Project 1

Intro to Machine Learning

Summer 2018

Amy Nguyen

Reading In The Data

For my regression scenario I have decided to use a data set containing Facebook metrics. In order to import the data I downloaded the excel file from UCI and saved it as a csv file. In order to read the data I used the read.delim() function as I needed to identify the separators for the columns which was the character ';'.

```
#Facebook Data
Facebook_df <- read.delim("C:/Users/amy19/Desktop/facebook.csv", sep=";")
```

For my classification scenario I have decided to use a data set that looked at movies released in 2014 and 2015 and tracked features such as genre, gross product, budget, etc. In order to import the data, I did the same steps as the above for importing the Facebook data. However since the columns were separated by commas I used the read.csv() function since I did not need to identify the separators.

```
#Movies Data
Movies_df <- read.csv("C:/Users/amy19/Desktop/2014Movies.csv")
```

Regression: Facebook Metrics

Data Source: http://archive.ics.uci.edu/ml/datasets/Facebook+Comment+Volume+Dataset

Naming Variables / Columns

In order to get a feel for the data let's begin by looking at the names of the variables and the structure.

```
#Variable Names
names(Facebook_df)
```

```
##  [1] "Page.total.likes"
##  [2] "Type"
##  [3] "Category"
##  [4] "Post.Month"
##  [5] "Post.Weekday"
##  [6] "Post.Hour"
##  [7] "Paid"
##  [8] "Lifetime.Post.Total.Reach"
##  [9] "Lifetime.Post.Total.Impressions"
## [10] "Lifetime.Engaged.Users"
## [11] "Lifetime.Post.Consumers"
```

```
## [12] "Lifetime.Post.Consumptions"
## [13] "Lifetime.Post.Impressions.by.people.who.have.liked.your.Page"
## [14] "Lifetime.Post.reach.by.people.who.like.your.Page"
## [15] "Lifetime.People.who.have.liked.your.Page.and.engaged.with.your.post"
## [16] "comment"
## [17] "like"
## [18] "share"
## [19] "Total.Interactions"
```

Since I'm not the biggest fan of the periods in the variable names and would like to simplify some of the names I'm going to rename the column names using the names() function once again.

```
names(Facebook_df) <- c("Total Page Likes", "Type", "Category", "Monthly Post
s", "Weekly Posts",
                        "Hourly Posts", "Paid", "Lifetime Post Reach", "Lifet
ime Post Impressions",
                        "Lifetime Engaged Users", "Lifetime Post Consumers",
"Lifetime Post Consumptions",
                        "Lifetime Post Impressions By Page Likers", "Lifetime
Post Reach By Page Likers",
                        "Lifetime Page Likers and Engagers", "Comments", "Lik
es", "Share",
                        "Total Interactions")
names(Facebook_df)
```

```
##  [1] "Total Page Likes"
##  [2] "Type"
##  [3] "Category"
##  [4] "Monthly Posts"
##  [5] "Weekly Posts"
##  [6] "Hourly Posts"
##  [7] "Paid"
##  [8] "Lifetime Post Reach"
##  [9] "Lifetime Post Impressions"
## [10] "Lifetime Engaged Users"
## [11] "Lifetime Post Consumers"
## [12] "Lifetime Post Consumptions"
## [13] "Lifetime Post Impressions By Page Likers"
## [14] "Lifetime Post Reach By Page Likers"
## [15] "Lifetime Page Likers and Engagers"
## [16] "Comments"
## [17] "Likes"
## [18] "Share"
## [19] "Total Interactions"
```

Now let's look at the structure of our data set using the str() function.

```
str(Facebook_df)
```

```
## 'data.frame':    499 obs. of  19 variables:
##  $ Total Page Likes                          : int  139441 139441 139441 139
441 139441 139441 139441 139441 139441 139441 ...
##  $ Type                                      : Factor w/ 4 levels "Link","Ph
oto",..: 2 3 2 2 2 3 2 2 3 2 ...
##  $ Category                                  : int  2 2 3 2 2 2 3 3 2 3 ...
##  $ Monthly Posts                             : int  12 12 12 12 12 12 12 12
12 12 ...
##  $ Weekly Posts                              : int  4 3 3 2 2 1 1 7 7 6 ...
##  $ Hourly Posts                              : int  3 10 3 10 3 9 3 9 3 10 .
..
##  $ Paid                                      : int  0 0 0 1 0 0 1 1 0 0 ...
##  $ Lifetime Post Reach                       : int  2752 10460 2413 50128 72
44 10472 11692 13720 11844 4694 ...
##  $ Lifetime Post Impressions                 : int  5091 19057 4373 87991 13
594 20849 19479 24137 22538 8668 ...
##  $ Lifetime Engaged Users                    : int  178 1457 177 2211 671 11
91 481 537 1530 280 ...
##  $ Lifetime Post Consumers                   : int  109 1361 113 790 410 107
3 265 232 1407 183 ...
##  $ Lifetime Post Consumptions                : int  159 1674 154 1119 580 13
89 364 305 1692 250 ...
##  $ Lifetime Post Impressions By Page Likers: int  3078 11710 2812 61027 62
28 16034 15432 19728 15220 4309 ...
##  $ Lifetime Post Reach By Page Likers        : int  1640 6112 1503 32048 320
0 7852 9328 11056 7912 2324 ...
##  $ Lifetime Page Likers and Engagers         : int  119 1108 132 1386 396 10
16 379 422 1250 199 ...
##  $ Comments                                  : int  4 5 0 58 19 1 3 0 0 3 ..
.
##  $ Likes                                     : int  79 130 66 1572 325 152 2
49 325 161 113 ...
##  $ Share                                     : int  17 29 14 147 49 33 27 14
31 26 ...
##  $ Total Interactions                        : int  100 164 80 1777 393 186
279 339 192 142 ...
```

From the structure we can see that there are 499 objects and 19 variables. Of the 19 variables 18 represent are quantitative data as indicated by the int data type. The only qualitative variable is "Type" which is represented by a factor.

Choosing Our Predictor And Target

For this set of data let's look at Total Page Likes, Paid, Lifetime Engaged Users and Lifetime Page Likers and Engagers to determine the number of total interactions a Facebook post receives. I choose these variables as my predictors because from personal use of Facebook I would think that they would make good predictors.

Let's start by simplifing our data frame and pulling out only the columns we want.

```
newFacebook <- subset(Facebook_df, select = c("Total Page Likes", "Paid", "Li
fetime Engaged Users", "Lifetime Page Likers and Engagers", "Total Interactio
ns"))
```

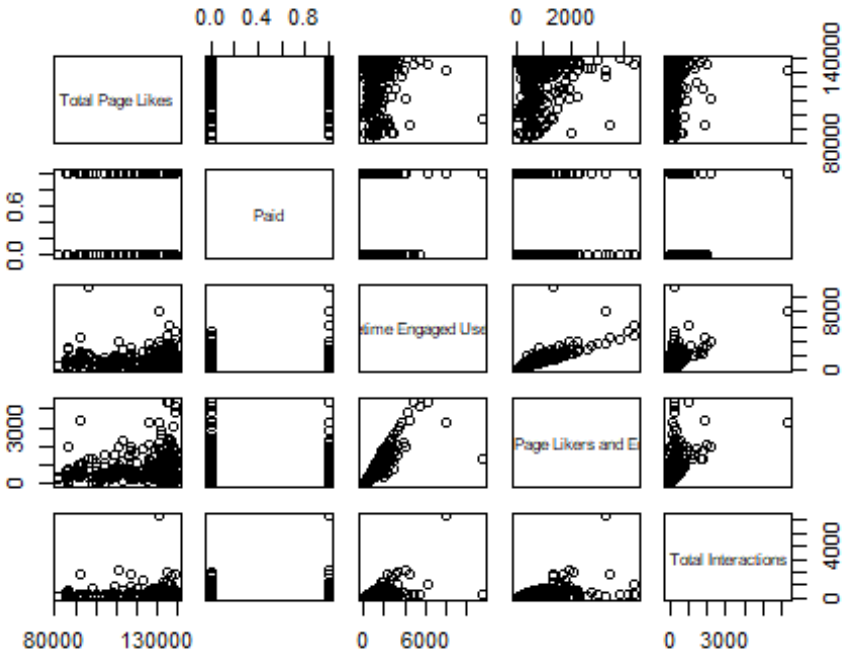
Variable Data

Now let's look at some data on each variable.

pairs()

Let's use the pairs() function to look at possible correlations between the variables.

```
pairs(newFacebook)
```



Total Page Likes

Total Page Likes are the number of likes for each individual Facebook page in our data set. Let's look at a summary for our 'Total Page Likes'.

```
summary(newFacebook$'Total Page Likes')

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   81370  113028  129600  123278  136393  139441
```

Paid

Paid is represented in as 0 and 1 where 0 means that the post is not paid and 1 means that the post is paid. Let's look at the data to get a taste of how many pages have paided posts and use sum to calculate the total number of pages that have a paid post in our data.

```
head(newFacebook$'Paid')
```

```
## [1] 0 0 0 1 0 0
```

```
tail(newFacebook$'Paid')
```

```
## [1] 0 0 0 0 0 0
```

```
paste("Total number of paid posts: ",sum(newFacebook$'Paid'))
```

```
## [1] "Total number of paid posts:  139"
```

Therefore, we can see of or 499 sampled pages, 139 pages have paid for a post which is approximately 28%.

Lifetime Engaged Users

Lifetime Engaged Users is the total number of people to have ever interacted with a page whether that is liking, commenting, or sharing a post. Let's look at a summary for this data.

```
summary(newFacebook$'Lifetime Engaged Users')
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     9.0   393.5   626.0   921.1  1062.0 11452.0
```

Lifetime Page Likers and Engagers

Lifetime Page Likers and Engagers is the total number of people that have both liked the page and interact with the posts that the page makes. Let's look at a summary for this data.

```
summary(newFacebook$'Lifetime Page Likers and Engagers')
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     9.0   291.0   413.0   610.6   656.5  4376.0
```

Total Interaction

Now let's get a summary of our target variable, 'Total Interaction'.

```
summary(newFacebook$'Total Interactions')
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     0.0    71.0   124.0   212.3   229.0  6334.0
```

Linear Regression Model

```
#Setting seed and randomly sampling
set.seed(0000)
i <- sample(1:nrow(newFacebook), nrow(newFacebook)*0.75, replace=FALSE)
```

```r
#Seperating into train and test data
ftrain <- newFacebook[i,]
ftest <- newFacebook[-i,]

#Creating linear model
lmf <- lm(`Total Interactions`~., data = ftrain)

#Summary/Metrics of the linear regression model
summary(lmf)
```

```
##
## Call:
## lm(formula = `Total Interactions` ~ ., data = ftrain)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2254.3   -69.2   -17.6    44.1  4513.2
##
## Coefficients:
##                                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)                      -3.139e+02  1.390e+02  -2.259   0.0245
## `Total Page Likes`                2.536e-03  1.116e-03   2.272   0.0237
## Paid                              3.007e+01  4.049e+01   0.743   0.4582
## `Lifetime Engaged Users`          2.203e-01  3.044e-02   7.238 2.66e-12
## `Lifetime Page Likers and Engagers` -1.619e-03  4.979e-02  -0.033   0.9741
##
## (Intercept)                      *
## `Total Page Likes`               *
## Paid
## `Lifetime Engaged Users`         ***
## `Lifetime Page Likers and Engagers`
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 340.1 on 369 degrees of freedom
## Multiple R-squared:  0.3351, Adjusted R-squared:  0.3279
## F-statistic: 46.49 on 4 and 369 DF,  p-value: < 2.2e-16
```

From the summary we can see a few key interesting points such as the fact that Lifetime Engaged Users is a good predictor but not Lifetime Page Likers and Engagers. We can also see that Total Page Likes makes an okay predictor as well.

Funny enough, 'Paid' which represents whether or not the post was paid to be spread is not a good predictor. This can be further proved is we look at the correlation using the cor() function.

```r
cor(ftrain$`Total Page Likes`, ftrain$Paid)
```

```
## [1] 0.04607148
```

We found the correlation to only be 0.046 meaning that there is a very small positive correlation. Based off this one might say that paid posts are not worth it if you are looking for user interaction.

Testing The Model

```
predFL<- predict(lmf, newdata = ftest, na.rm= TRUE)
print(cbind(Predicted = head(predFL, n=10), Actual = head(ftest$`Total Intera
ctions`, n=10)))
```

```
##    Predicted Actual
## 1   78.71163    100
## 2  358.88947    164
## 3   78.47027     80
## 4  554.62173   1777
## 5  186.87696    393
## 8  187.38021    339
## 9  374.74236    192
## 17  72.79645     54
## 18 440.88810    713
## 20  63.58059     42
```

Correlation Value

```
corFL <-cor(predFL, ftest$`Total Interactions`)
print(paste("Correlation Value: ", corFL))
```

```
## [1] "Correlation Value:  0.637964890429431"
```

MSE and RMSE Values

```
mse1 <- mean((predFL-ftest$`Total Interactions`)^2, na.rm=TRUE)
print(paste("MSE value: ", mse1))
```

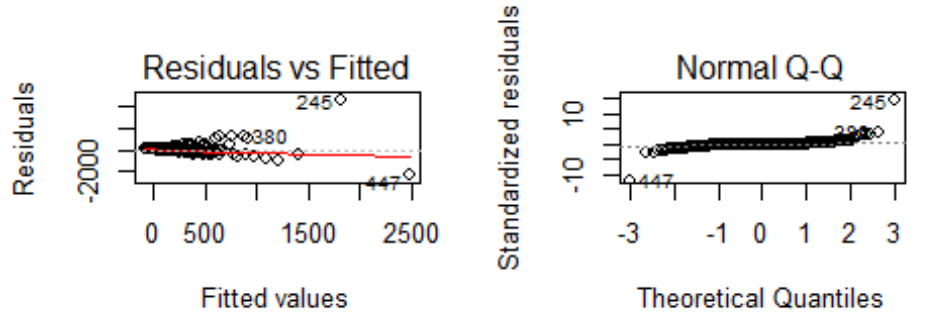
```
## [1] "MSE value:  38918.7332140609"
```

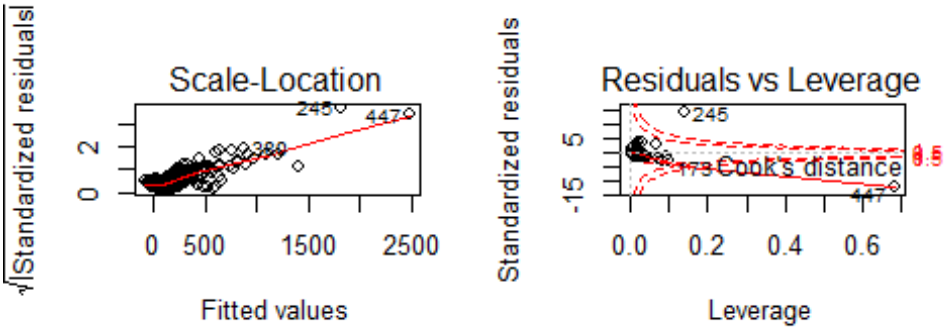
```
rmse1 <-sqrt(mse1)
print(paste("RMSE value: ", rmse1))
```

```
## [1] "RMSE value:  197.278314099804"
```

Plotting The Residuals

```
par(mfrow=c(2,2))
plot(lmf)
```





kNN Regression

First let's load in our libraries.

```
library(caret)

## Loading required package: lattice

## Loading required package: ggplot2
```

Using kNN Algorithm:

```
ftrain$`Total Interactions` <- as.integer(ftrain$`Total Interactions`)
ftest$`Total Interactions` <- as.integer(ftest$`Total Interactions`)
```

Looking For The Best 'k' Value

```
cor_k <- rep(0, 20)
mse_k <- rep(0, 20)
i <- 1
for (k in seq(1, 25)){
  fit_k <- knnreg(ftrain[,1:4],ftrain[,5], k=k)
  pred_k <- predict(fit_k, ftest[,1:4])
  cor_k[i] <- cor(pred_k, ftest$`Total Interactions`)
  mse_k[i] <- mean((pred_k - ftest$`Total Interactions`)^2)
  print(paste("k=", k, cor_k[i], mse_k[i]))
  i <- i + 1
}
```

```
## [1] "k= 1 0.27568620978021 66233.488"
## [1] "k= 2 0.320801360182282 60899.58"
## [1] "k= 3 0.294199486290544 61835.1386666667"
## [1] "k= 4 0.295301165007662 62208.7"
## [1] "k= 5 0.302616460642878 61204.80544"
## [1] "k= 6 0.293483566029209 61796.9934965986"
## [1] "k= 7 0.259436918861251 64520.2014693877"
## [1] "k= 8 0.271536434370549 62510.3552993827"
## [1] "k= 9 0.311106031184 59839.1250587654"
## [1] "k= 10 0.338540099599214 58216.1867186777"
## [1] "k= 11 0.343040455442112 57832.9763966942"
## [1] "k= 12 0.361766907548672 56895.5318576594"
## [1] "k= 13 0.365720502763106 56597.2852766574"
## [1] "k= 14 0.358857748967806 56904.3967546485"
## [1] "k= 15 0.35676851613737 57053.6036301389"
## [1] "k= 16 0.3483704467416 57510.8838122944"
## [1] "k= 17 0.337776464874238 58044.1381167927"
## [1] "k= 18 0.318769529535014 58719.7239541739"
## [1] "k= 19 0.319126884201841 58654.7721105817"
## [1] "k= 20 0.327688340716295 58076.52086"
## [1] "k= 21 0.321802739415185 58311.1301613801"
## [1] "k= 22 0.321737798734947 58298.9517844366"
## [1] "k= 23 0.310050517934438 58498.3672064167"
## [1] "k= 24 0.302291331379084 58842.8313898667"
## [1] "k= 25 0.322075034751806 57884.5318654107"
```

From the above equation we can see that the best k value is k=13.

```
#Storing correlation, mse, and rmse value for best k for later comparison.
fit_bestK <- knnreg(ftrain[,1:4], ftrain[,5] , k=13)
pred_bestK <- predict(fit_bestK, ftest[,1:4])

cor_bestK <- cor(pred_bestK, ftest$`Total Interactions`)
mse_bestK <- mean((pred_bestK - ftest$`Total Interactions`)^2)
rmse_bestK <- sqrt(mse_bestK)
```

Linear Regression vs kNN

Let's compare the two models to see which one is better for predicting how many total interactions a post will get.

First let's look at correlation for the two models.

```
print(paste("Linear Correlation = ", corFL))

## [1] "Linear Correlation =  0.637964890429431"

print(paste("kNN Correlation = ", cor_bestK))

## [1] "kNN Correlation =  0.365720502763106"
```

0.63789 > 0.3657 therefore the Linear Regression model wins this round.

Now let's look at MSE and RMSE.

```
print(paste("Linear MSE = ", mse1))
```

```
## [1] "Linear MSE =  38918.7332140609"
```

```
print(paste("Linear RMSE = ", rmse1))
```

```
## [1] "Linear RMSE =  197.278314099804"
```

```
print(paste("kNN MSE = ", mse_bestK))
```

```
## [1] "kNN MSE =  56597.2852766574"
```

```
print(paste("kNN RMSE = ", rmse_bestK))
```

```
## [1] "kNN RMSE =  237.901839582332"
```

In the case of MSE and RMSE Linear regression had the lower value.

Therefore we can conclude that between Linear Regression and kNN the best model for our data is Linear Regression.

Classification: Conventional and Social Media Movies

Data Source"
http://archive.ics.uci.edu/ml/datasets/CSM+%28Conventional+and+Social+Media+Movies%29+Dataset+2014+and+2015

Understanding The Variables

Before we begin our classification of the data let's start by looking at what our data holds.

```
#Variable Names
names(Movies_df)
```

```
##  [1] "Movie"               "Year"                "Ratings"
##  [4] "Genre"               "Gross"               "Budget"
##  [7] "Screens"             "Sequel"              "Sentiment"
## [10] "Views"               "Likes"               "Dislikes"
## [13] "Comments"            "Aggregate.Followers"
```

```
#Structure
str(Movies_df)
```

```
## 'data.frame':    194 obs. of  14 variables:
##  $ Movie              : Factor w/ 194 levels "13 Sins","22 Jump Street",..
: 1 2 3 4 5 6 7 8 9 10 ...
##  $ Year               : int  2014 2014 2014 2014 2014 2014 2014 2014 2014
2014 ...
##  $ Ratings            : num  6.3 7.1 6.2 6.3 4.7 4.6 6.1 7.1 6.5 6.1 ...
##  $ Genre              : int  8 1 1 1 8 3 8 1 10 8 ...
##  $ Gross              : int  9130 192000000 30700000 106000000 17300000 29
```

```
000 42600000 5750000 26000000 48600000 ...
##  $ Budget             : num  4.00e+06 5.00e+07 2.80e+07 1.10e+08 3.50e+06
5.00e+05 4.00e+07 2.00e+07 2.80e+07 1.25e+07 ...
##  $ Screens            : int  45 3306 2872 3470 2310 NA 3158 818 2714 2253
...
##  $ Sequel             : int  1 2 1 2 2 1 1 1 1 1 ...
##  $ Sentiment          : int  0 2 0 0 0 0 0 2 3 0 ...
##  $ Views              : int  3280543 583289 304861 452917 3145573 91137 30
13011 1854103 2213659 5218079 ...
##  $ Likes              : int  4632 3465 328 2429 12163 112 9595 2207 2210 1
1709 ...
##  $ Dislikes           : int  425 61 34 132 610 7 419 197 419 532 ...
##  $ Comments           : int  636 186 47 590 1082 1 1020 593 382 770 ...
##  $ Aggregate.Followers: int  1120000 12350000 483000 568000 1923800 310000
8153000 130655 125646 21697300 ...
```

Looking at the data I would like to predict the Genre based on Rating, Gross, Budget, Likes, Dislikes, and Comments. However first let's change Genre into a factor for classification and look at it's levels.

```
Movies_df$Genre <- factor(Movies_df$Genre)
levels(Movies_df$Genre)

##  [1] "1"  "2"  "3"  "6"  "7"  "8"  "9"  "10" "12" "15"
```

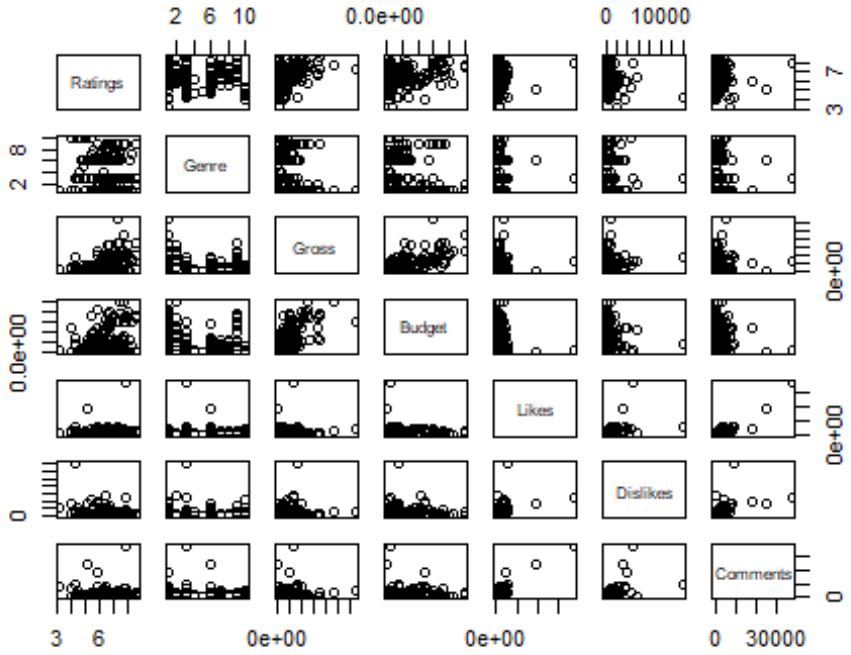
Now let's create a new dataframe to store all of the variables that we will be looking at.

```
newMovies <- subset(Movies_df, select = c("Ratings", "Genre", "Gross", "Budge
t", "Likes", "Dislikes", "Comments"))
```

pairs() and cor()

Let's use the pairs() and cor() funtion to look at correlations.

```
pairs(newMovies)
```



To break down our variables. Our target is Genre which represents the genre of the movie. In our dataset labels were not given so our genres are: 1,2,3,6,7,8,9,10, 12, and 15.

Now to break down the predictors. Rating is the critic rating of the movie. Gross is the gross profit of the movie (gross = profit - budget). The Budget is how much money was spent to make the movie. Likes are the number of likes on Youtube, dislikes are the number of dislikes on Youtube and Comments are the number of comments on Youtube.

```
summary(newMovies)
```

```
##     Ratings          Genre        Gross                Budget
##  Min.   :3.100   1      :56   Min.   :     2470   Min.   :    70000
##  1st Qu.:5.800   8      :46   1st Qu.: 14150000   1st Qu.: 11250000
##  Median :6.400   3      :40   Median : 43850000   Median : 29500000
##  Mean   :6.399   12     :12   Mean   : 74622822   Mean   : 52112875
##  3rd Qu.:7.100   2      :11   3rd Qu.: 92750000   3rd Qu.: 69500000
##  Max.   :8.700   9      :10   Max.   :643000000   Max.   :250000000
##                  (Other):19
##      Likes            Dislikes          Comments
##  Min.   :     1   Min.   :    0.0   Min.   :    0.0
##  1st Qu.:  2090   1st Qu.:  123.2   1st Qu.:  267.5
##  Median :  6562   Median :  362.0   Median :  876.0
##  Mean   : 13597   Mean   :  715.0   Mean   : 1913.5
##  3rd Qu.: 15900   3rd Qu.:  750.8   3rd Qu.: 2102.8
##  Max.   :370552   Max.   :13960.0   Max.   :38363.0
##
```

Logistic Regression Model

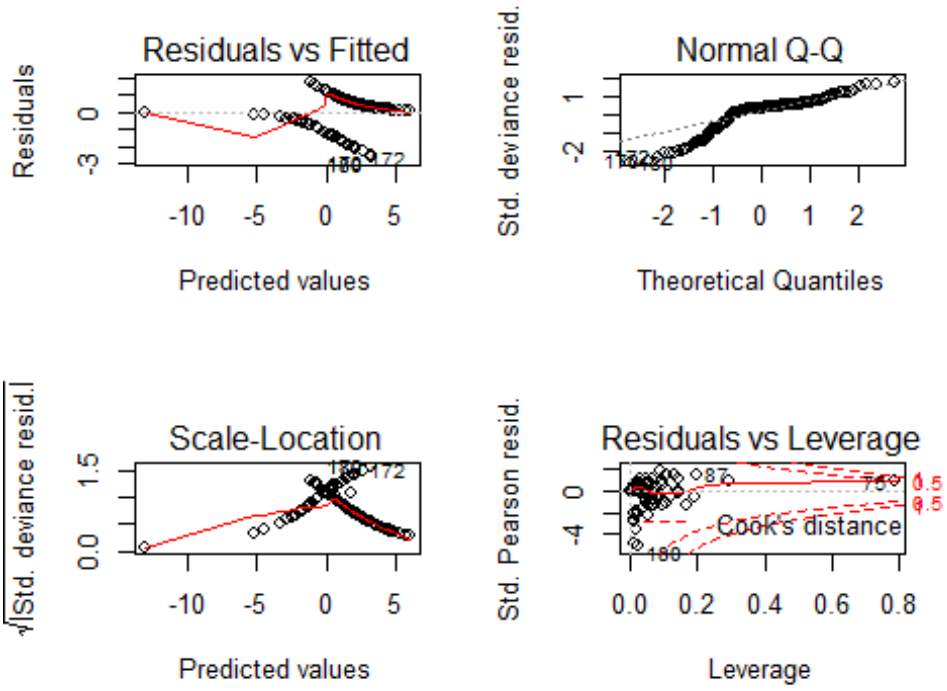
Now that we have explored the variables let's use a logistic regression model to see if we can classify our data into the correct genre.

```
#Setting seed and randomly sampling
set.seed(0001)
i <- sample(1:nrow(newFacebook), nrow(newFacebook)*0.75, replace=FALSE)

#Separating into train and test data
mtrain <- newMovies[i,]
mtest <- newMovies[-i,]

#Creating linear Model
glmm <- glm(Genre~. , data=mtrain, family="binomial")

#Plotting to look at graph
par(mfrow=c(2,2))
plot(glmm)
```




```
#Summary/Metrics of the linear regression model
summary(glmm)

##
## Call:
## glm(formula = Genre ~ ., family = "binomial", data = mtrain)
##
```

```
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.5528  -0.1256   0.3918   0.5988   1.6679
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -2.611e-01  1.711e+00  -0.153  0.87875
## Ratings      4.164e-01  2.791e-01   1.492  0.13571
## Gross       -4.547e-09  3.470e-09  -1.310  0.19010
## Budget      -1.627e-08  5.583e-09  -2.915  0.00356 **
## Likes        1.444e-04  4.772e-05   3.027  0.00247 **
## Dislikes     6.440e-04  5.404e-04   1.192  0.23344
## Comments    -1.170e-03  3.574e-04  -3.273  0.00106 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 179.10  on 154  degrees of freedom
## Residual deviance: 119.82  on 148  degrees of freedom
##   (219 observations deleted due to missingness)
## AIC: 133.82
##
## Number of Fisher Scoring iterations: 6
```

Looking at our summary we can see that our best predictors for genre are Budget, Likes, and Comments. That being said, although they are our best predictors the two stars indicate that they are only okay predictors and non-necessarily good predictors.

Testing The Model

Now let's test our model by checking the accuracy.

```
prob <- predict(glmm, newdata = mtest, type="response")
predML <- ifelse(prob>0.5, 1, 0)
acc <- mean(predML==mtest$Genre)
print(paste("accuracy = ", acc))
```

```
## [1] "accuracy =  0.256410256410256"
```

```
#Test Genre Values
mtest$Genre
```

```
##  [1] 8  1  1  1  8  1  10 8  3  3  1  3  1  12 2  1  1  8  8  12 1  12 8
## [24] 3  2  3  15 8  9  3  1  3  1  1  1  1  1  8  8
## Levels: 1 2 3 6 7 8 9 10 12 15
```

```
#Accuracy Table
table(predML, mtest$Genre)
```

```
##
## predML  1  2  3  6  7  8  9 10 12 15
##      0  5  2  0  0  0  0  0  0  0  0
##      1 10  0  7  0  0  9  1  1  3  1
```

We can see that with the logistic regression model we have an accuracy of 0.25641% which means that our predictors and our model do not do that great of a job identifying the genre of a movie.

kNN Classification

Since our logistic regression model didn't produce the best accuracy why don't we try using the kNN classification instead.

```
trainVector <-c(1,3,4,5,6,7)
#Set Seed
set.seed(1111)
ind <- sample(2, nrow(newMovies), replace=TRUE, prob=c(0.75, 0.25))
train_MK <- newMovies[ind==1, trainVector]
test_MK <- newMovies[ind==2, trainVector]
trainLabels <- newMovies[ind==1, 2]
testLabels <- newMovies[ind==2, 2]
```

Now let's classify and compute the accuracy.

```
library(class)

pred_MK <- knn(train=train_MK, test=test_MK, cl=trainLabels, k= 7)

results <- pred_MK == testLabels
acc2 <- length(which(results==TRUE)) / length(results)
print(paste("accuracy = ", acc2))

## [1] "accuracy =  0.352941176470588"
```

Logistic Regression vs kNN Classification

In order to compare the two models lets look at the most important the accuracy.

```
print(paste("Logistic Accuracy = ", acc))

## [1] "Logistic Accuracy =  0.256410256410256"

print(paste("kNN Accuracy = ", acc2))

## [1] "kNN Accuracy =  0.352941176470588"
```

Looking at our data we found the kNN Classification to be more accurate and thus the better algorithm for predicting the genre of the movie.