1. Overview of the Data set

The data set is called "Letter Recognition Data Set" and was taken from the UCI Repository: https://archive.ics.uci.edu/ml/datasets/letter+recognition. The data set consists of 20 different fonts and each letter within these 20 fonts was randomly distorted to produce a file of 20,000 unique stimuli. Each stimulus was converted into 16 primitive numerical attributes that were in turn submitted to classifier system. The 20 different fonts were designed by Dr. Allen V. Hershey, a mathematical physicist at the U.S. Naval Weapons Laboratory. Further detail about the data collection process is described in the paper "Letter Recognition Using Holland-Style Adaptive Classifiers" written by the creators of this data set Peter W. Frey & David J Slate.
https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.466.6265&rep=rep1&type=pdf

The data set consists of 20,000 observation and 17 variables. Among these 17 data variables, 'Letter' is the dependent variable and rest 16 features are independent variables. The dependent variable had letters from A to Z and that I have replaced with numbers ranging from 1 to 26 respectively. The data set has 16 features, some details about these features are as below:

Features	Description about Features
V2	The horizontal position, counting pixels from the left edge of the image, of the center of the smallest rectangular box that can be drawn with all "on" pixels inside the box.
V3	The vertical position, counting pixels from the bottom, of the above box.
V4	The width, in pixels, of the box.
V5	The height, in pixels, of the box.
V6	The total number of "on" pixels in the character image.
V7	The mean horizontal position of all "on" pixels relative to the center of the box and divided by the width of the box.
V8	The mean vertical position of all "on" pixels relative to the center of the box and divided by the height of the box.
V9	The mean squared value of the horizontal pixel distances as measured in V7 above.
V10	The mean squared value of the vertical pixel distances as measured in V8 above.
V11	The mean product of the horizontal and vertical distances for each "on" pixel as measured in V7 and V8 above.
V12	The mean value of the squared horizontal distance times the vertical distance for each "on" pixel.
V13	The mean value of the squared vertical distance times the horizontal distance for each "on" pixel.
V14	The mean number of edges encountered when making systematic scans from left to right at all vertical positions within the box.
V15	The sum of the vertical positions of edges encountered as measured in V14 above
V16	The mean number of edges encountered when making systematic scans of the image from bottom to top over all horizontal positions within the box.
V17	The sum of horizontal positions of edges encountered as measured in 15 above.

As the number of classes of the dependent variable is more than two, it is a multiclass classification. All the features i.e., the independent variables are continuous numerical values, so they are non-categorical.

2. Load the Data

The data set, containing all the variables and labels is present in the file named '**letter-recognition.data**' and it was loaded into variable called dataset. The files contains data separated by '**,**' so the files is read and data is divided into columns after split by '**,**'.

```
datasetFile = "letter-recognition.data"
dataset = read.csv(dataSetFile, header=FALSE, sep=",")
```

Figure: Data Load

The dataset contains 20000 observation and 17 variables which can be seen using the below command.

```
> dim(dataset)
[1] 20000    17
```

Figure: Number of Rows & Columns in the Dataset

The dataset had columns named from V1 to V17, I have replaced the dependent variable column as Letter. The columns V2 to V17 are independent variables.

```
> # Some rows from the data set
> head(dataset)
  Letter V2 V3 V4 V5 V6 V7 V8 V9 V10 V11 V12 V13 V14 V15 V16 V17
1      T  2  8  3  5  1  8 13  0   6   6  10   8   0   8   0   8
2      I  5 12  3  7  2 10  5  5   4  13   3   9   2   8   4  10
3      D  4 11  6  8  6 10  6  2   6  10   3   7   3   7   3   9
4      N  7 11  6  6  3  5  9  4   6   4   4  10   6  10   2   8
5      G  2  1  3  1  1  8  6  6   6   6   5   9   1   7   5  10
6      S  4 11  5  8  3  8  8  6   9   5   6   6   0   8   9   7
>
> # Column names
> names(dataset)
 [1] "Letter" "V2"     "V3"     "V4"     "V5"     "V6"     "V7"     "V8"     "V9"     "V10"    "V11"
[12] "V12"    "V13"    "V14"    "V15"    "V16"    "V17"
```

Figure: Some rows & the Column names of the Data set

The below figure consists of the summary of the data set, this includes all feature variables.

```
> summary(dataFeatureVariable)
       V2               V3               V4               V5
 Min.   : 0.000   Min.   : 0.000   Min.   : 0.000   Min.   : 0.000
 1st Qu.: 3.000   1st Qu.: 5.000   1st Qu.: 4.000   1st Qu.: 4.000
 Median : 4.000   Median : 7.000   Median : 5.000   Median : 6.000
 Mean   : 4.024   Mean   : 7.035   Mean   : 5.122   Mean   : 5.372
 3rd Qu.: 5.000   3rd Qu.: 9.000   3rd Qu.: 6.000   3rd Qu.: 7.000
 Max.   :15.000   Max.   :15.000   Max.   :15.000   Max.   :15.000
       V6               V7               V8               V9              V10
 Min.   : 0.000   Min.   : 0.000   Min.   : 0.0   Min.   : 0.000   Min.   : 0.000
 1st Qu.: 2.000   1st Qu.: 6.000   1st Qu.: 6.0   1st Qu.: 3.000   1st Qu.: 4.000
 Median : 3.000   Median : 7.000   Median : 7.0   Median : 4.000   Median : 5.000
 Mean   : 3.506   Mean   : 6.898   Mean   : 7.5   Mean   : 4.629   Mean   : 5.179
 3rd Qu.: 5.000   3rd Qu.: 8.000   3rd Qu.: 9.0   3rd Qu.: 6.000   3rd Qu.: 7.000
 Max.   :15.000   Max.   :15.000   Max.   :15.0   Max.   :15.000   Max.   :15.000
      V11              V12              V13              V14
 Min.   : 0.000   Min.   : 0.000   Min.   : 0.000   Min.   : 0.000
 1st Qu.: 7.000   1st Qu.: 5.000   1st Qu.: 7.000   1st Qu.: 1.000
 Median : 8.000   Median : 6.000   Median : 8.000   Median : 3.000
 Mean   : 8.282   Mean   : 6.454   Mean   : 7.929   Mean   : 3.046
 3rd Qu.:10.000   3rd Qu.: 8.000   3rd Qu.: 9.000   3rd Qu.: 4.000
 Max.   :15.000   Max.   :15.000   Max.   :15.000   Max.   :15.000
      V15              V16              V17
 Min.   : 0.000   Min.   : 0.000   Min.   : 0.000
 1st Qu.: 8.000   1st Qu.: 2.000   1st Qu.: 7.000
 Median : 8.000   Median : 3.000   Median : 8.000
 Mean   : 8.339   Mean   : 3.692   Mean   : 7.801
 3rd Qu.: 9.000   3rd Qu.: 5.000   3rd Qu.: 9.000
 Max.   :15.000   Max.   :15.000   Max.   :15.000
```

Figure: Summary of the Data set

3. Exploratory Data Analysis

Before we begin training the data set, let's first check the quality of the data and make the data ready for further analysis. To start with, I have replaced dependent variable column name from V1 to '**Letter**' for better readability.

```
> # View column names in the data set
> colnames(dataset)
 [1] "V1"  "V2"  "V3"  "V4"  "V5"  "V6"  "V7"  "V8"  "V9"  "V10" "V11" "V12" "V13" "V14" "V15" "V16"
[17] "V17"
>
> # Change the dependent variable column name to Letter
> colnames(dataset)[1] <- "Letter"
> colnames(dataset)
 [1] "Letter" "V2"     "V3"     "V4"     "V5"     "V6"     "V7"     "V8"     "V9"     "V10"    "V11"
[12] "V12"    "V13"    "V14"    "V15"    "V16"    "V17"
>
> # position of the class variable
> which(names(dataset)=='Letter')
[1] 1
```

Let's look at the spectral count, values of class variables & the type of classification.

```
> datasetTable = table(dataset$Letter)
> datasetTable

  A   B   C   D   E   F   G   H   I   J   K   L   M   N   O   P   Q   R   S   T   U   V   W   X   Y
789 766 736 805 768 775 773 734 755 747 739 761 792 783 753 803 783 758 748 796 813 764 752 787 786
  Z
734
>
> names(datasetTable)
 [1] "A" "B" "C" "D" "E" "F" "G" "H" "I" "J" "K" "L" "M" "N" "O" "P" "Q" "R" "S" "T" "U" "V" "W" "X"
[25] "Y" "Z"
> print(ifelse(length(datasetTable)==2, "Binary Classification", "MultiClassClassification"))
[1] "MultiClassClassification"
>
```

Figure: Depicting Spectral Count & Type of Classification

The output of the 'names(datasetTable)' shows that the classes contain values in the range of A-Z, i.e., 26 Letters and hence it is a Multi Class Classification. Before moving forward, we will replace the letters in numeric values so that classification can be done properly. After the update the feature variable will contain numeric values as shown below.

```
> dataset$Letter = factor(dataset$Letter,
+                         levels = c("A", "B","C","D","E","F","G","H","I","J","K","L","M","N","O","P",
+                                    "Q","R","S","T","U","V","W","X","Y","Z"),
+                         labels = c(1, 2, 3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,2
5,26))
>
>
> datasetTable = table(dataset$Letter)
> datasetTable

  1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25
789 766 736 805 768 775 773 734 755 747 739 761 792 783 753 803 783 758 748 796 813 764 752 787 786
 26
734
> names(datasetTable)
 [1] "1"  "2"  "3"  "4"  "5"  "6"  "7"  "8"  "9"  "10" "11" "12" "13" "14" "15" "16" "17" "18" "19"
[20] "20" "21" "22" "23" "24" "25" "26"
```

Figure: Depicting Spectral Count with Letters replaced

The same can be visualized using Bar Chart

```
> barplot(datasetTable, main="Letter Distribution", xlab="Letters",
 ylab="Count of Letters", col=c("darkblue","red"))
>
```

Figure: Letter Distribution v/s Count of Letters

From the above figure we can understand that the count of U (i.e., 21) is more in the data as compared to any other letters which is 813. In addition, it can be observed that H and Z have least count (i.e., 8 & 26) to any other letters which is about 734.

Let's see by plotting histogram, how the distribution of each feature variables looks like.

```
> z1 <- ggplot(data=dataset,aes(V2)) + geom_histogram(breaks=seq(0,15,by=1),color="black",fill
="red") + labs(x="V2",y="Count")
> z2 <- ggplot(data=dataset,aes(V3)) + geom_histogram(breaks=seq(0,15,by=1),color="black",fill
="red") + labs(x="V3",y="Count")
> z3 <- ggplot(data=dataset,aes(V4)) + geom_histogram(breaks=seq(0,15,by=1),color="black",fill
="red") + labs(x="V4",y="Count")
> z4 <- ggplot(data=dataset,aes(V5)) + geom_histogram(breaks=seq(0,15,by=1),color="black",fill
="red")+ labs(x="V5",y="Count")
> plot_grid(z1, z2, z3, z4, ncol = 2, nrow = 2)
```

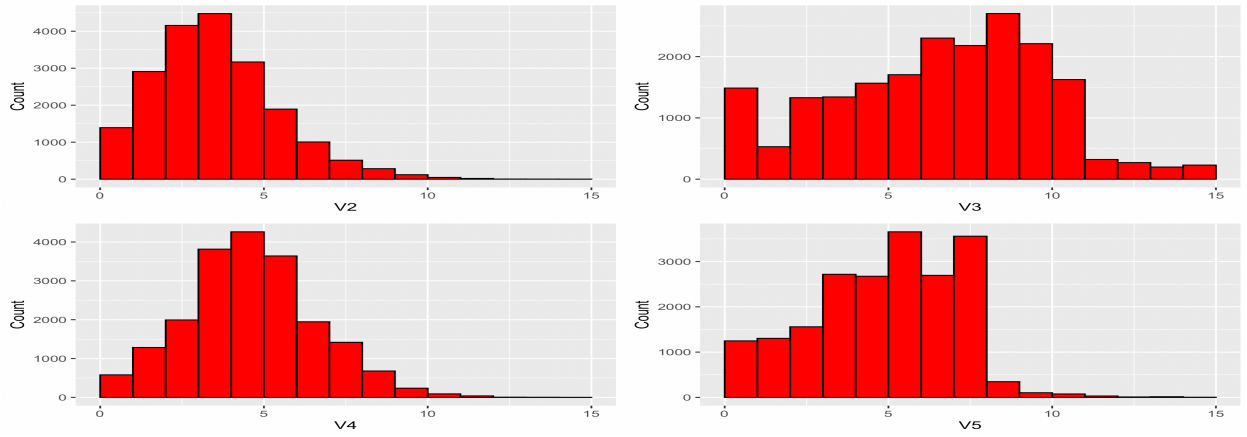

Figure: Distribution of feature variable V2-V5 using Histogram

```
> z5 <- ggplot(data=dataset,aes(V6)) + geom_histogram(breaks=seq(0,15,by=1),color="black",fill
="red") + labs(x="V6",y="Count")
> z6 <- ggplot(data=dataset,aes(V7)) + geom_histogram(breaks=seq(0,15,by=1),color="black",fill
="red") + labs(x="V7",y="Count")
> z7 <- ggplot(data=dataset,aes(V8)) + geom_histogram(breaks=seq(0,15,by=1),color="black",fill
="red") + labs(x="V8",y="Count")
> z8 <- ggplot(data=dataset,aes(V9)) + geom_histogram(breaks=seq(0,15,by=1),color="black",fill
="red")+ labs(x="V9",y="Count")
> plot_grid(z5, z6, z7, z8, ncol = 2, nrow = 2)
```

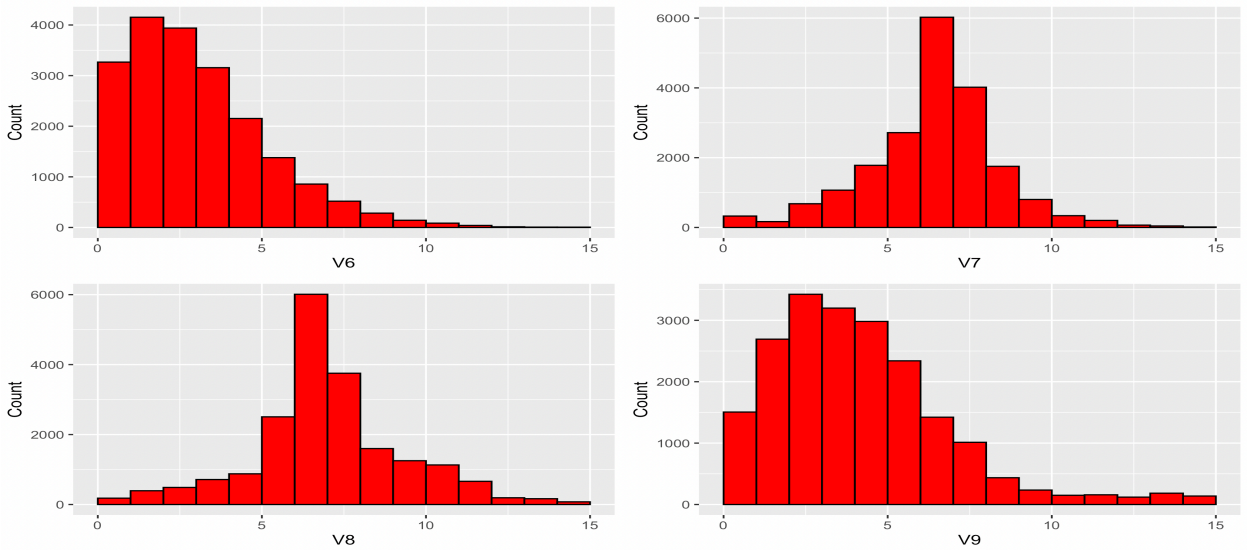

Figure: Distribution of feature variable V6-V8 using Histogram

```
> z9 <- ggplot(data=dataset,aes(V10)) + geom_histogram(breaks=seq(0,15,by=1),color="black",fill
="red") + labs(x="V10",y="Count")
> z10 <- ggplot(data=dataset,aes(V11)) + geom_histogram(breaks=seq(0,15,by=1),color="black",fill
="red") + labs(x="V11",y="Count")
> z11 <- ggplot(data=dataset,aes(V12)) + geom_histogram(breaks=seq(0,15,by=1),color="black",fill
="red") + labs(x="V12",y="Count")
> z12 <- ggplot(data=dataset,aes(V13)) + geom_histogram(breaks=seq(0,15,by=1),color="black",fill
="red")+ labs(x="V13",y="Count")
> plot_grid(z9, z10, z11, z12, ncol = 2, nrow = 2)
```

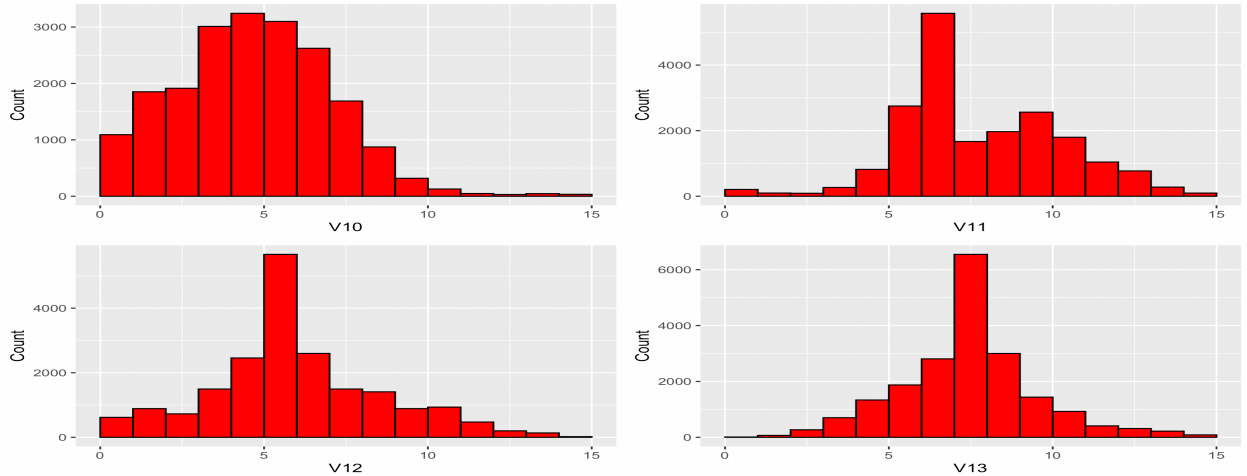

Figure: Distribution of feature variable V10-V13 using Histogram

```
> z13 <- ggplot(data=dataset,aes(V14)) + geom_histogram(breaks=seq(0,15,by=1),color="black",fill
="red") + labs(x="V14",y="Count")
> z14 <- ggplot(data=dataset,aes(V15)) + geom_histogram(breaks=seq(0,15,by=1),color="black",fill
="red") + labs(x="V15",y="Count")
> z15 <- ggplot(data=dataset,aes(V16)) + geom_histogram(breaks=seq(0,15,by=1),color="black",fill
="red") + labs(x="V16",y="Count")
> z16 <- ggplot(data=dataset,aes(V17)) + geom_histogram(breaks=seq(0,15,by=1),color="black",fill
="red")+ labs(x="V17",y="Count")
> plot_grid(z13, z14, z15, z16, ncol = 2, nrow = 2)
```

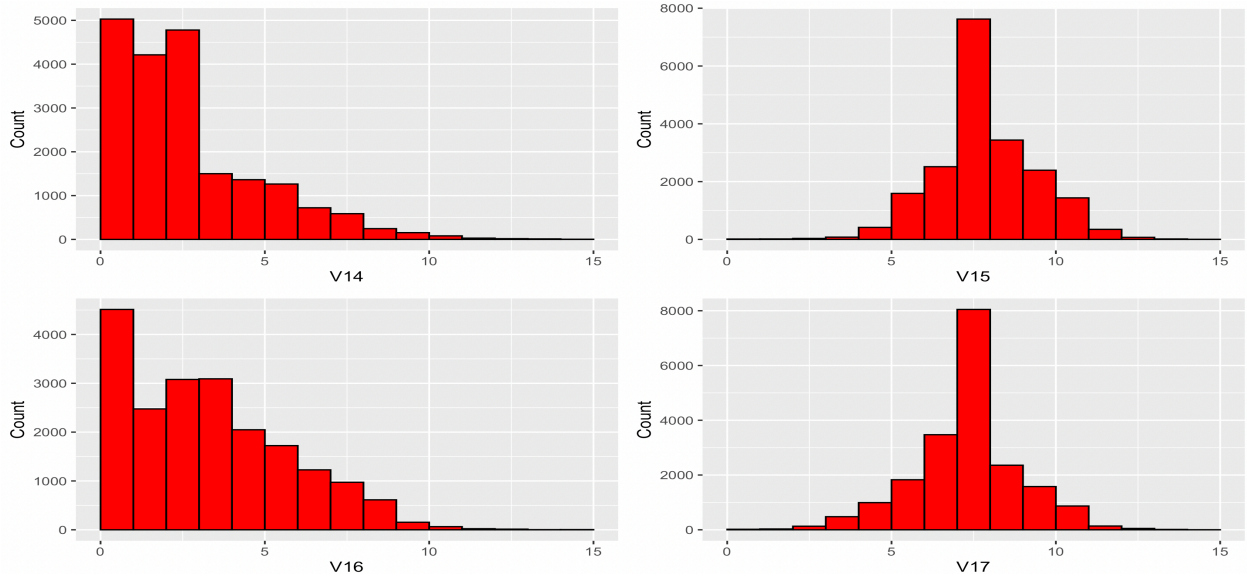

Figure: Distribution of feature variable V14-V17 using Histogram

From the above figures we can understand that V2, V3, V4, V5, V6, V9, V10, V14, V15 are right skewed and V12 is slightly right skewed. Other attributes such as V7, V11, V15, V17 are almost symmetric while V13 is symmetric.

Next step is to check if the data contains any missing or null values. The below shows that there are no missing or null values in our dataset.

```
> sum(is.na(dataset))
[1] 0
```

Figure: Check for Missing or Null Value

By looking at the spectral count, the data set seems to be balanced. The same can be checked by confirming for any class label, if the number of observations of one class to another is less than 1:13, in that case, the class would be imbalanced. Here the result also confirms that the data set is balanced.

```
> if(any(datasetTable<(nrow(dataset)*1/26*1/13) | any(datasetTable>(nrow(dataset)*1/26*13)))) {
+   3 + imbalanced_classes=names(datasetTable)[datasetTable<(nrow(dataset)*1/26*1/13) |
+                                              datasetTable>(nrow(dataset)*1/26*13)]
+   print("Imbalanced Classes:")
+   print(imbalanced_classes)
+ }else {
+   print("Data set is Balanced")
+ }
[1] "Data set is Balanced"
```

Figure: Check if Data Set is Balanced or Imbalanced

The next to check is to see if standardization or normalization is needed for the data set. To check we will plot mean & standard deviation of each feature column.

```
> par(mfrow=c(2,1),mar=c(2,4,2,2))
> plot(mean_data)
> plot(std_dev_data)
> std_dev_data=round(apply(dataset[,-T],2,sd),4)
> mean_data=round(apply(dataset[,-T],2,mean),4)
> par(mfrow=c(2,1),mar=c(2,4,2,2))
> plot(mean_data)
> plot(std_dev_data)
```

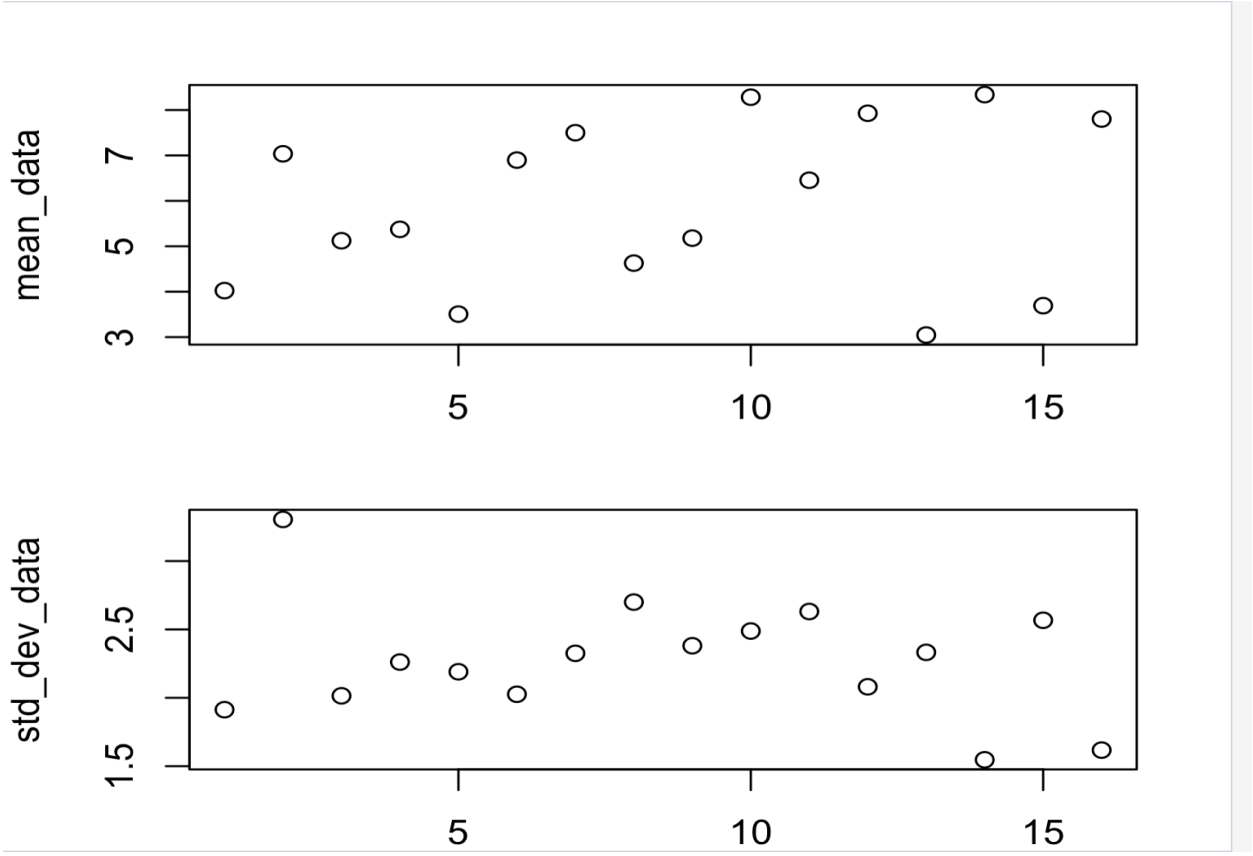

Figure: Mean & Standard Deviation of each Feature Column

From the graph it looks that normalization will be needed, we will do normalization at a later point in time before we begin the training phase.

Next step is to check if outliers are observed in our data set. This can be done by plotting box plot for each variable and this will also help in understanding the range of the data.

```
> p1 <- ggplot(dataset,aes(x="",y=V2)) + geom_boxplot(fill = "darkolivegreen1", color = "black",
width=0.5) + theme(axis.title.y=element_blank()) + labs(x="V2")
> p2 <- ggplot(dataset,aes(x="",y=V3)) + geom_boxplot(fill = "darkolivegreen1", color = "black")
+ theme(axis.title.y=element_blank()) + labs(x="V3")
> p3 <- ggplot(dataset,aes(x="",y=V4)) + geom_boxplot(fill = "darkolivegreen1", color = "black")
+ theme(axis.title.y=element_blank()) + labs(x="V4")
> p4 <- ggplot(dataset,aes(x="",y=V5)) + geom_boxplot(fill = "darkolivegreen1", color = "black")
+ theme(axis.title.y=element_blank()) + labs(x="V5")
> p5 <- ggplot(dataset,aes(x="",y=V6)) + geom_boxplot(fill = "darkolivegreen1", color = "black",
width=0.5) + theme(axis.title.y=element_blank()) + labs(x="V6")
> p6 <- ggplot(dataset,aes(x="",y=V7)) + geom_boxplot(fill = "darkolivegreen1", color = "black")
+ theme(axis.title.y=element_blank()) + labs(x="V7")
> p7 <- ggplot(dataset,aes(x="",y=V8)) + geom_boxplot(fill = "darkolivegreen1", color = "black")
+ theme(axis.title.y=element_blank()) + labs(x="V8")
> p8 <- ggplot(dataset,aes(x="",y=V9)) + geom_boxplot(fill = "darkolivegreen1", color = "black")
+ theme(axis.title.y=element_blank()) + labs(x="V9")
> plot_grid(p1, p2, p3, p4,p5,p6,p7,p8,ncol = 8, nrow = 1)
```

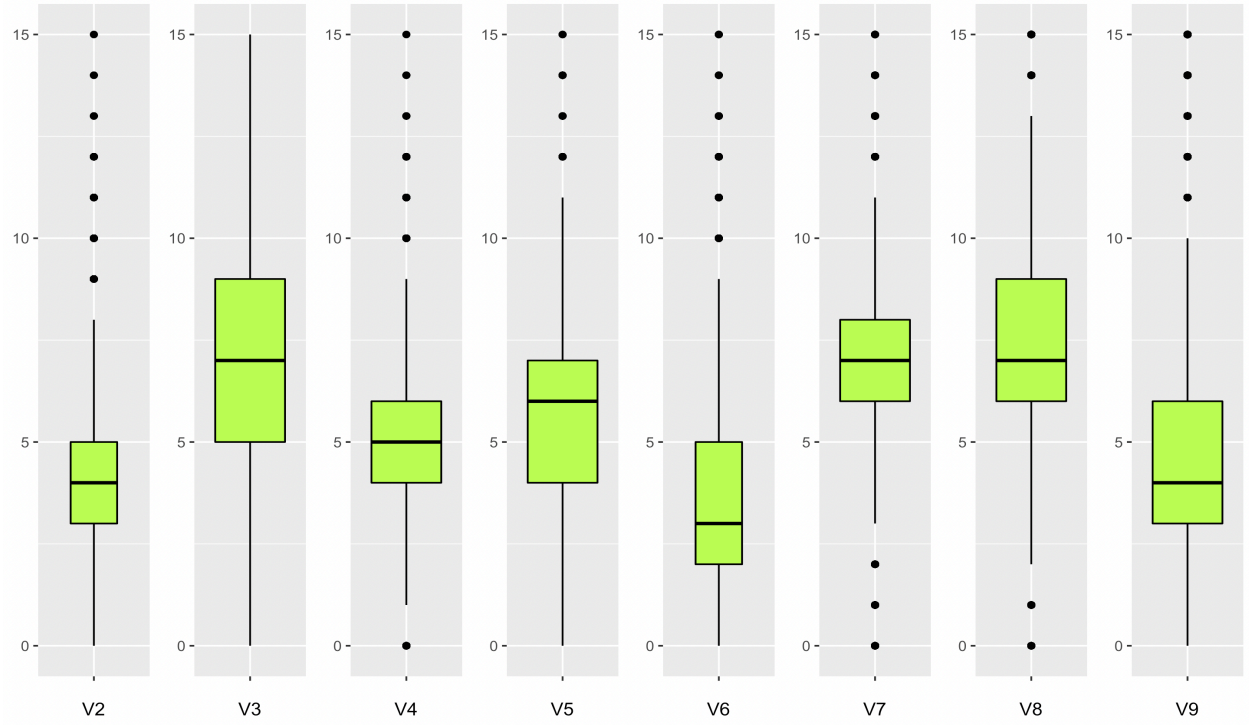

Figure: Box plot of Feature Variable from V2-V9

```
> p9 <- ggplot(dataset,aes(x="",y=V10)) + geom_boxplot(fill = "darkolivegreen1", color = "black",w
idth=0.5) + theme(axis.title.y=element_blank()) + labs(x="V10")
> p10 <- ggplot(dataset,aes(x="",y=V11)) + geom_boxplot(fill = "darkolivegreen1", color = "black")
+ theme(axis.title.y=element_blank()) + labs(x="V11")
> p11 <- ggplot(dataset,aes(x="",y=V12)) + geom_boxplot(fill = "darkolivegreen1", color = "black")
+ theme(axis.title.y=element_blank()) + labs(x="V12")
> p12 <- ggplot(dataset,aes(x="",y=V13)) + geom_boxplot(fill = "darkolivegreen1", color = "black")
+ theme(axis.title.y=element_blank()) + labs(x="V13")
> p13 <- ggplot(dataset,aes(x="",y=V14)) + geom_boxplot(fill = "darkolivegreen1", color = "black",
width=0.5) + theme(axis.title.y=element_blank()) + labs(x="V14")
> p14 <- ggplot(dataset,aes(x="",y=V15)) + geom_boxplot(fill = "darkolivegreen1", color = "black")
+ theme(axis.title.y=element_blank()) + labs(x="V15")
> p15 <- ggplot(dataset,aes(x="",y=V16)) + geom_boxplot(fill = "darkolivegreen1", color = "black")
+ theme(axis.title.y=element_blank()) + labs(x="V16")
> p16 <- ggplot(dataset,aes(x="",y=V17)) + geom_boxplot(fill = "darkolivegreen1", color = "black")
+ theme(axis.title.y=element_blank()) + labs(x="V17")
> plot_grid(p9, p10, p11, p12,p13, p14, p15, p16, ncol = 8, nrow = 1)
```

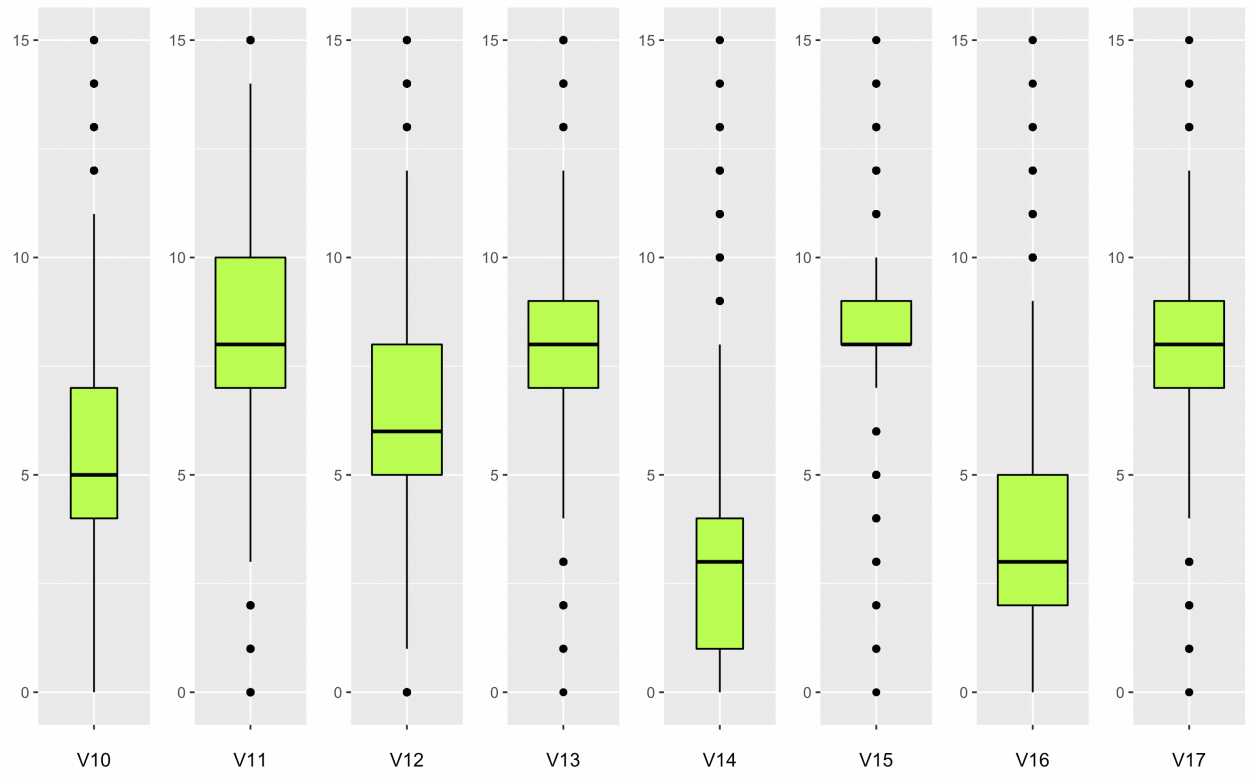

Figure: Box plot of Feature Variable from V10-V17

The box plots shows that the outliers are present in the data set. The feature variables V2, V4, V6, V9, V10, V14 and V16 outliers are present on higher end of the plot while for others except V3 the outliers are present on both sides of the plot.

The next step will help us to identify the number of outliers in the data set. This shows that out of 20,000 observations, there are 1736 outliers which is quite less as compared to the whole data. So, the effect of outlier might be very less but to verify we will remove these outliers after correctly predicting feature importance and at a later stage we will be testing if the presence of outliers causes any major change in prediction.

```
> outliers_row=c() # Loop over the feature columns
> for(i in colnames(dataFeatureVariable)) {
+   data_mean=mean(dataset[,i]) # Mean of the data in feature column i
+   data_sd=sd(dataset[,i]) # Standard deviation of the data in feature column i
+   low_cutoff=data_mean-3*data_sd # Lower cutoff value
+   upper_cutoff=data_mean+3*data_sd # Upper cutoff value
+   outliers_idx=which(dataset[,i]<low_cutoff | dataset[,i]>upper_cutoff)
+   outliers_row=c(outliers_row,outliers_idx)
+ }
> outliers_row=unique(outliers_row) # Remove duplicated row indices
> print(paste("Number of Outilers =",length(outliers_row)))
[1] "Number of Outilers = 1736"
>
```

Figure: Depicting number of Outliers in our Data set

Let's understand the relation between feature variables. To identify this, we can use Pearson's Coefficient to understand dependency between each feature variable with other variables.

```
> library(corrplot)
corrplot 0.92 loaded
> k1 = cor(dataset[sapply(dataset,is.numeric)],method="pearson")
> corrplot(k1,method="number")
> corrplot(k1,method="pie")
```

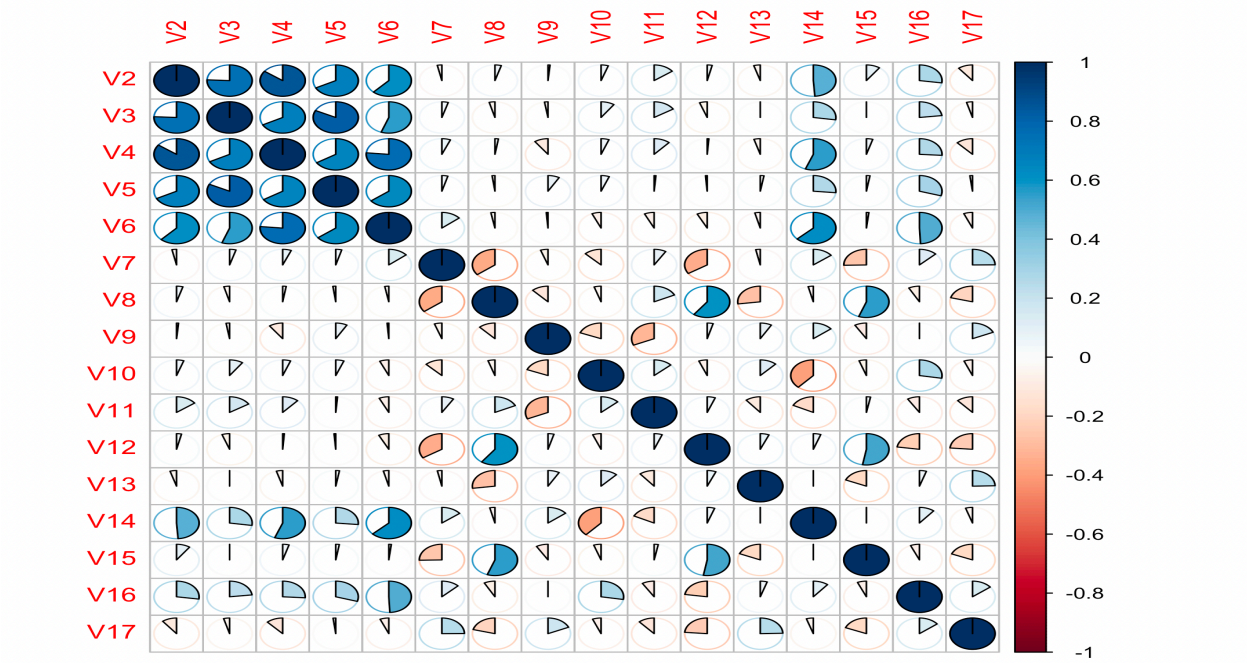

Figure: Pearson's Coefficient for Feature Variables

It can be observed from the above figure that V2 have significant positive correlations with V3, V4, V5, V6 and V14. For V3 significant positive correlations with V2, V4, V5 and V6. For V4 significant positive correlations can be observed with V2, V3, V5, V6 and V14. For V5 significant positive correlations can be observed with V2, V3, V4 and V5. For V6 significant positive correlations can be observed with V2, V3, V4, V5, V14 and V16. For V8 significant positive correlation are observed with V12 and V15. V12 is showing significant positive correlation with V15. For V14 positive correlations can be observed with V2, V4 and V6. V15 has significant positive correlations with V9 and V12. V16 shows significant positive correlation with V6. For variables V7, V8, V9, V10, V11, V12, V13 and V14 shows negative correlations.

We will also identify if any constant predictor exists, a constant predictor exists when the number of distinct values in a feature column is less than 2. The same we are checking with our data set

```
> const_pred = unlist(lapply(colnames(dataset),FUN=function(x) {
+   TBL=table(dataset[[x]])
+   ifelse(length(names(TBL)) < 2, -1*x,x)}
+ ))
> print(ifelse(any(const_pred<0),"Constant Predictors Exist","No Constant Predictors"))
[1] "No Constant Predictors"
```

Figure: Check Constant Predictors

The other way to identify the number of correlated predictors is as shown below. Using cor method we can identify the correlational data. The correlated index of 0 indicates that there is no correlation and index of 1 indicates that they are fully correlated. Accordingly, if the correlated predictors are correlated, they will have the correlation index greater than 0.5. From the below code we can understand that the number of correlated predictors for our data set is 7 & the correlated predictors are V2, V3, V4, V5, V11, V13, V14.

```
> # check for correlated predictors
> cordata=cor(dataset[,-which(names(dataset)=='Letter')])
> print(ifelse(any(abs(cordata[cordata!=1])>0.5),"Correlated Predictors Exist","
+ No Correlated Predictors"))
[1] "Correlated Predictors Exist"
> cor_index=which(abs(cordata)>0.5 & abs(cordata)!=1, arr.ind = T)
> cor_index=cor_index[!duplicated(cbind(pmax(cor_index[,1], cor_index[,2]), pmin(cor_index[,
1], cor_index[,2]))),]
> tbl_cor_index=table(cor_index[,1])
> cor_index_num=length(tbl_cor_index) # Number of correlated predictors
> print(paste("Number of correlated predictors = ",cor_index_num))
[1] "Number of correlated predictors =  7"
>
> names(tbl_cor_index)
[1] "2"  "3"  "4"  "5"  "11" "13" "14"
> cor_attributes=as.numeric(names(tbl_cor_index))
>
> # show correlated variables
> corelated_variables = ''
> for(i in cor_attributes) {
+   corelated_variables <- paste(corelated_variables, colnames(dataset[i]))
+ }
> corelated_variables
[1] " V2 V3 V4 V5 V11 V13 V14"
>
```

Figure: Check for Correlated Predictors

We will remove these correlated predictors at later stage before training our model and see its impact.

The next step is to divide into Training & Testing Data set which is done as shown below.

```
> # Divide into training & testing data set
> set.seed(43)
> randomized=dataset[sample(1:nrow(dataset),nrow(dataset)),]
> tridx=sample(1:nrow(dataset),0.7*nrow(dataset),replace=F)
> trainingDataset = randomized[tridx,]
> testingDataset = randomized[-tridx,]
>
```

Figure: Divide into Training & Testing Dataset

We will confirm by checking if the class distribution is similar between training & testing data set.

```
> # confirm by checking if the data set split is consistent throughout
> table(dataset$Letter)/nrow(dataset)

      1       2       3       4       5       6       7       8       9      10
0.03945 0.03830 0.03680 0.04025 0.03840 0.03875 0.03865 0.03670 0.03775 0.03735
     11      12      13      14      15      16      17      18      19      20
0.03695 0.03805 0.03960 0.03915 0.03765 0.04015 0.03915 0.03790 0.03740 0.03980
     21      22      23      24      25      26
0.04065 0.03820 0.03760 0.03935 0.03930 0.03670
> table(trainingDataset$Letter)/nrow(trainingDataset)

         1          2          3          4          5          6          7
0.03885714 0.03650000 0.03614286 0.03985714 0.03842857 0.03971429 0.03964286
         8          9         10         11         12         13         14
0.03550000 0.03850000 0.03685714 0.03750000 0.03828571 0.03885714 0.03942857
        15         16         17         18         19         20         21
0.03835714 0.04057143 0.03942857 0.03771429 0.03750000 0.04000000 0.04071429
        22         23         24         25         26
0.03928571 0.03685714 0.04021429 0.03850000 0.03678571
> table(testingDataset$Letter)/nrow(testingDataset)

         1          2          3          4          5          6          7
0.04083333 0.04250000 0.03833333 0.04116667 0.03833333 0.03650000 0.03633333
         8          9         10         11         12         13         14
0.03950000 0.03600000 0.03850000 0.03566667 0.03750000 0.04133333 0.03850000
        15         16         17         18         19         20         21
0.03600000 0.03916667 0.03850000 0.03833333 0.03716667 0.03933333 0.04050000
        22         23         24         25         26
0.03566667 0.03933333 0.03733333 0.04116667 0.03650000
```

Figure: Check for Class Distribution

Further analysis we will do before training the model, this will help in understanding the behavior of the algorithm based on our Data set.