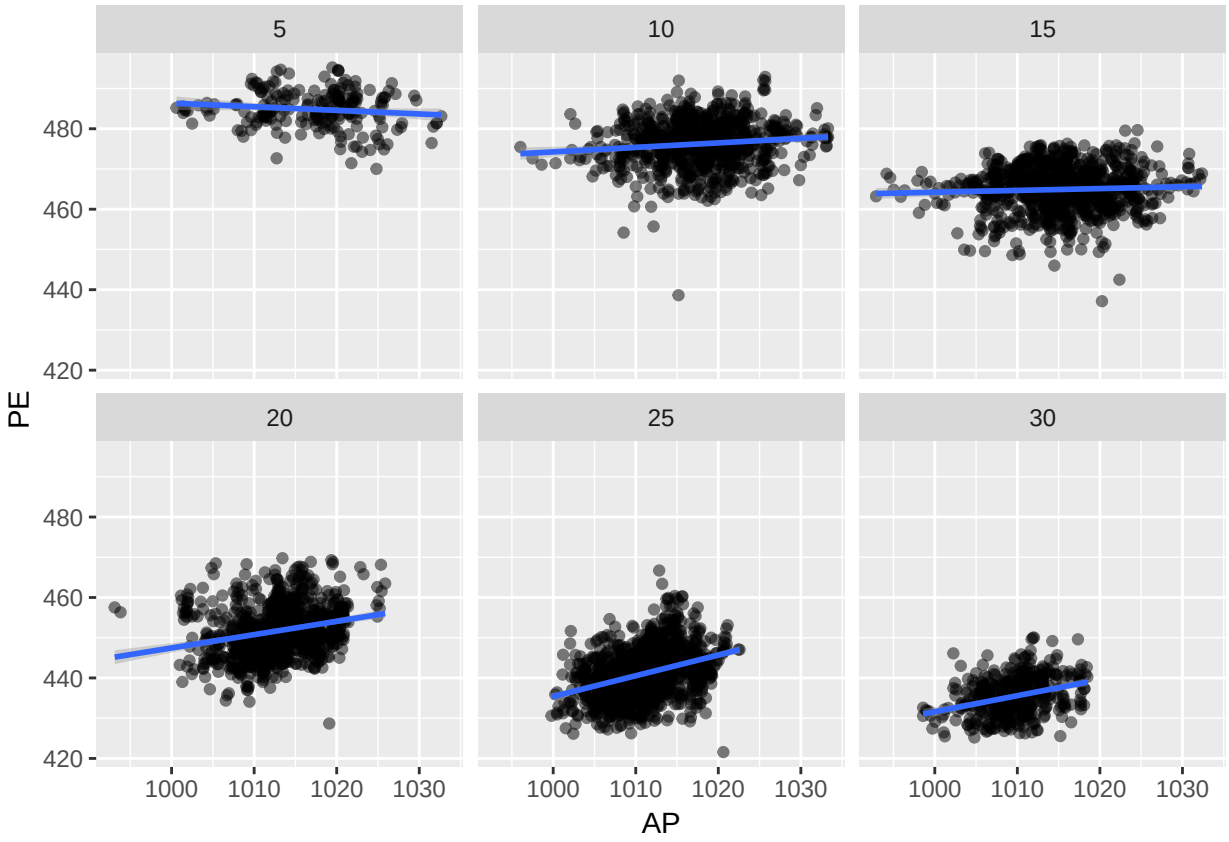
We see that AT is reasonably correlated with AT and RH, while correlation between RH and AP is weak.

Checking causality of AP and RH on PE

To check the causality of AP, we plot stratify the values of AT and check the slopes of AP vs PE

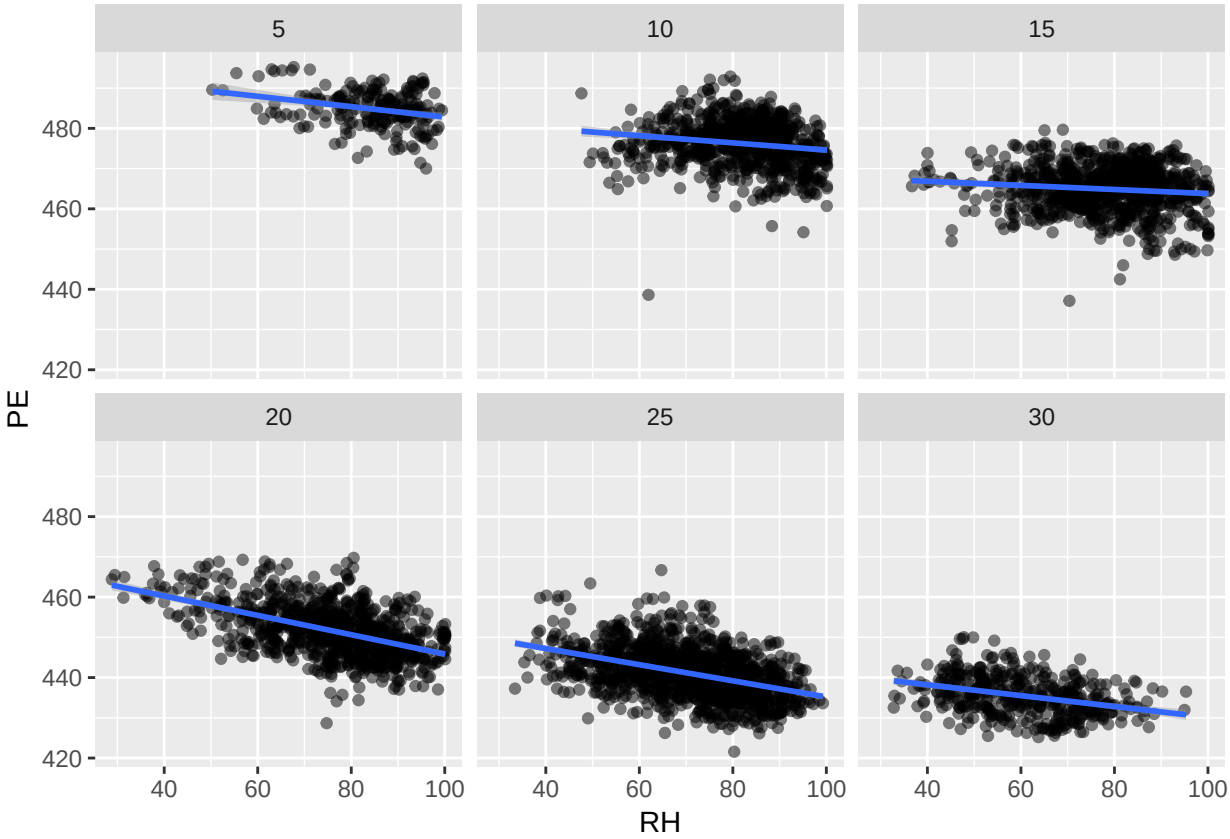
```
train_set %>% mutate(AT_strata = round_any(AT, 5)) %>%
                        filter(AT >= 5 & AT <=30)%>%
  ggplot(aes(AP, PE)) +
```

```
  geom_point(alpha = 0.5) +
  geom_smooth(method = "lm") +
  facet_wrap( ~ AT_strata)
```



To check the causality of RH, we plot stratify the values of AT and check the slopes of RH vs PE

```
train_set %>% mutate(AT_strata = round_any(AT, 5)) %>%
                      filter(AT >= 5 & AT <=30)%>%
  ggplot(aes(RH, PE)) +
  geom_point(alpha = 0.5) +
  geom_smooth(method = "lm") +
  facet_wrap( ~ AT_strata)
```



Looking at the graphs above, one can say both RH and AP do have effect on PE (keeping AT constant), but the effect has been introduced. Also RH and PE are negatively correlated (opposite) of what we saw earlier which indicates a very big positive coefficient for AT

Thus one can safely assume that this model can be modelled by a multivariate linear regression model

$$Y_i = \beta_0 + \beta_i x_i + \varepsilon_i,\ i = 1, \ldots, N.$$

Modelling Approach

Creating a reusable LSE function

As the heading suggests, we will create a reusable function to compute LSE (Least Square Error) values for our various models

```
LSE <- function(actual, predicted){
  mean((actual - predicted)^2)
}
```

Model 1: Using Average of Training data set

The most basic model, which we can use as a start is to assume that all predicted output values Y is equal to mean rating of training data set.

```r
y_hat <- mean(train_set$PE)
LSE_average<- LSE(test_set$PE, y_hat)
LSE_table <- data_frame(method = "Random Guess",
                        LSE = LSE_average)
paste("LSE - Average =",LSE_average,sep = " ")
```

```
## [1] "LSE - Average = 288.623249124123"
```

Model 2: Multivariate Linear Regression

Lets define a multivariate model

$$Y_{PE} = \beta_0 + \beta_{AP}x_{AP} + \beta_{AT}x_{AT} + \beta_{RH}x_{RH} + \beta_V x_V + \varepsilon$$

The model can be fitted using following code

```r
fit <- lm(PE ~ ., data = train_set)
y_hat <- predict(fit, test_set)
LSE_lm <- LSE(test_set$PE,y_hat)
LSE_table <- bind_rows(LSE_table, data_frame(method="Linear Regression", LSE = LSE_lm))
paste("LSE - Linear Regression =",LSE_lm, sep = " ")
```

```
## [1] "LSE - Linear Regression = 21.5333935971823"
```

The summary statistics for this model are shown below

```r
summary(fit)
```

```
##
## Call:
## lm(formula = PE ~ ., data = train_set)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -43.519  -3.119  -0.109   3.158  17.623
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) 461.901455  13.474644  34.279  < 2e-16 ***
## AT           -1.977318   0.021074 -93.826  < 2e-16 ***
## V            -0.242419   0.010030 -24.170  < 2e-16 ***
## AP            0.055004   0.013075   4.207 2.64e-05 ***
## RH           -0.153463   0.005737 -26.751  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.479 on 4778 degrees of freedom
## Multiple R-squared:  0.9318, Adjusted R-squared:  0.9318
## F-statistic: 1.633e+04 on 4 and 4778 DF,  p-value: < 2.2e-16
```

Model 3: K Nearest Neighbour Model

Linear regressions are rigid in nature and hence wee can try and make our estimation better by using **KNN algorithm**

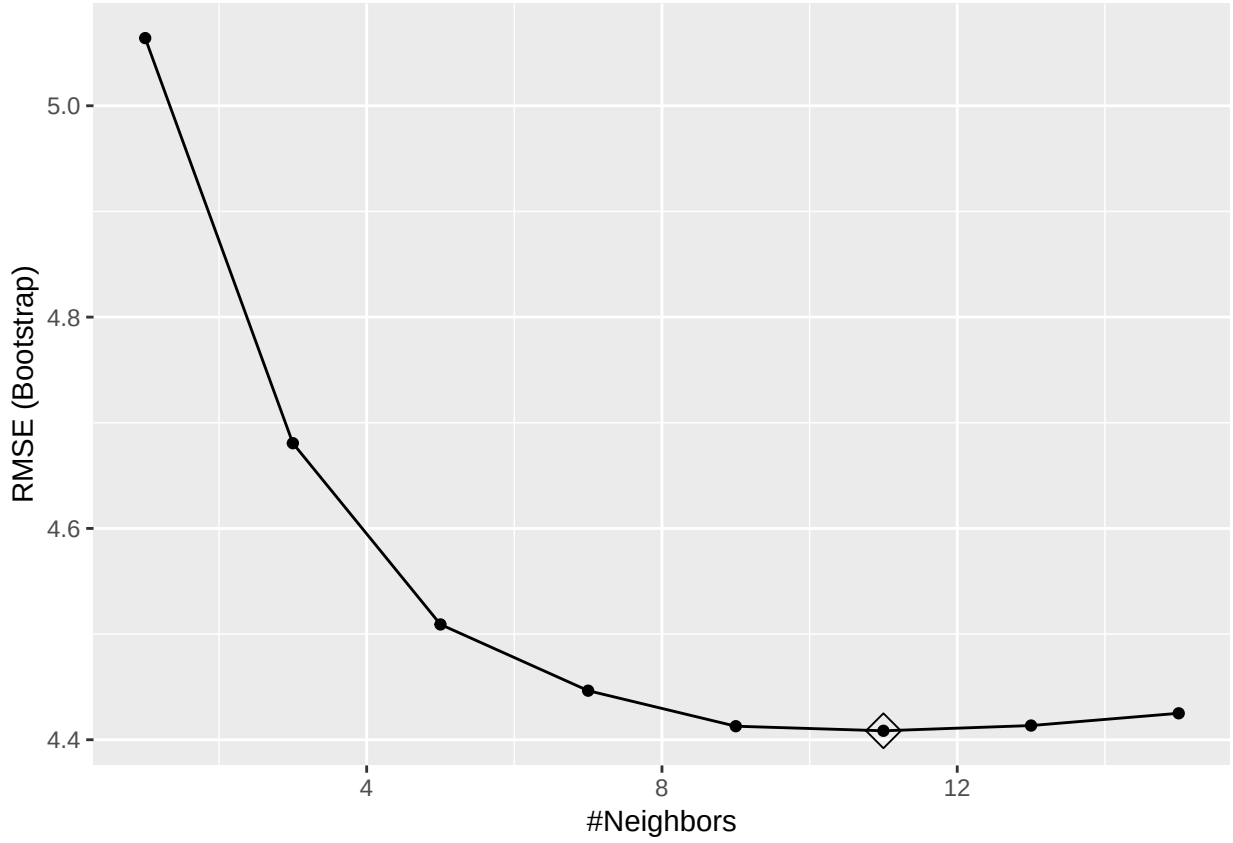
The model can be fitted using following code

```
knn_fit<-train(PE ~ .,method="knn", data=train_set,
               tuneGrid = data.frame(k = seq(1, 15, 2)))
y_hat_knn <- predict(knn_fit, test_set, type="raw")
LSE_knn <- LSE(test_set$PE,y_hat_knn)
LSE_table <- bind_rows(LSE_table, data_frame(method="KNN",
                                             LSE = LSE_knn))
paste("LSE - KNN =",LSE_knn, sep = " ")
```

```
## [1] "LSE - KNN = 18.3497601667681"
```

The best value of K is found using the code below

```
ggplot(knn_fit, highlight = TRUE)
```




```
knn_fit$bestTune
```

```
##    k
## 6 11
```

If we assume our model to be univariate in nature with only AT as input parameter, we can plot the outcome of our prediction algorithm on part of our data set (mini_set) for variable AT and visually see how it pans out

```
mini_fit<-train(PE ~ .,method="knn", data=mini_set,
                tuneGrid = data.frame(k = seq(1, 15, 2)))
mini_set %>%
  mutate(y_hat = predict(mini_fit)) %>%
  ggplot() +
  geom_point(aes(AT, PE)) +
  geom_step(aes(AT, y_hat), col="red")
```

Model 4: Regression Trees Model

Regression trees help us to estimate conditional probabilites $f(x) = \mathrm{E}(Y|X = x)$ where Y is value of estimated value of PE based on given value of x (AT, AP, V, RH). The idea of regression tree is to build a decision tree by partitioning the predictor space in J non overlapping Regions $R_1, R_2, \ldots, R_J$ and recursively keep on partitioning to minimize the RSS (Residual Sum of Squares) for given partitions

For complete reference on Regression Trees, please refer this wiki link in footnote [1]

In our data visualization excercise done earlier, when we stratified AP and RH keeping AT constant, we did not observe any clear classification pattern. Regression Tree patterns are helpful if data can be grouped into clusters around a certain value of input variable. So our hypothesis is that this model would not improve our LSE scores considerably. Let's test this

[1] https://en.wikipedia.org/wiki/Decision_tree_learning

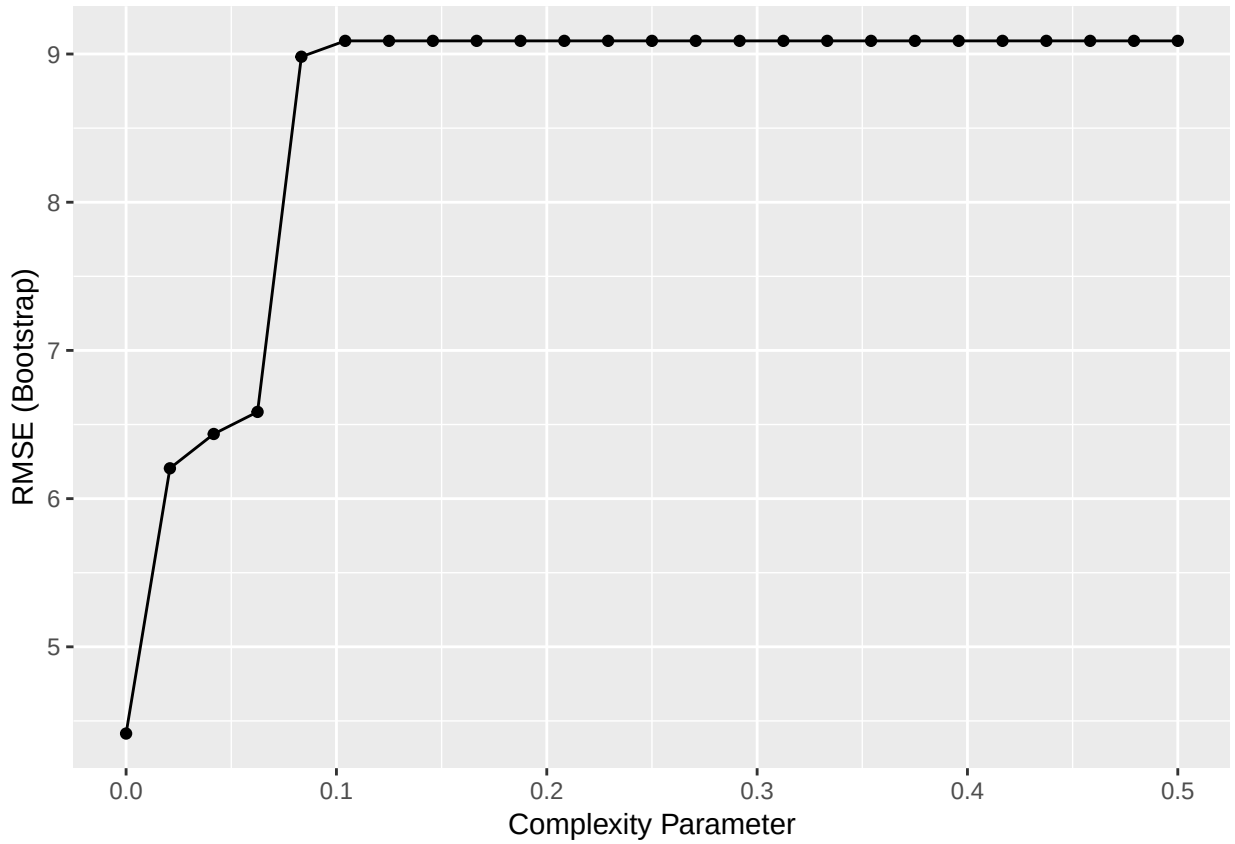
```
rpart_fit<-train(PE ~ .,method="rpart", data=train_set,
                 tuneGrid = data.frame(cp = seq(0, 0.5, len = 25)))
y_hat_rpart <- predict(rpart_fit, test_set)
LSE_rpart <- LSE(test_set$PE,y_hat_rpart)
paste("LSE - Rpart =",LSE_rpart, sep = " ")
```

```
## [1] "LSE - Rpart = 18.0431545733846"
```

```
LSE_table <- bind_rows(LSE_table, data_frame(method="Regression Tree",
                                             LSE = LSE_rpart))
```

The best value of CP (Complexity Paramter) is found using the code below

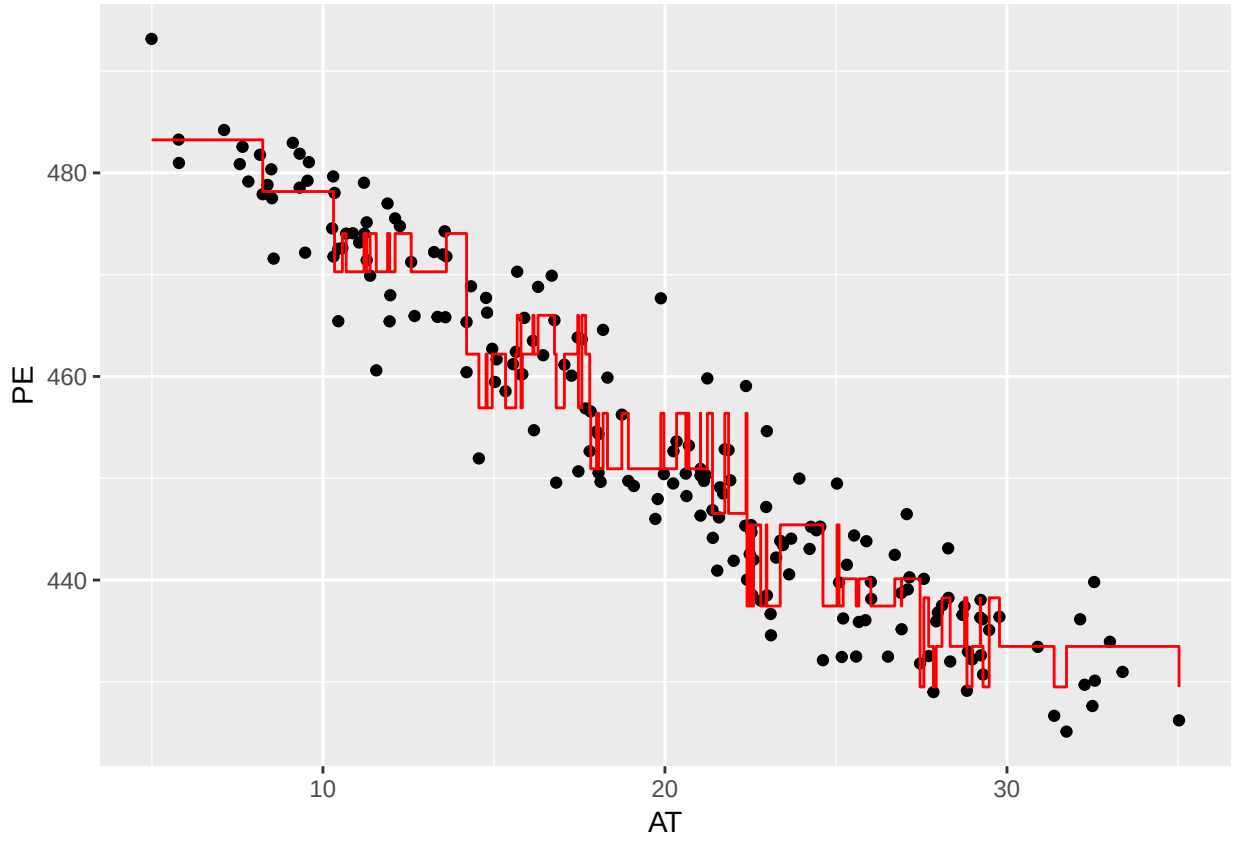
```
ggplot(rpart_fit)
```



If we assume our model to be univariate in nature with only AT as input parameter, we can plot the outcome of our prediction algorithm on part of our data set (mini_set) for variable AT and visually see how it pans out

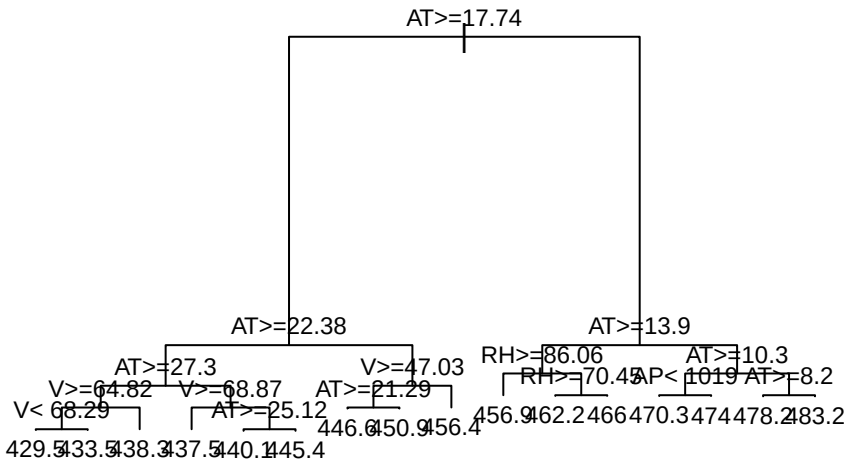
```
mini_fit<-train(PE ~ .,method="rpart", data=mini_set,
                tuneGrid = data.frame(cp = seq(0, 0.5, len = 25)))
mini_set %>%
  mutate(y_hat = predict(mini_fit)) %>%
  ggplot() +
```

```
  geom_point(aes(AT, PE)) +
  geom_step(aes(AT, y_hat), col="red")
```



The classification tree for this univariate model would look like below

```
plot(mini_fit$finalModel, margin = 0.1)
text(mini_fit$finalModel, cex = 0.75)
```



Model 5: Random Forest Algorithm

Since our regression trees did not work well, it would be a good idea to try Random Forest Algoritms. These algorithms are an extension to our classification trees, but they improve performance by averaging multiple decision trees. The idea is to create many prediction variables using regression trees - which are essentially step function and then average all of them to create smooth variation of the curves

For complete reference on Random Forests, please refer this wiki link in footnote [2]

The Random Forest can be modeled using below algorithm

```
rf_fit<-train(PE ~ .,method="Rborist", data=train_set)
y_hat_rf <- predict(rf_fit, test_set)
LSE_rf <- LSE(test_set$PE,y_hat_rf)
LSE_table <- bind_rows(LSE_table, data_frame(method="Random Forest", LSE = LSE_rf))
paste("LSE - Random Forest =",LSE_rf, sep = " ")
```

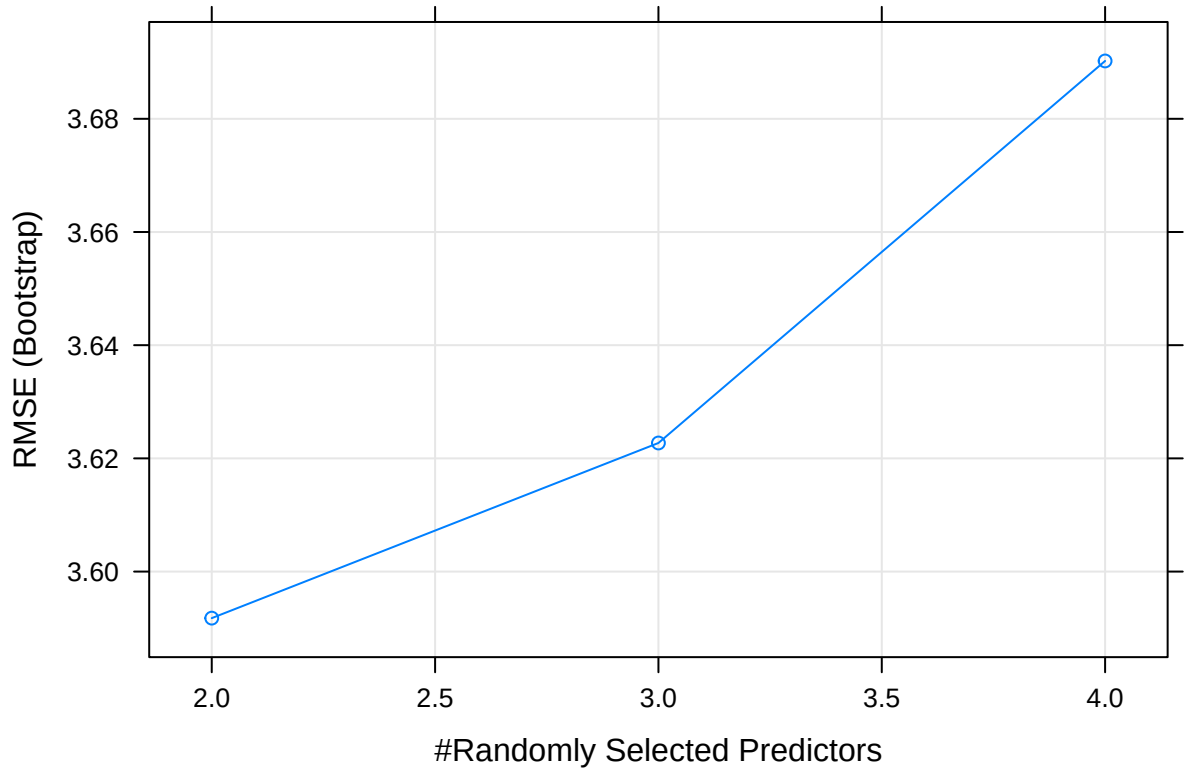
```
## [1] "LSE - Random Forest = 12.7882005210653"
```

We see that there is a huge improvement in the LSE values because of random forest algorithm

We can also check the error rate as a function of number of trees by using the code below

[2] https://en.wikipedia.org/wiki/Random_forest

```
plot(rf_fit)
```



If we assume our model to be univariate in nature with only AT as input parameter, we can plot the outcome of our prediction algorithm on part of our data set (mini_set) for variable AT and visually see how it pans out

```
mini_fit<-train(PE ~ .,method="Rborist", data=mini_set)
mini_set %>%
  mutate(y_hat = predict(mini_fit)) %>%
  ggplot() +
  geom_point(aes(AT, PE)) +
  geom_step(aes(AT, y_hat), col="red")
```

Results

The results of our various models can be seen in the table below

```
LSE_table
```

```
## # A tibble: 5 x 2
##   method              LSE
##   <chr>             <dbl>
## 1 Random Guess      289.
## 2 Linear Regression  21.5
## 3 KNN                18.3
## 4 Regression Tree    18.0
## 5 Random Forest      12.8
```

Conclusion

In this paper, we tried to predict the Electric Power output of a Combined Cycle Power Plant using UCI ML repository data set

1. We first analyzed the data and found no significant issues related to empty or invalid values in data set
2. We then did some basic inferencing and data visualization techniques to make the following inferences

i) Two of input variables (AT, V) are heavily and negatively correlated with predictor variable PE
ii) AP variable is weakly correlated with PE
iii) RH variable is weekly correlated with big variance, but follows binormal variate distribution with predictor PE
iv) We also checked for confounding between AT, RH, AP and V and found that AT has correlation with RH and AP (~ +0.5 & -0.5 respectively). Stratifying AT and checking RH and AP provided a proof that there is some causality effect because of these 2 parameters

All the above observations guided us to move to a linear regression model. To check the efficacy of our model we used LSE metric

We then started building our model step by step iteratively and starting adding the required factors

1. Base Model (Model 1): Predicted rating is same as average rating of training set
2. Model 2: Linear Regression Model. This reduced our LSE considerably
3. Model 3: K Nearest Neighbour (KNN) model. This improved our LSE slightly
4. Model 4: Regression Tree Classification model. This model failed to beat our KNN model
5. Model 5: Random Forest model. This model, averaged out the common pitfalls

Finally we summarized the LSE values for all our models and found Model 5 to be working best for our given dataset

References

1. Course Notes from Edx course HarvardX - PH125.8x, Data Science: Machine Learning [https://www.edx.org/course/data-science-machine-learning]
2. Introduction to Machine Learning by Rafael A. Irizarry [https://rafalab.github.io/dsbook/introduction-to-machine-learning.html]
3. Machine Learning online course by MIT at coursera.org [https://www.coursera.org/learn/machine-learning]