# IE7275 DATA MINING CASE STUDY REPORT

***Hem Bhupaal Reddy***

**Executive Summary:**

Spotify is a digital music service that enables users to remotely source millions of different songs and podcasts on various record labels from a laptop, smartphone, or another device. Spotify also assigns each song a popularity score, based on the total number of clicks/listens. Predicting the popularity of the song should be a challenge and interesting topic in nowadays music field. Songs characteristic could be separated into 12 factors, which could be used for predicting the popularity of a specific song. For instance, songs' loudness or liveness could be an important effective factor for the popularity of the new song. The product manager or artist could use the factors to compose a song that fit the popular trend in that year. In this case, we proposed a solution that used the data mining techs, such as (K-Nearest Neighbor)KNN, Multilinear Regression, Random Forest, Support Vector Machines, as well as Neural Net, to build a model to predict the popularity of a song. Meanwhile, a dimension reduction method, Principal Component Analysis (PCA), had been implemented for comparing the result from the original dataset. The model is estimated via accuracy and robustness using Lift Charts and RMSE. The term project aims to build a model to predict the popularity of a song based on its kernel elements.

1. **Background and Introduction**

**Background**

The global record music industry is worth $21.5 billion US Dollars in 2019 with an annual growth rate of 5.4% and the greatest share of the revenue is generated by the streaming services. With 286 million users and 144 million premium subscribers in Q3 of 2020, it has become one of the most ideal platforms for artists to reach their audience. At the heart of Spotify lives a massive and growing dataset. We thought of analyzing the most popular songs using Data Mining techniques. The content on Spotify, of course, doesn’t come for free but many in-app purchases bring in the revenue for the app. On average a customer shells out $10 for a monthly subscription to the app. For this amount, the customer’s expectation with the app is also high. Spotify invests a chunk of its I revenue in R&D to bring the best service to the table for the customer. Customer also expects better service in term of the content.

**The Problem**

Predicting the popularity of a song based on its characteristics could be interesting and challenge in the modern music field. Song Writers and Product Managers could use the model to predict the popularity of a new song based on its features, for instance, the acoustics could be a significant factor that affects the popularity of the song in 2018; therefore, the artists could add or reduce the acoustic elements when he or she composed the songs.

**The Goal of the study**

Building a model to successfully predict the popularity of a song based on its specific factors. Consumers can use the power of analytics to segregate songs into broader genres or categories based on the popularity or recommendations which may not belong to the same genre but have similar features that can be used to make custom playlists for the customer in his/her local machine. Recommendations for new songs from different genres/languages which have similar features such as danceability, energy, valence, etc can be made to a user depending on their listening history and preferences.

**Our Solution**

In this case, we proposed a solution that used the data mining techs, such as (K-Nearest Neighbor)KNN, Multilinear Regression, Random Forest, Support Vector Machines, as well as Neural Net, to build a model to predict the popularity of a song. Meanwhile, a dimension reduction method, Principal Component Analysis (PCA), had been implemented for comparing the result from the original dataset. The model is estimated via accuracy and robustness using Lift Charts and RMSE.

1. **Data Exploration and Visualization**

**Data Description**

The dataset was gained from Kaggle website, named “*Spotify Dataset 1921-2020, 160k+ Tracks.*” The dataset file contains more than 170.000 songs collected from *Spotify Web API*, and also the songs were grouped by artist, year, or genre in the data section. The datasets was updated recently by *Yamaç Eren* Ay, the last updated time is 2020- 11-25. In this case, we select the year 2018 dataset as our case study analyzing data source.

*Table 2.1 Data Description*

**Numerical Variables:**

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Value Type** | **Description** |
| Popularity | int | Measuring the popularity of the song, Output varibale Ranges from 0 to 100 |
| Acousticness | float | A confidence measure ranging from 0.0 to 1.0 of whether the track is acoustic |
| Danceability | float | Danceability describes how suitable a track is for dancing based on a combination of musical elements including tempo, rhythm stability, beat strength, and overall regularity. Ranges from 0 to 1 |
| Duration\_ms | int | Duration of the track in milliseconds. ranging from 200k to 300k |
| Energy | float | Energy is a measure ranging from 0.0 to 1.0 and represents a perceptual measure of intensity and activity. Typically, energetic tracks feel fast, loud, and noisy. |
| Instrumentalness | float | Predicts whether a track contains no vocals. “Ooh” and “aah” sounds are treated as instrumental in this context. The closer the instrumentalness value is to 1.0, the greater likelihood the track contains no vocal content. Ranges from 0 to 1 |
| Liveness | float | Detects the presence of an audience in the recording. Higher liveness values represent an increased probability that the track was performed live. Ranges from 0 to 1 |
| Loudness |  | The overall loudness of a track in decibels (dB) ranging from -60 to 0 |
| Speechiness | float | Speechiness detects the presence of spoken words in a track. The more exclusively speech-like the recording (e.g. talk show, audio book, poetry), the closer to 1.0 the attribute value. Ranges from 0 to 1 |
| Tempo | float | The overall estimated tempo of a track in beats per minute (BPM). In musical terminology, tempo is the speed or pace of a given piece and derives directly from the average beat duration. ranging from 50 to 150 |
| Valence | float | A measure ranging from 0.0 to 1.0 describing the musical positiveness conveyed by a track. Tracks with high valence sound more positive, while tracks with low valence sound more negative. Ranges from 0 to 1 |
| Year | int | Year in which song was released. Ranges from 1921 to 2020. |

**Dummy Variables:**

|  |  |  |
| --- | --- | --- |
| Mode | int | Mode indicates the modality (major or minor) of a track, the type of scale from which its melodic content is derived. 0 = Minor, 1 = Major |

**Categorical Variable:**

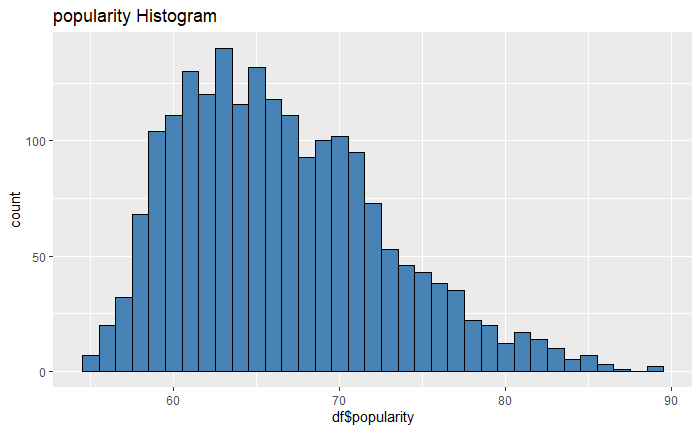
|  |  |  |
| --- | --- | --- |
| Key | int | The estimated overall key of the track. ranging from 0 to 11 |

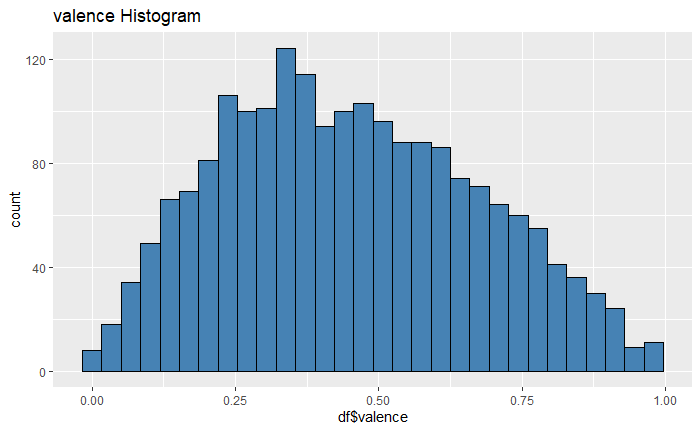
**Data Visualization**

A function *describe()* had been used for find the distributions for helping us visualize the data roughly.

*Table 2.2 Distribution of the Spotify Dataset 2018*

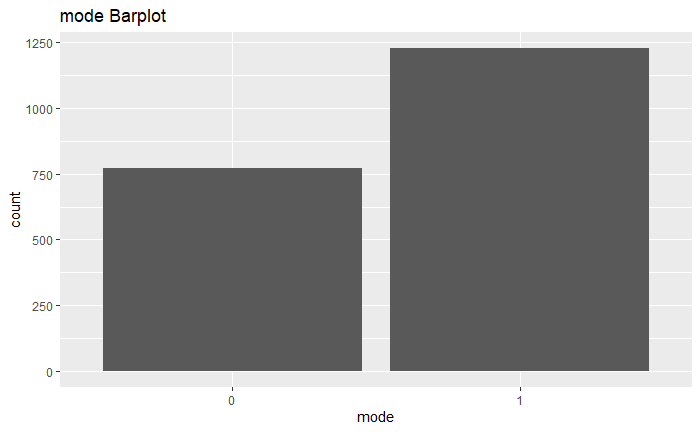




*Figure 2.1 Histogram of the output variable – Popularity*

*Figure 2.2 Histogram of input variable - Valence*

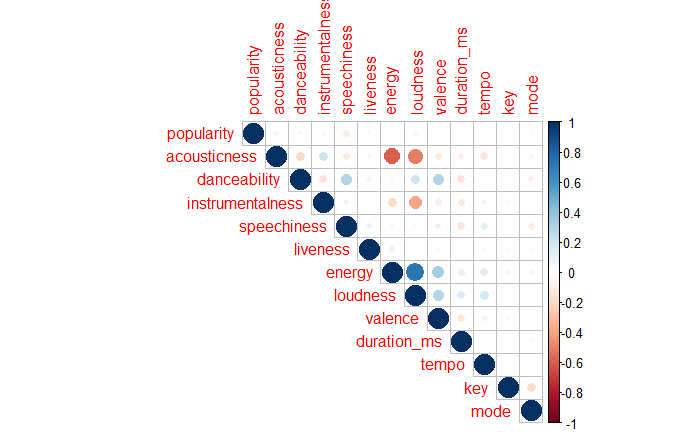
From the histogram of the popularity we can see the popularity of the dataset is mostly concentrated between 60 – 70. Other histogram plot had been checked in Appendix code.



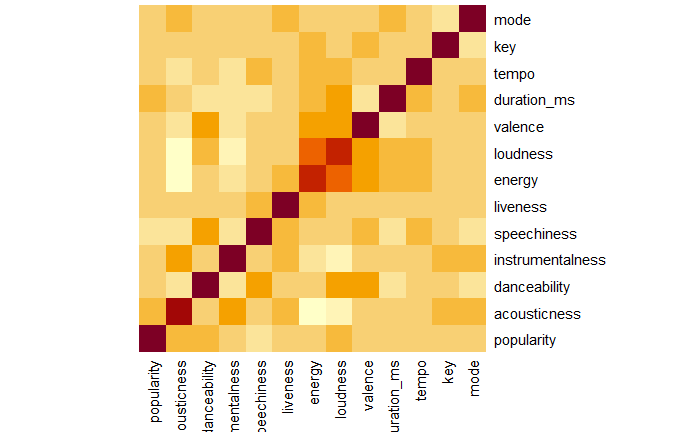
*Figure 2.3 Barplot of Mode*

From the above plots we can see that basically the variables are followed the normal distribution. However, the instrumentalness and liveness have the skewed distribution(left). Which means the mode value < median < mean. It may affect the model results. Therefore, the normalization and standardization is necessary for the input dataset.

Moreover, The correlation matrix also had been used for analyzing the correlation between each variables:



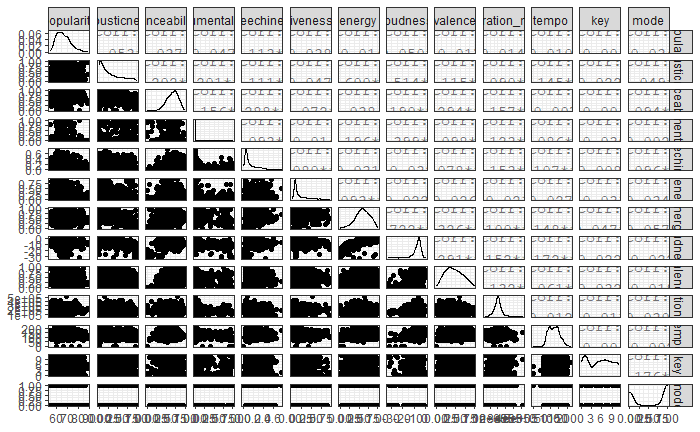
*Figure 2.4 Correlation Matrix of the Variables*



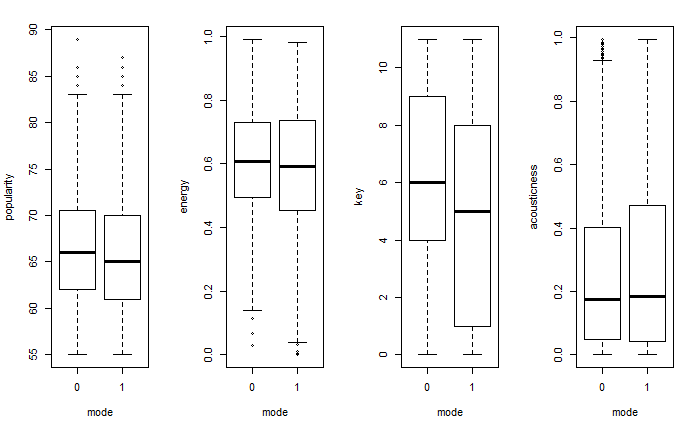
*Figure 2.5 Heatmap of all variables*

From the above correlation matrix and heatmap, the acousticness have strong negative correlation to energy and loudness; meanwhile, the energy had strong positive correlation to loudness. This is easy to understand since the song that have large acousticness is basically the classic or light music type, which would not have strong energy and loudness.

Furthermore, a scatter matrix had been plotted for analyzing the kernel density estimation and correlation double-checking.



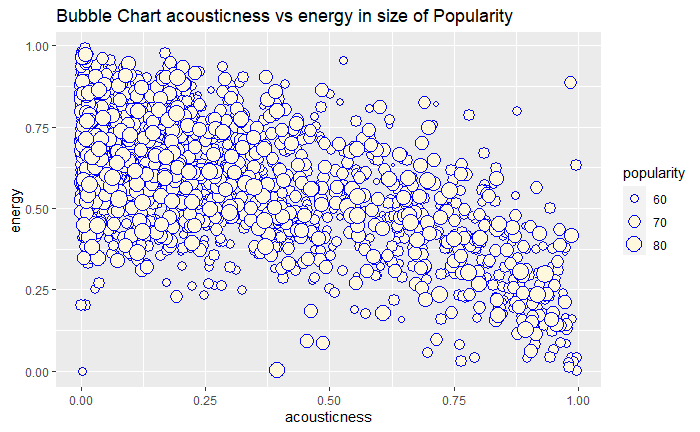
*Figure 2.6 Scatter Matrix for all variables*



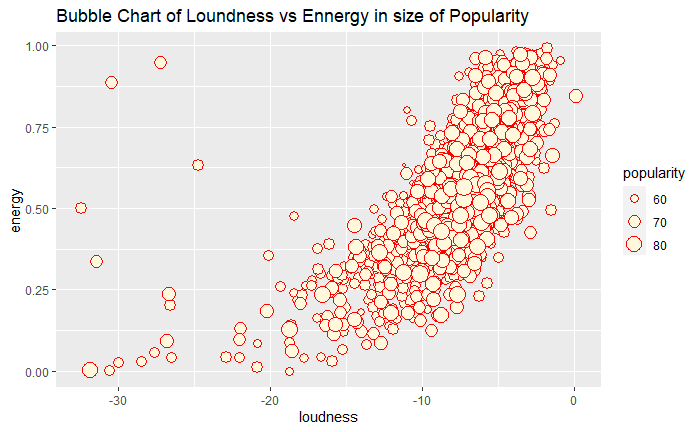
*Figure 2.7 Boxplot for mode*

From Figure 2.7, we also use the boxplots to see the distribution for the dummy variable mode. Also, from the appendix code, the boxplot for key factor had been plotted. The results show that the dummy variable basically had some right-skewed distribution for the popularity probability density.

Last, the bubble charts in size of popularity had been used for analyzing the distribution of popularity in different combination of variables:



*Figure 2.8 Bubble Chart 1*



*Figure 2.9 Bubble Chart 2*

1. **Data Preparation and Preprocessing**

**Data Summery**

The dataset that we selected for the case study has 12 numerical attributes with one categorical attribute. The Popularity column, a numerical variable range from 1 to 100, was select as the response variable. All the other variables were used for input variables.

**Variable Selection**

Since our dataset did not contain many correlated attributes, we select 12 factors as the input variables, except the variable explicit. The reason why we did not select this variable is that from developer.spotify.com we cannot find the exact definition of the factor. Therefore, Popularity was selected as the output variable, the input variables contain acousticness, danceability, instrumentalness, speechiness, liveness, energy, loudness, valence, duration\_ms, tempo, key, mode.

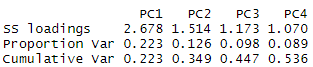
**Preprocessing and Dimensionality Reduction**

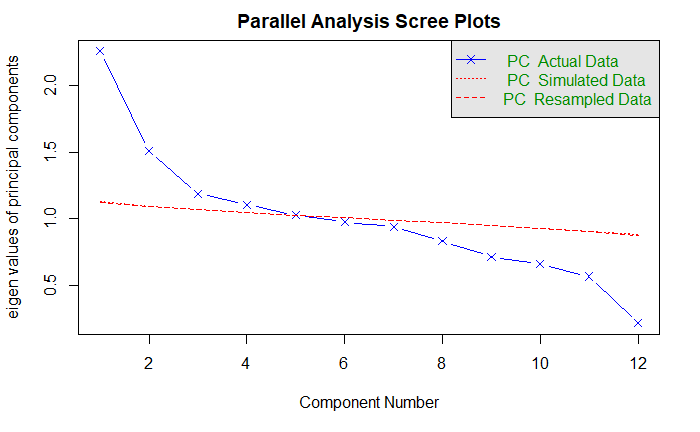
*Data Clean Up:*

We clean up the dataset by analyzing the histogram and box plots. The original dataset’s popularity column has few data points that the popularity is equal to 0, which is much far away from the mean value from the histogram plot and box plot. It means that these points are not the real situation since the popularity of the song could not be 0 (just a thought, maybe the 0 means the song is not been published.) Therefore, we deleted these data points and make the dataset into 2000 rows and 13 columns.

After that, we also used PCA analysis to reduce the dimension into 4 PCA factors, which captured 53.6% percent variance. The number of component (4) were selected via parallel analysis and scree plot.

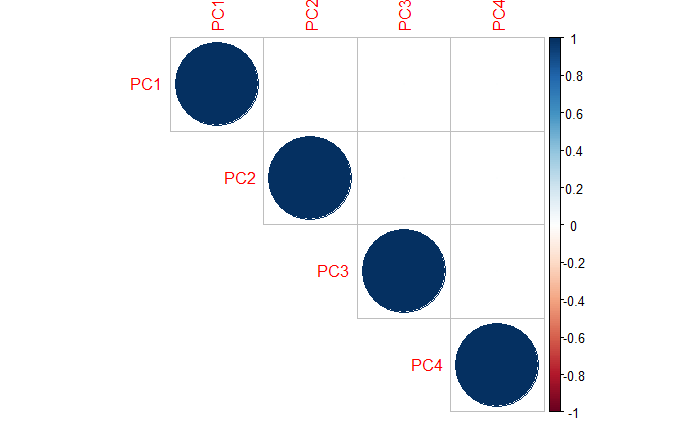
*Table 3.1 PCA cumulative variance*





*Figure 3.1 Parallel Analysis and Scree Plot*

We also analyze the correlation of each PCA factors:



*Figure 3.2 Correlation Plot of PCA dataset*

We separate the dataset into 60% for training(1200rows), and 40% for validation(800rows).

The normalized and standardize function had been defined and used in original dataset and PCA dataset for different models.

1. **Data Mining Tech and Implementation**

We then use the following five algorithms on for the prediction problem:

1. K Nearest Neighbors (KNN)

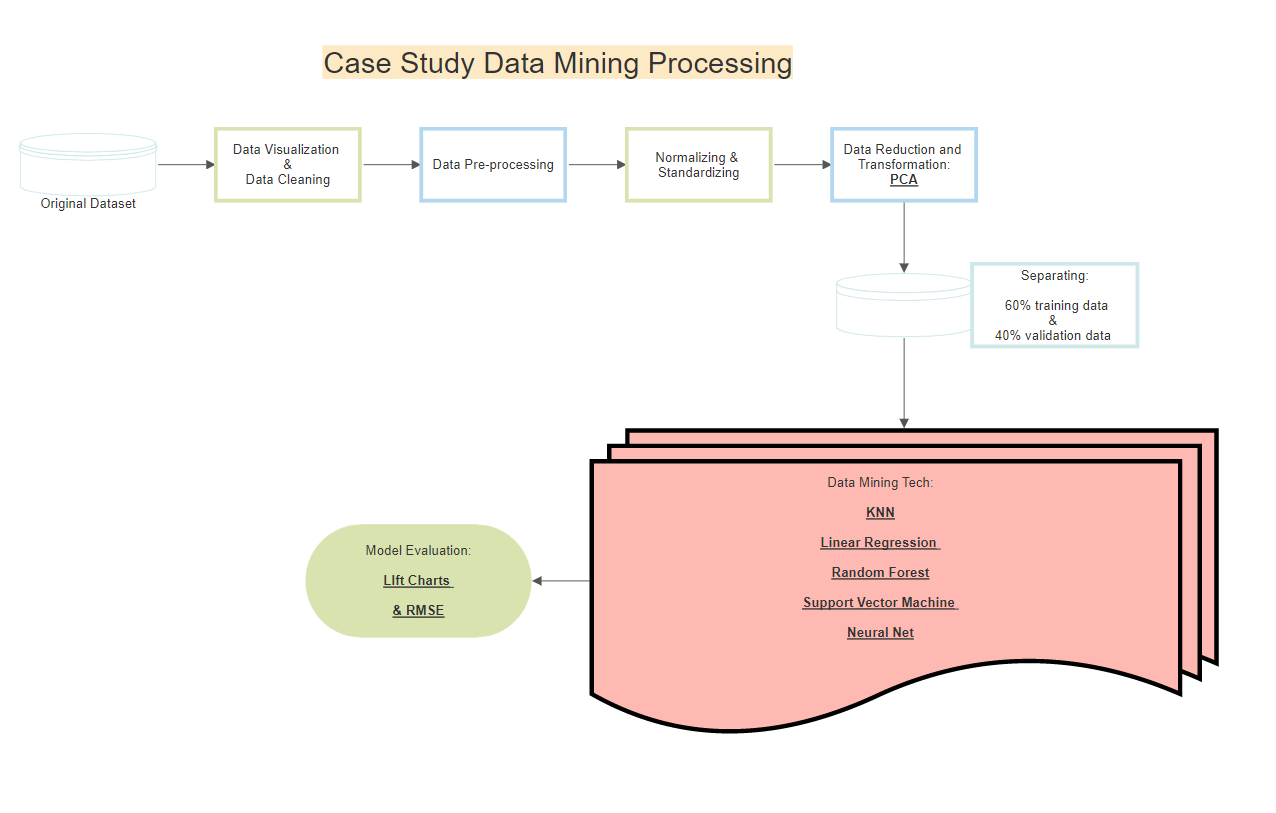
2. Multilinear Regression

3. Random Forests

4. Support Vector Machine

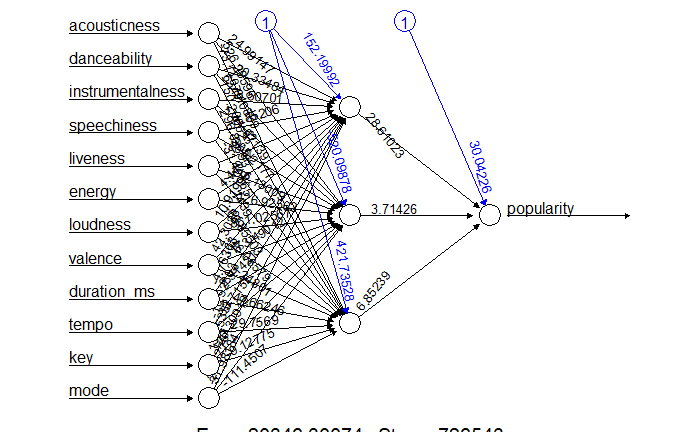
5. Artificial Neural Network

*Flow Chart of Modeling Process*

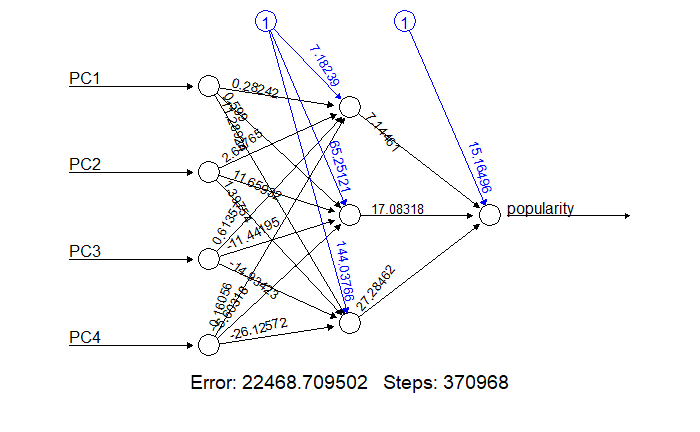


1. **Performance Evaluation**

*Neural Network Plots*



*Figure 5.1 Neural Net Plot with Original Dataset*



*Figure 5.2 Neural Net Plot with PCA dataset*

The above plots are the neural network architectures that had the best performance in the case.

*Lift Charts*

**Hint:** the legend for the plots

*Blue Line = KNN model*

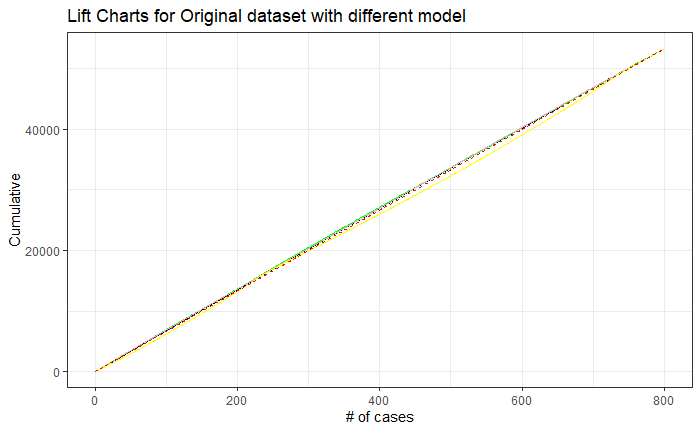
*Black Line = Linear Regression model*

*Green Line = Random Forest model*

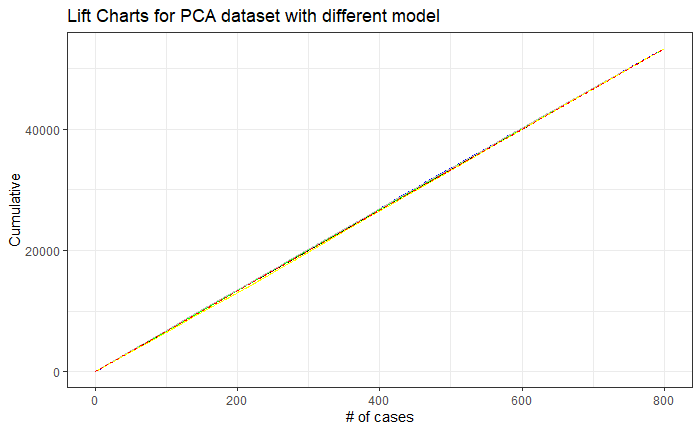
*Grey Line = Support Vector Machines model*

*Yellow Line = Neural Net model*

*Dashed red line = base line*



*Figure 5.3 Lift Charts for Original Datasets with different models*



*Figure 5.3 Lift Charts for PCA Datasets with different models*

From the above Lift Charts, we can see the model results are not so well (just a little lift). We also check the accuracy for each model, have similar results:

*Table 5.1 Model Accuracy and Correlation Coefficient*



From the above table, we can see that the random forest has the best performance for the case with the smallest Root Mean Square Error (RMSE) and the largest Correlation Coefficient. The result is the same as the lift charts result.

1. **Discussion and Recommendation**

From the model results, the Random Forest performed best in the case since it has the smallest RMSE for the prediction model. Comparatively, the Neural Net and SVM performed not so well in this case. The reason this phenomenon happens is the poor feature selection. Since our group members are not professional musicians or songwriters. We are not familiar with the musical domain knowledge, the features that we select for this case could not be enough to predict the popularity of the songs.

Moreover, from the Lift Charts results, we could see all the model performance is not so well in this case. The reason here could be the factor selection is not as good as we thought. The factors like loudness, liveness, etc., not enough for predicting the song’s popularity. The poor performance for each model could be fixed and improved by adding more reasonable factors, for instance, the popularity of the composers’ as well as the singers’ reputation could be a significant effect of the popularity of a song. After optimizing the input variables, the model performance could be different.

For further analysis, besides operating the input variables, another thing that needs to do is to change the popularity from a numerical variable into a dummy variable, like setting up a cutoff value of 66(mean of the popularity) for separating the song is popular or not. And using that as a factor to make the model for a recommendation songs model.

1. **Summary**

In this case study, different data-mining tech had been implemented and validated for predicting the songs’ popularity based on Spotify Dataset 2018. The datasets had been separated into training and validation datasets, and necessary normalization and standardization had been used for improving the model performance. The Random Forest model had the best performance for this case.

**Appendix: R code for the Case Study**

---

title: "Final Project - Group3"

author: "Siyu"

date: "2020/12/5"

output: word\_document

---

```{r setup, include=FALSE}

# Case Study

library(tidyverse)

library(psych)

library(caret)

library(FNN)

library(ISLR)

library(tree)

library(randomForest)

library(neuralnet)

library(ROCR)

library(e1071)

library(gains)

library(ggplot2)

library(reshape2)

library(rpart)

library(rpart.plot)

library(corrplot)

library(ggthemes)

library(dplyr)

library(scales)

library(GGally)

library(car)

library(gvlma)

library(MASS)

library(leaps)

library(bootstrap)

library(gmodels)

library(caret)

library(forecast)

```

## R Markdown

```{r }

# Load Project\_data2018.csv

setwd("C:/Users/msy92/Desktop/IE 7275 HW")

df <-data.frame(read.csv("Project\_data2018.csv"))

describe(df)

str(df)

write.csv(describe(df), "decrib\_2018.csv")

```

################################################################ Data Visualization

```{r }

# Let's look at Popularity Histogram

ggplot(df, aes(df$popularity)) +geom\_histogram(binwidth = 1, color = "black", fill = "steelblue") +ggtitle("popularity Histogram")

# Bar plot of mode

ggplot(df, aes(as.factor(df$mode), fill = df$popularity)) +geom\_histogram(stat = "count") +ggtitle("mode Barplot") +labs(x = "mode")

#his of input varibale

par(mfcol = c(3,3))

ggplot(df, aes(df$acousticness)) +geom\_histogram( color = "black", fill = "steelblue") +ggtitle("acousticness Histogram")

ggplot(df, aes(df$danceability)) +geom\_histogram( color = "black", fill = "steelblue") +ggtitle("danceability Histogram")

ggplot(df, aes(df$instrumentalness)) +geom\_histogram( color = "black", fill = "steelblue") +ggtitle("instrumentalness Histogram")

ggplot(df, aes(df$speechiness)) +geom\_histogram( color = "black", fill = "steelblue") +ggtitle("speechiness Histogram")

ggplot(df, aes(df$liveness)) +geom\_histogram( color = "black", fill = "steelblue") +ggtitle("liveness Histogram")

ggplot(df, aes(df$energy)) +geom\_histogram( color = "black", fill = "steelblue") +ggtitle("energy Histogram")

ggplot(df, aes(df$loudness)) +geom\_histogram( color = "black", fill = "steelblue") +ggtitle("loudness Histogram")

ggplot(df, aes(df$valence)) +geom\_histogram( color = "black", fill = "steelblue") +ggtitle("valence Histogram")

ggplot(df, aes(df$tempo)) +geom\_histogram( color = "black", fill = "steelblue") +ggtitle("tempo Histogram")

#boxplot of mode & key factors

par(mfcol = c(1,4))

boxplot(df$popularity ~ df$mode, xlab = "mode", ylab = "popularity")

boxplot(df$energy ~ df$mode, xlab = "mode", ylab = "energy")

boxplot(df$key ~ df$mode, xlab = "mode", ylab = "key")

boxplot(df$acousticness ~ df$mode, xlab = "mode", ylab = "acousticness")

par(mfcol = c(1,4))

boxplot(df$popularity ~ df$key, xlab = "key", ylab = "popularity")

boxplot(df$energy ~ df$key, xlab = "key", ylab = "energy")

boxplot(df$liveness ~ df$key, xlab = "key", ylab = "liveness")

boxplot(df$tempo ~ df$key, xlab = "key", ylab = "tempo")

#heatmap

par(mfcol = c(1,1))

heatmap(cor(df),Rowv = NA, Colv = NA)

cor.mat <- cor(df)

melt.cor.mat <- melt(cor.mat)

melt.cor.mat

# corrplot

cor <-round(cor(df[,1:13]),2)

write.csv(cor, "correlation\_df.csv")

corrplot(cor, type = "upper")

#generate scatter matrix

ggpairs(df)+ theme\_bw()

#scatterplotMatrix(df, spread = FALSE, lty.smooth = 2, main = "Scatter Plot Matrix" )

#bubble charts of popularity

ggplot(df, aes( x = acousticness ,y = energy, size = popularity)) + geom\_point(shape =21, color = "blue", fill = "cornsilk") + labs(x ="acousticness", y = "energy", size = "popularity", title = "Bubble Chart acousticness vs energy in size of Popularity")

ggplot(df, aes( x = loudness ,y = energy, size = popularity)) + geom\_point(shape =21, color = "red", fill = "cornsilk") + labs(x ="loudness", y = "energy", size = "popularity", title = "Bubble Chart of Loundness vs Ennergy in size of Popularity")

```

################################################################ Data Pre-processing

```{r }

#Define the normalize function& standardize funciton

normalize <-function(x) {

return(((x -min(x))) / (max(x) -min(x)))

}

standardize <- function(x){

return((x-mean(x))/sd(x))

}

# standardize the data frame

df.norm <-as.data.frame(cbind(as.data.frame(lapply(df[2:13], standardize)),df$popularity)) %>%rename(popularity = "df$popularity")

df.norm

```

################################################################ Data Reduction and Transformation

```{r }

#Performing PCA on the data

# Perform Scree Plot and Parallel Analysis

fa.parallel(df.norm[, 2:13], fa = "pc", n.iter = 100)

# Perform PCA with 4 components

#extract the components and rotate the component

pc <-principal(df[, 2:13], nfactors = 4, rotate = "none", scores = TRUE)

pc\_var <-principal(df.norm[, 2:13], nfactors = 4, rotate = "varimax", scores = TRUE)

#compute the components score

head(pc$scores)

pc$loadings

pc <-cbind(as.data.frame(pc$scores), df.norm$popularity) %>%rename(popularity = "df.norm$popularity")

pc

# corrplot of pca data

cor(pc)

cor <-cor(pc[,1:4])

corrplot(cor, type = "upper")

```

#########################Splitting data into training and validation sets

```{r }

#default seperation of traning and validate datasets

set.seed(111)

indices.df <- sample(c(1:dim(df)[1]), dim(df)[1]\*0.6)

train.df <- df[indices.df, ]

valid.df <- df[-indices.df, ]

head(train.df)

head(valid.df)

# Generate the training and validation datasets without PCA

set.seed(111)

indices.dfnorm <- sample(c(1:dim(df.norm)[1]), dim(df.norm)[1]\*0.6)

train.dfnorm <- df.norm[indices.dfnorm, ]

valid.dfnorm <- df.norm[-indices.dfnorm, ]

head(train.dfnorm)

head(valid.dfnorm)

# Generate training and validation datasetes with PCA

set.seed(111)

indices.pc <- sample(c(1:dim(pc)[1]), dim(pc)[1]\*0.6)

train.pc <-pc[indices.pc, ]

valid.pc <-pc[-indices.pc, ]

head(train.pc)

head(valid.pc)

```

################################################################ Data Mining Techniques

######################### Implementing KNN

```{r }

# compute knn for different k on validation without PCA

accuracy.df <- data.frame(k = seq(1, 50, 1), RMSE = rep(0, 50))

for(i in 1:50) {

song.knn <- knnreg(train.dfnorm[, 1:12],train.dfnorm$popularity, k = i)

song.knn.pred <- predict(song.knn, valid.dfnorm[,1:12])

accuracy.df[i, 2] <- data.frame(accuracy(valid.dfnorm$popularity, song.knn.pred))$RMSE

}

accuracy.df

#k =21 have the min RMSE

song.knn <- knnreg(train.dfnorm[, 1:12],train.dfnorm$popularity, k = 21)

song.knn.pred <- predict(song.knn, valid.dfnorm[,1:12])

accuracy(valid.dfnorm$popularity, song.knn.pred)

cor(valid.dfnorm$popularity,song.knn.pred)

#Lift Chart plot

gain.knn <- gains(valid.dfnorm$popularity,song.knn.pred, groups = length(song.knn.pred))

plot(c(0, gain.knn$cume.pct.of.total\*sum(valid.dfnorm$popularity))~c(0, gain.knn$cume.obs), xlab = "# cases", ylab = "Cumulative", main = "Lift Chart of KNN(Original Dataset)", type = "l")

lines(c(0, sum(valid.dfnorm$popularity))~c(0, dim(valid.dfnorm)[1]), lty = 2)

# compute knn for different k on validation with PCA

accuracy.pc <- data.frame(k = seq(1, 50, 1), RMSE = rep(0, 50))

for(i in 1:50) {

pc.knn <- knnreg(train.pc[, 1:4],train.pc$popularity, k = i)

pc.knn.pred <- predict(pc.knn, valid.pc[,1:4])

accuracy.pc[i, 2] <- data.frame(accuracy(valid.pc$popularity, pc.knn.pred))$RMSE

}

accuracy.pc

#k =47 have the min RMSE

pc.knn <- knnreg(train.pc[, 1:4],train.pc$popularity, k = 47)

pc.knn.pred <- predict(pc.knn, valid.pc[,1:4])

accuracy(valid.pc$popularity, pc.knn.pred)

cor(valid.pc$popularity,pc.knn.pred)

#Lift Chart plot

gain.knn <- gains(valid.dfnorm$popularity,pc.knn.pred, groups = length(pc.knn.pred))

plot(c(0, gain.knn$cume.pct.of.total\*sum(valid.dfnorm$popularity))~c(0, gain.knn$cume.obs), xlab = "# cases", ylab = "Cumulative", main = "Lift Chart of KNN(PCA)", type = "l")

lines(c(0, sum(valid.dfnorm$popularity))~c(0, dim(valid.dfnorm)[1]), lty = 2)

```

######################### Implementing linear Regression

```{r }

# On Original Dataset

song.lm <- lm(popularity~., train.dfnorm)

muti1.song.lm<-lm(popularity~ acousticness+danceability+instrumentalness+speechiness+loudness, train.dfnorm)

muti2.song.lm<-lm(popularity~ acousticness+danceability+instrumentalness+speechiness+loudness+liveness+mode, train.dfnorm)

#

song.lm.pred <- predict(song.lm, valid.dfnorm)

muti1.song.lm.pred <- predict(muti1.song.lm, valid.dfnorm)

muti2.song.lm.pred <- predict(muti2.song.lm, valid.dfnorm)

#

accuracy(valid.dfnorm$popularity, song.lm.pred)

accuracy(valid.dfnorm$popularity, muti1.song.lm.pred)

accuracy(valid.dfnorm$popularity, muti2.song.lm.pred)

#

cor(valid.dfnorm$popularity, song.lm.pred)

cor(valid.dfnorm$popularity, muti1.song.lm.pred)

cor(valid.dfnorm$popularity, muti2.song.lm.pred)

#

#Lift Chart plot

gain.lm <- gains(valid.dfnorm$popularity,song.lm.pred, groups = length(song.lm.pred))

plot(c(0, gain.lm$cume.pct.of.total\*sum(valid.dfnorm$popularity))~c(0, gain.lm$cume.obs), xlab = "# cases", ylab = "Cumulative", main = "Lift Chart of Linear Regression(Original Dataset)", type = "l")

lines(c(0, sum(valid.dfnorm$popularity))~c(0, dim(valid.dfnorm)[1]), lty = 2)

# On PCA Dataset

pc.lm <- lm(popularity~., train.pc)

#

pc.lm.pred <- predict(pc.lm, valid.pc)

#

accuracy(valid.pc$popularity, pc.lm.pred)

#

cor(valid.pc$popularity, pc.lm.pred)

#Lift Chart plot

gain.lm <- gains(valid.dfnorm$popularity,pc.lm.pred, groups = length(pc.lm.pred))

plot(c(0, gain.lm$cume.pct.of.total\*sum(valid.dfnorm$popularity))~c(0, gain.lm$cume.obs), xlab = "# cases", ylab = "Cumulative", main = "Lift Chart of Linear Regression(PCA)", type = "l")

lines(c(0, sum(valid.dfnorm$popularity))~c(0, dim(valid.dfnorm)[1]), lty = 2)

```

######################### Implementing Random Forests Tech

```{r }

# On original dataset

set.seed(111)

rf.df <-randomForest(popularity~ ., data = train.dfnorm,ntree = 500,

mtry = 4, nodesize = 5, importance = TRUE)

rf.pred.df <-predict(rf.df, valid.dfnorm)

accuracy(valid.dfnorm$popularity, rf.pred.df)

cor(valid.dfnorm$popularity,rf.pred.df)

#Lift Chart plot

gain.rf <- gains(valid.dfnorm$popularity,rf.pred.df, groups = length(rf.pred.df))

plot(c(0, gain.rf$cume.pct.of.total\*sum(valid.dfnorm$popularity))~c(0, gain.rf$cume.obs), xlab = "# cases", ylab = "Cumulative", main = "Lift Chart of Random Forests Tech(Original Dataset)", type = "l")

lines(c(0, sum(valid.dfnorm$popularity))~c(0, dim(valid.dfnorm)[1]), lty = 2)

# On PCA Dataset

set.seed(111)

rf.pc <-randomForest(popularity~ ., data = train.pc,ntree = 500,

mtry = 4, nodesize = 5, importance = TRUE)

rf.pred.pc <-predict(rf.pc, valid.pc)

accuracy(valid.pc$popularity, rf.pred.pc)

cor(valid.pc$popularity,rf.pred.pc)

#Lift Chart plot

gain.rf <- gains(valid.dfnorm$popularity,rf.pred.pc, groups = length(rf.pred.pc))

plot(c(0, gain.rf$cume.pct.of.total\*sum(valid.dfnorm$popularity))~c(0, gain.rf$cume.obs), xlab = "# cases", ylab = "Cumulative", main = "Lift Chart of Random Forests Tech(PCA Dataset)", type = "l")

lines(c(0, sum(valid.dfnorm$popularity))~c(0, dim(valid.dfnorm)[1]), lty = 2)

```

######################### Implementing Support Vector Machine Tech

```{r }

# On Original Dataset

svm.df <- svm(popularity~ ., data = train.dfnorm)

#

svm.pred.df <- predict(svm.df, valid.dfnorm)

#

accuracy(valid.dfnorm$popularity, svm.pred.df)

cor(valid.dfnorm$popularity,svm.pred.df)

#Lift Chart plot

gain.svm <- gains(valid.dfnorm$popularity,svm.pred.df, groups = length(svm.pred.df))

plot(c(0, gain.svm$cume.pct.of.total\*sum(valid.dfnorm$popularity))~c(0, gain.svm$cume.obs), xlab = "# cases", ylab = "Cumulative", main = "Lift Chart of SVM(Ori Dataset)", type = "l")

lines(c(0, sum(valid.dfnorm$popularity))~c(0, dim(valid.dfnorm)[1]), lty = 2)

# On PCA Dataset

svm.pc <- svm(popularity~ ., data = train.pc)

#

svm.pred.pc <- predict(svm.pc, valid.pc)

#

accuracy(valid.pc$popularity, svm.pred.pc)

cor(valid.pc$popularity,svm.pred.pc)

#Lift Chart plot

gain.svm <- gains(valid.dfnorm$popularity,svm.pred.pc, groups = length(svm.pred.pc))

plot(c(0, gain.svm$cume.pct.of.total\*sum(valid.dfnorm$popularity))~c(0, gain.svm$cume.obs), xlab = "# cases", ylab = "Cumulative", main = "Lift Chart of SVM(PCA Dataset)", type = "l")

lines(c(0, sum(valid.dfnorm$popularity))~c(0, dim(valid.dfnorm)[1]), lty = 2)

```

######################### Implementing Neural Networks Tech

```{r }

# On Original Dataset

set.seed(111)

nn.df <- neuralnet(popularity~ ., data = train.dfnorm, hidden =3,linear.output = T,stepmax = 1e6)

# display weights

nn.df$weights

# plot network

plot(nn.df,main = "Artificial Neural Net (Ori)",rep="best")

#

nn.pred.df <- compute(nn.df, valid.dfnorm)

#

nn.pred.value.df <- nn.pred.df$net.result

#

accuracy(valid.dfnorm$popularity, nn.pred.value.df)

cor(valid.dfnorm$popularity, nn.pred.value.df)

#Lift Chart plot

gain.nn <- gains(valid.dfnorm$popularity,nn.pred.value.df, groups = length(nn.pred.value.df))

plot(c(0, gain.nn$cume.pct.of.total\*sum(valid.dfnorm$popularity))~c(0, gain.nn$cume.obs), xlab = "# cases", ylab = "Cumulative", main = "Lift Chart of Nerual Net(Ori Dataset)", type = "l")

lines(c(0, sum(valid.dfnorm$popularity))~c(0, dim(valid.dfnorm)[1]), lty = 2)

# On PCA Dataset

set.seed(111)

nn.pc <- neuralnet(popularity~ ., data = train.pc, hidden =3,linear.output = T,stepmax = 1e6)

# display weights

nn.pc$weights

# plot network

plot(nn.pc,main = "Artificial Neural Net (PCA)",rep="best")

#

nn.pred.pc <- compute(nn.pc, valid.pc)

#

nn.pred.value.pc <- nn.pred.pc$net.result

#

accuracy(valid.pc$popularity, nn.pred.value.pc)

cor(valid.pc$popularity, nn.pred.value.pc)

#Lift Chart plot

gain.nn <- gains(valid.dfnorm$popularity,nn.pred.value.pc, groups = length(nn.pred.value.pc))

plot(c(0, gain.nn$cume.pct.of.total\*sum(valid.dfnorm$popularity))~c(0, gain.nn$cume.obs), xlab = "# cases", ylab = "Cumulative", main = "Lift Chart of Neural Net(PCA Dataset)", type = "l")

lines(c(0, sum(valid.dfnorm$popularity))~c(0, dim(valid.dfnorm)[1]), lty = 2)

```

##############################################################Lift Charts Plots

```{r }

#

# Plot all lift charts for Original Dataset

df.lift <- data.frame(case = c(1:dim(valid.dfnorm)[1]),

knnreg = cumsum(valid.dfnorm$popularity[order(song.knn.pred, decreasing = T)]),

linearreg = cumsum(valid.dfnorm$popularity[order(song.lm.pred, decreasing = T)]),

rfreg = cumsum(valid.dfnorm$popularity[order(rf.pred.df, decreasing = T)]),

svmreg = cumsum(valid.dfnorm$popularity[order(svm.pred.df, decreasing = T)]),

nnreg = cumsum(valid.dfnorm$popularity[order(nn.pred.value.df, decreasing = T)]),

baseline = c(1:dim(valid.dfnorm)[1])\*mean(valid.dfnorm$popularity)

)

ggplot(df.lift, aes(x = case)) +

geom\_line(aes(y = knnreg), color = "blue") +

geom\_line(aes(y = linearreg), color = "black") +

geom\_line(aes(y=rfreg), color = "green" ) +

geom\_line(aes(y=svmreg), color = "grey") +

geom\_line(aes(y=nnreg), color = "yellow") +

geom\_line(aes(y=baseline), color = "red", linetype = "dashed") +

theme\_bw()+

labs(x = "# of cases", y = "Cumulative",title = "Lift Charts for Original dataset with different model")

# Plot all lift charts for PCA Dataset

pc.lift <- data.frame(case = c(1:dim(valid.pc)[1]),

knnreg = cumsum(valid.pc$popularity[order(pc.knn.pred, decreasing = T)]),

linearreg = cumsum(valid.pc$popularity[order(pc.lm.pred, decreasing = T)]),

rfreg = cumsum(valid.pc$popularity[order(rf.pred.pc, decreasing = T)]),

svmreg = cumsum(valid.pc$popularity[order(svm.pred.pc, decreasing = T)]),

nnreg = cumsum(valid.pc$popularity[order(nn.pred.value.pc, decreasing = T)]),

baseline = c(1:dim(valid.pc)[1])\*mean(valid.pc$popularity)

)

ggplot(pc.lift, aes(x = case),shape) +

geom\_line(aes(y = knnreg), color = "blue") +

geom\_line(aes(y = linearreg), color = "black") +

geom\_line(aes(y=rfreg), color = "green" ) +

geom\_line(aes(y=svmreg), color = "grey") +

geom\_line(aes(y=nnreg), color = "yellow") +

geom\_line(aes(y=baseline), color = "red", linetype = "dashed") +

theme\_bw()+

labs(x = "# of cases", y = "Cumulative", title = "Lift Charts for PCA dataset with different model")

```