USE CASE STUDY REPORT

Prediction of NBA's The Most Valuable Player based on Data Mining

Group No.: Group 14

Student Names: Anxi Liu and Anyuan Xu

Executive Summary:

In this project, we aim to find a supervised machine learning algorithm which can predict the NBA's MVP winner with the highest accuracy and use this algorithm to predict the MVP winner for the current season.

Firstly, we obtain the raw dataset on the website and do the data exploration and visualization. Secondly, we do the data preparation and preprocessing by using correlation analysis and PCA analysis, thereby we obtain two kinds of dataset one is normalized dataset by correlation analysis the other is the PCA dataset. Thirdly, we partition the dataset into training (60%) and validation (40%) dataset and implement all data mining techniques which we have learned and can be used for the prediction problem on the training dataset. Then, after we obtain the models, we apply them on the validation dataset to evaluate performance by comparing the lift chart and prediction accuracy such as: RMSE and MAE.

Finally, we conclude that the best case is that we use support vector regression (SVR) to train our model on the dataset obtained from correlation analysis. Consequently, we recommend that we can use the SVR method to predict the MVP for the current NBA season.

I. Background and Introduction

Background

Nowadays, the big data technology has profound effects on the athletic field increasingly. More and more sports leagues such as the US four professional sports leagues intend to use data to drive sports decisions, therefore professional analytics team is created in each team which can provide support to each team in many aspects such as teams and players performance, sports injury prevention, players marketing, awards evaluation and so on. In addition, some well-known IT companies like SAP and IBM collaborate with sports teams in big data field to support in the information retrieve, data collection and storage, data processing and data analytics, which can help to improve both team performance and commercial value.

This project will focus on a specific awards evaluation: predicting The Most Valuable Player (MVP) in National Basketball Association (NBA). This award is decided by the media members across the NBA, and they will select athletes who meet the standards in their hearts by the end of regular season. The MVP award is created in the 1955–56 season, it aims to reward the best performing and most consistent player of the regular season. Many sports companies will predict the MVP before the result comes out and it will provide reference to the NBA league and many sponsors.

The problem

Our problem is a prediction problem. In the regular season, the NBA will continue to update a list including top 10 possible MVP candidates and the media members' selection will almost come from this list. In the end, the player with the highest percentage of votes will be awarded. Therefore, our problem is that how to predict which player will achieve the highest percentage of votes according to this list.

The goal of this project

Our goal is to find a supervised machine learning algorithm which can predict the MVP winner with the highest accuracy and use this algorithm to predict the MVP winner for the current season.

The solution

1. Find the attributes regarding the players in the candidate list and obtain our data set. All the records from the dataset come from https://www.basketball-reference.com.

2. Perform data exploration, visualization, preparation, and preprocessing.

3. For this prediction problem, implement all machine learning algorithms which we have learned and can be used for the prediction problem on the training dataset and fit the model respectively.

4. Evaluate the performance on the validation dataset and select the algorithm that has the highest accuracy.

II. Data Exploration and Visualization

Data description

As we can see in the Figure 1, our data set consists of 412 observations and 23 variables. All the variables excepted the player's and team's name are numerical variable, thus there is no categorical variables.

For the 412 observations, they contain top 10 NBA MVP candidates in 40 seasons from 1979-1980 season to 2019-2020 season, and we select past 40 years records because an important attribute: the 3-point was introduced in 1979-1980 season.

For the 23 variables, they include the attribute of MVP votes, team performance, player individual performance and individual advanced data. Apparently, the variable named "Share" is our outcome variable (aka response) and it represent the percentage of votes. Other variables will be filtered to compose our input variables (aka predictors).

```
'data.frame':   412 obs. of  23 variables:
 $ Rank        : int  1 2 3 4 5 5 5 8 9 1 ...
 $ Player      : chr  "Kareem Abdul-Jabbar" "Julius Erving" "George Gervin" "Larry Bird" ...
 $ Age         : int  32 29 27 23 31 25 26 26 24 30 ...
 $ Tm          : chr  "LAL" "PHI" "SAS" "BOS" ...
 $ Pts.Won     : num  147 31.5 19 15 2 2 2 1.5 1 454 ...
 $ Pts.Max     : int  221 221 221 221 221 221 221 221 221 690 ...
 $ Share       : num  0.665 0.143 0.086 0.068 0.009 0.009 0.009 0.007 0.005 0.658 ...
 $ G           : num  82 78 78 82 80 81 81 82 82 82 ...
 $ Team.Wins   : num  60 59 41 61 61 56 50 56 41 62 ...
 $ Overall.Seed: int  2 3 10 1 1 4 6 4 9 2 ...
 $ MP          : num  38.3 36.1 37.6 36 35.8 36.3 32 36.2 38.3 35 ...
 $ PTS         : num  24.8 26.9 33.1 21.3 14.1 19 16.5 22.1 25.8 24.6 ...
 $ TRB         : num  10.8 7.4 5.2 10.4 2.5 5.1 10.3 3.4 14.5 8 ...
 $ AST         : num  4.5 4.6 2.6 4.5 8.4 4.1 2.3 4.8 1.8 4.4 ...
 $ STL         : num  1 2.2 1.4 1.7 1.3 1.8 1.2 2.4 1 2.1 ...
 $ BLK         : num  3.4 1.8 1 0.6 0.1 1 1.7 0.5 1.3 1.8 ...
 $ FG.         : num  0.604 0.519 0.528 0.474 0.482 0.422 0.499 0.482 0.502 0.521 ...
 $ X3P.        : num  0 0.2 0.314 0.406 0.222 0.207 0 0.194 0 0.222 ...
 $ FT.         : num  0.765 0.787 0.852 0.836 0.83 0.78 0.71 0.788 0.719 0.787 ...
 $ WS          : num  14.8 12.5 10.6 11.2 8.9 7.4 9.1 11.6 11.9 13.8 ...
 $ WS.48       : num  0.227 0.213 0.173 0.182 0.148 0.12 0.169 0.187 0.183 0.231 ...
 $ VORP        : num  6.8 6.5 3.1 5.4 1.5 2.8 4.3 4.5 3.8 7.2 ...
 $ BPM         : num  6.7 7.2 2.2 5.3 0 1.7 4.6 4.1 2.8 8 ...
```

Figure 1: the structure of the data frame

Data exploration and visualization

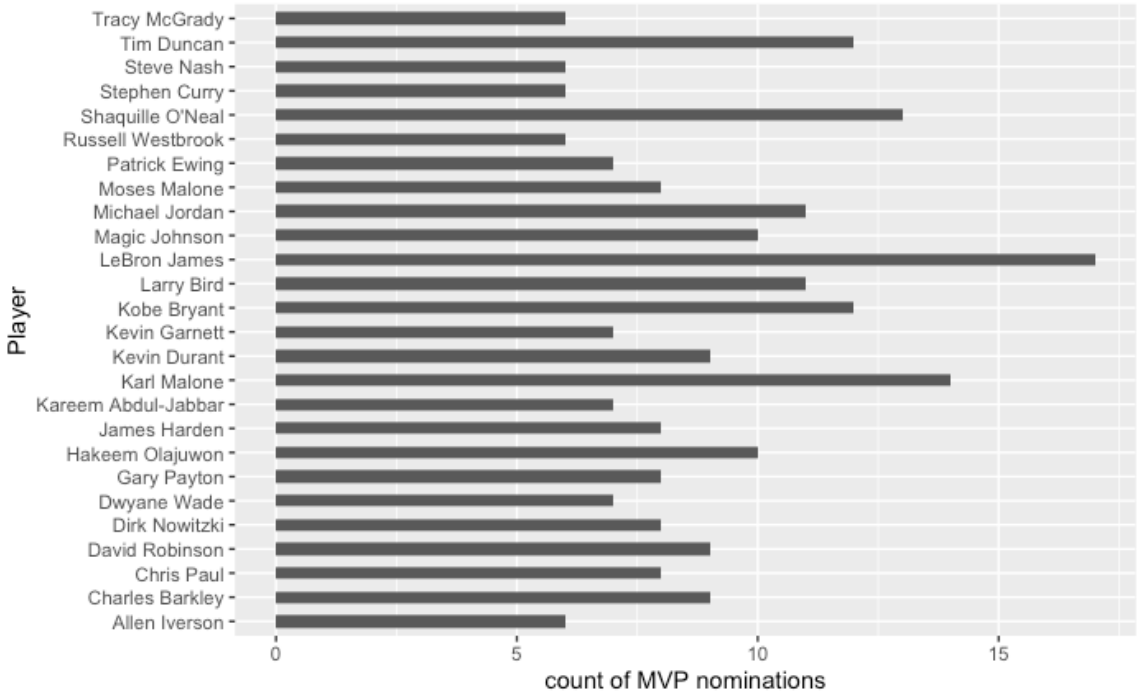


Figure 2: bar chart for the count of MVP candidates

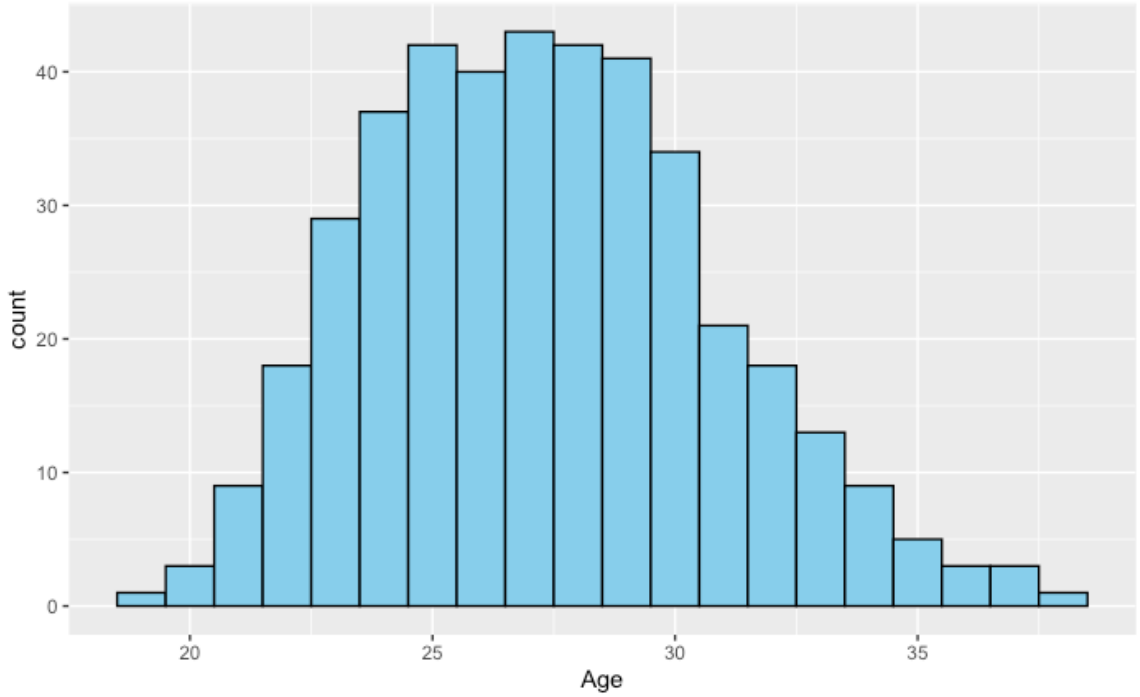


Figure 3: histogram plot for the distribution of MVP candidates' age

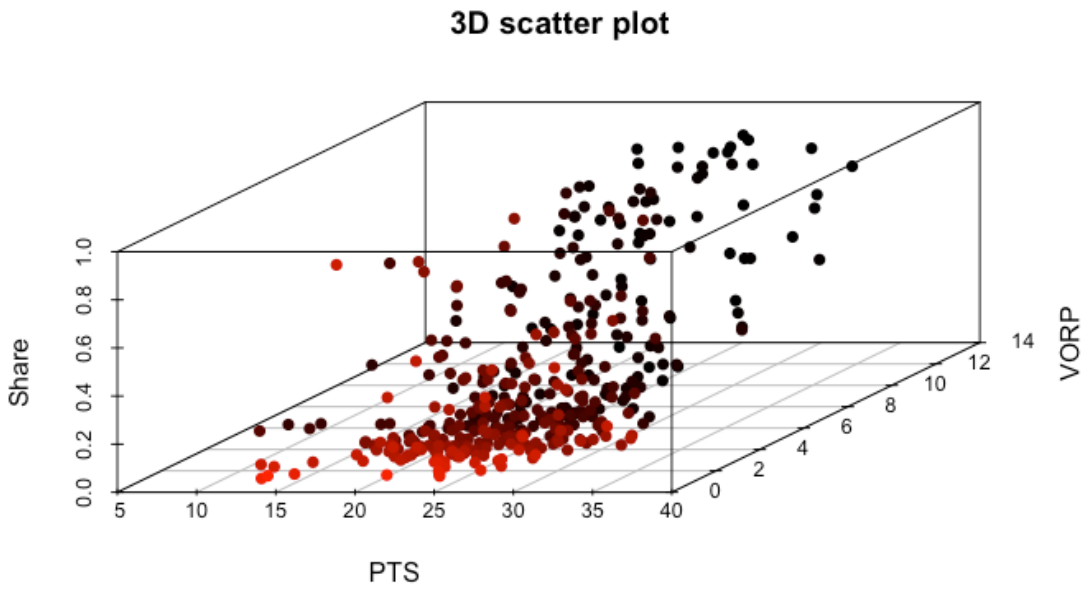


Figure 4: 3D scatter plot with the player features: Share, PTS, and VORP

III. Data Preparation and Preprocessing

Data summary, variable selection, and normalization
The summary statistics of variables is shown in figure 5, and variables named "Player" and "Tm" are character and others are numerical variables.

We do the first variable selection by domain knowledge, some irrelevant variable should be removed such as: "Rank", "Player", "Age", "Tm", etc.

Moreover, we normalize the predictors to eliminate the influence of variables of different dimensions on parameter learning.

```
      Rank           Player               Age             Tm              Pts.Won          Pts.Max          Share       
 Min.   : 1.000   Length:412         Min.   :19.0   Length:412         Min.   :   1.0   Min.   : 221   Min.   :0.00100  
 1st Qu.: 3.000   Class :character   1st Qu.:25.0   Class :character   1st Qu.:  29.0   1st Qu.: 920   1st Qu.:0.02775  
 Median : 5.000   Mode  :character   Median :27.0   Mode  :character   Median : 101.5   Median :1130   Median :0.10600  
 Mean   : 5.502                      Mean   :27.3                      Mean   : 266.2   Mean   :1043   Mean   :0.25173  
 3rd Qu.: 8.000                      3rd Qu.:30.0                      3rd Qu.: 427.2   3rd Qu.:1230   3rd Qu.:0.40175  
 Max.   :12.000                      Max.   :38.0                      Max.   :1310.0   Max.   :1310   Max.   :1.00000  
       G          Team.Wins      Overall.Seed          MP             PTS              TRB              AST       
 Min.   :44.0   Min.   :24.00   Min.   : 1.000   Min.   :28.00   Min.   : 6.90   Min.   : 2.200   Min.   : 1.000  
 1st Qu.:74.0   1st Qu.:49.00   1st Qu.: 3.000   1st Qu.:35.30   1st Qu.:20.88   1st Qu.: 5.300   1st Qu.: 3.000  
 Median :79.0   Median :55.00   Median : 5.000   Median :37.20   Median :23.90   Median : 7.700   Median : 4.700  
 Mean   :76.5   Mean   :53.57   Mean   : 6.129   Mean   :36.92   Mean   :23.74   Mean   : 8.046   Mean   : 5.298  
 3rd Qu.:81.0   3rd Qu.:58.00   3rd Qu.: 9.000   3rd Qu.:38.60   3rd Qu.:27.02   3rd Qu.:10.800   3rd Qu.: 7.000  
 Max.   :82.0   Max.   :73.00   Max.   :24.000   Max.   :43.70   Max.   :37.10   Max.   :18.700   Max.   :14.500  
      STL              BLK              FG.              X3P.              FT.              WS             WS.48       
 Min.   :0.500   Min.   :0.100   Min.   :0.3840   Min.   :0.0000   Min.   :0.4230   Min.   : 5.10   Min.   :0.0780  
 1st Qu.:1.000   1st Qu.:0.400   1st Qu.:0.4677   1st Qu.:0.1913   1st Qu.:0.7338   1st Qu.:10.20   1st Qu.:0.1777  
 Median :1.500   Median :0.800   Median :0.4970   Median :0.3025   Median :0.7870   Median :11.90   Median :0.2040  
 Mean   :1.491   Mean   :1.107   Mean   :0.4990   Mean   :0.2591   Mean   :0.7786   Mean   :12.22   Mean   :0.2068  
 3rd Qu.:1.800   3rd Qu.:1.600   3rd Qu.:0.5260   3rd Qu.:0.3613   3rd Qu.:0.8472   3rd Qu.:14.00   3rd Qu.:0.2343  
 Max.   :3.200   Max.   :4.600   Max.   :0.6690   Max.   :0.5000   Max.   :0.9480   Max.   :21.20   Max.   :0.3220  
      VORP             BPM         
 Min.   : 0.500   Min.   :-1.300  
 1st Qu.: 4.000   1st Qu.: 4.000  
 Median : 5.300   Median : 5.400  
 Mean   : 5.481   Mean   : 5.651  
 3rd Qu.: 6.700   3rd Qu.: 7.400  
 Max.   :12.400   Max.   :15.600  
```

Figure 5: the summary statistics of variables

Correlation analysis

As we can see from the figure 6, some predictors have strong correlations that will cause multicollinearity problems. According to the correlation matrix, we remove predictors which have strong positive and negative correlations such as "Team. Wins", "X3P.", "WS.48", "BPM", etc. Thereby, we have created our reduced dimensionality dataset by correlation analysis.

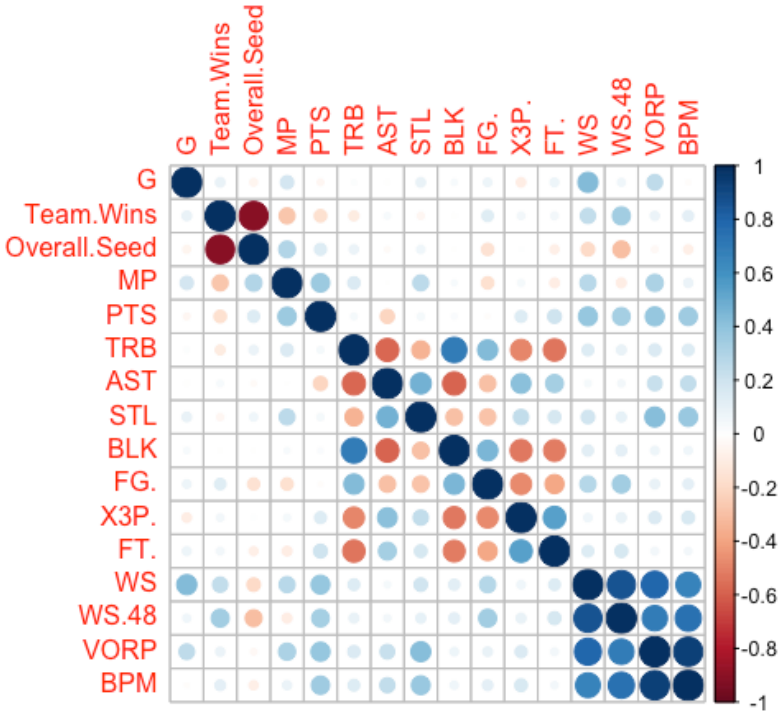


Figure 6: correlation matrix of predictors

PCA analysis

Since our dataset consists of many correlated attributes, it is indispensable to select only linearly independent attributes. We can also use PCA analysis to extract the components from the dataset, to achieve the purpose of reducing the dimension. Firstly, we determine to extract 5 components from 16 variables by parallel analysis according to the figure 7. Then, our 5 PCA components analysis and the correlation between them are illustrated in figure 8 and figure 9, and that show there are no correlation between these 5 components. Finally, we obtain our dataset with PCA components.

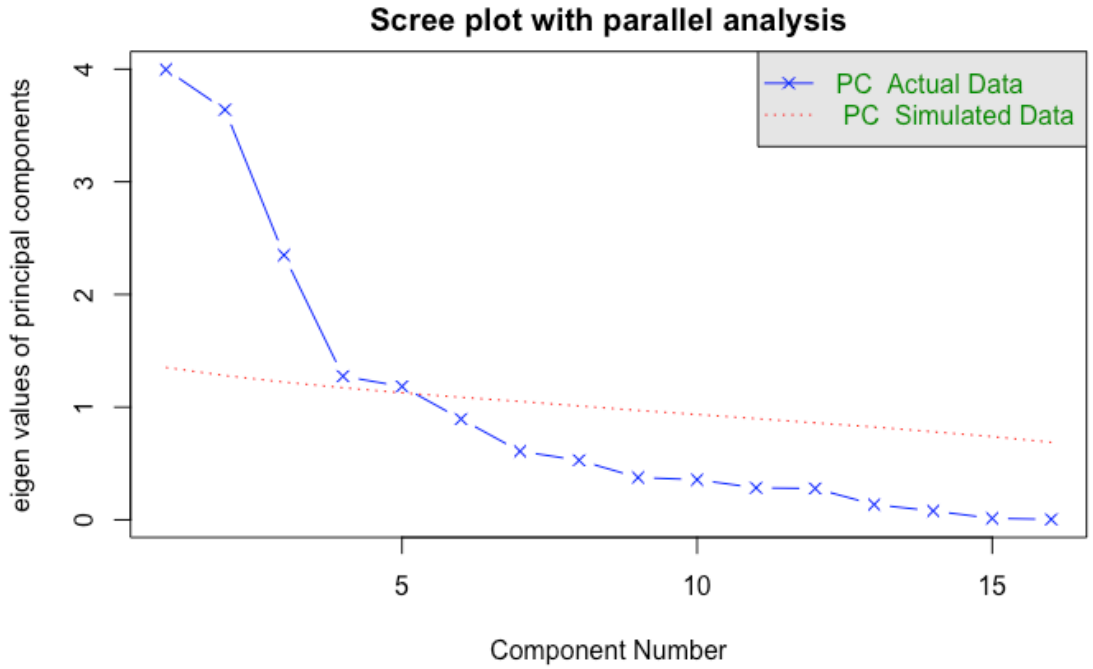


Figure 7: parallel analysis

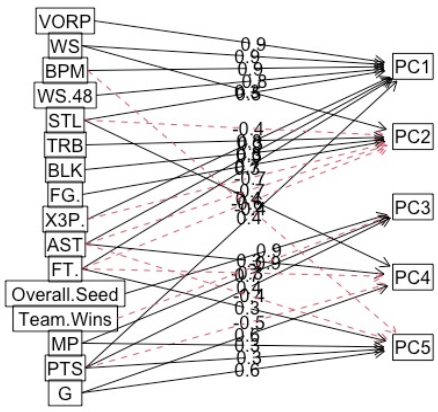


Figure 8: PCA components analysis

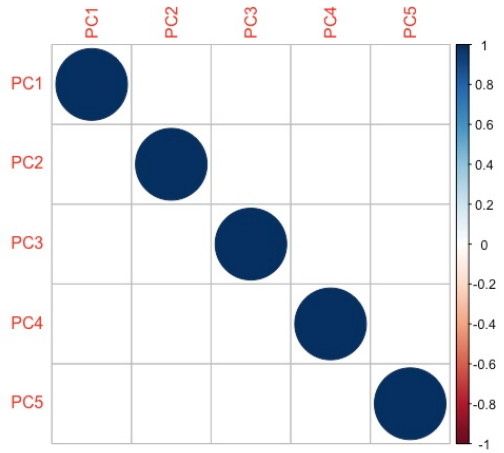


Figure 9: correlation matrix of PCA components

IV. Data Mining Techniques and Implementation

The machine learning algorithms we used for our prediction problem are as follows.

1. Linear Regression
2. K Nearest Neighbors (k-NN) Regression
3. Regression Tree
4. Random Forests
5. Neural Network
6. Support Vector Regression (SVR)

We implement these data mining techniques on the datasets we obtained from both correlation analysis and PCA analysis. And the flow chart figure 10 is as follows.

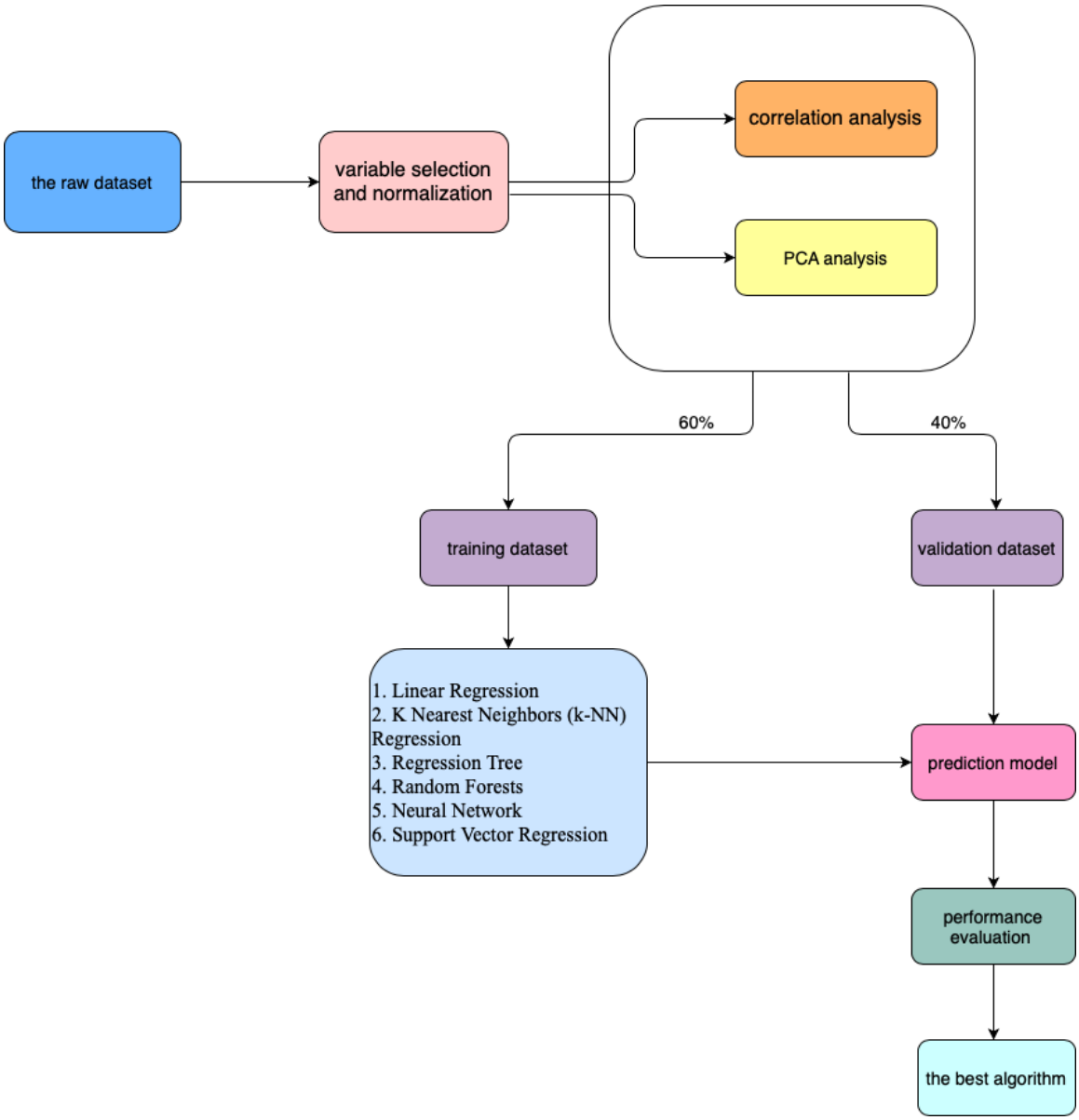


Figure 10:data mining process flow chart

Our regression tree and neural network can be represented as charts as follows.

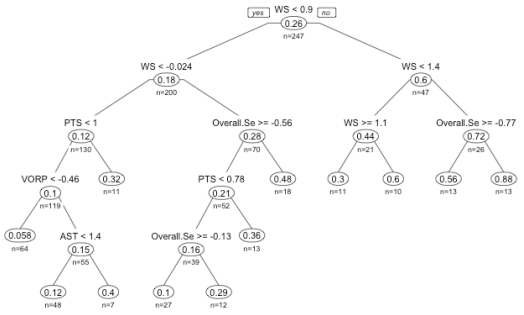


Figure 11: regression tree

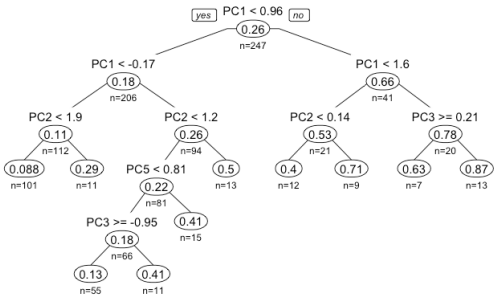


Figure 12: regression tree (PCA)

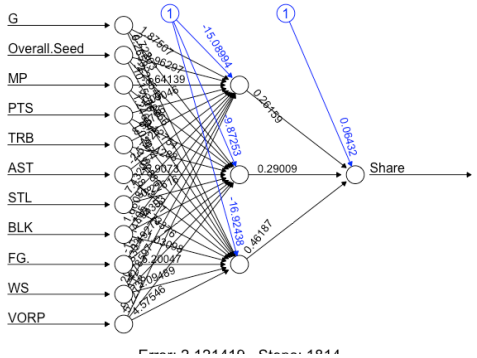


Figure 13: neural network with 3 hidden layers

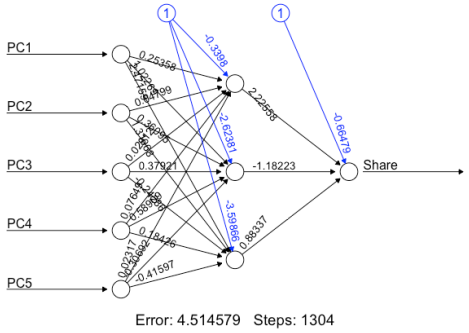


Figure 14: neural network with 3 hidden layers (PCA)

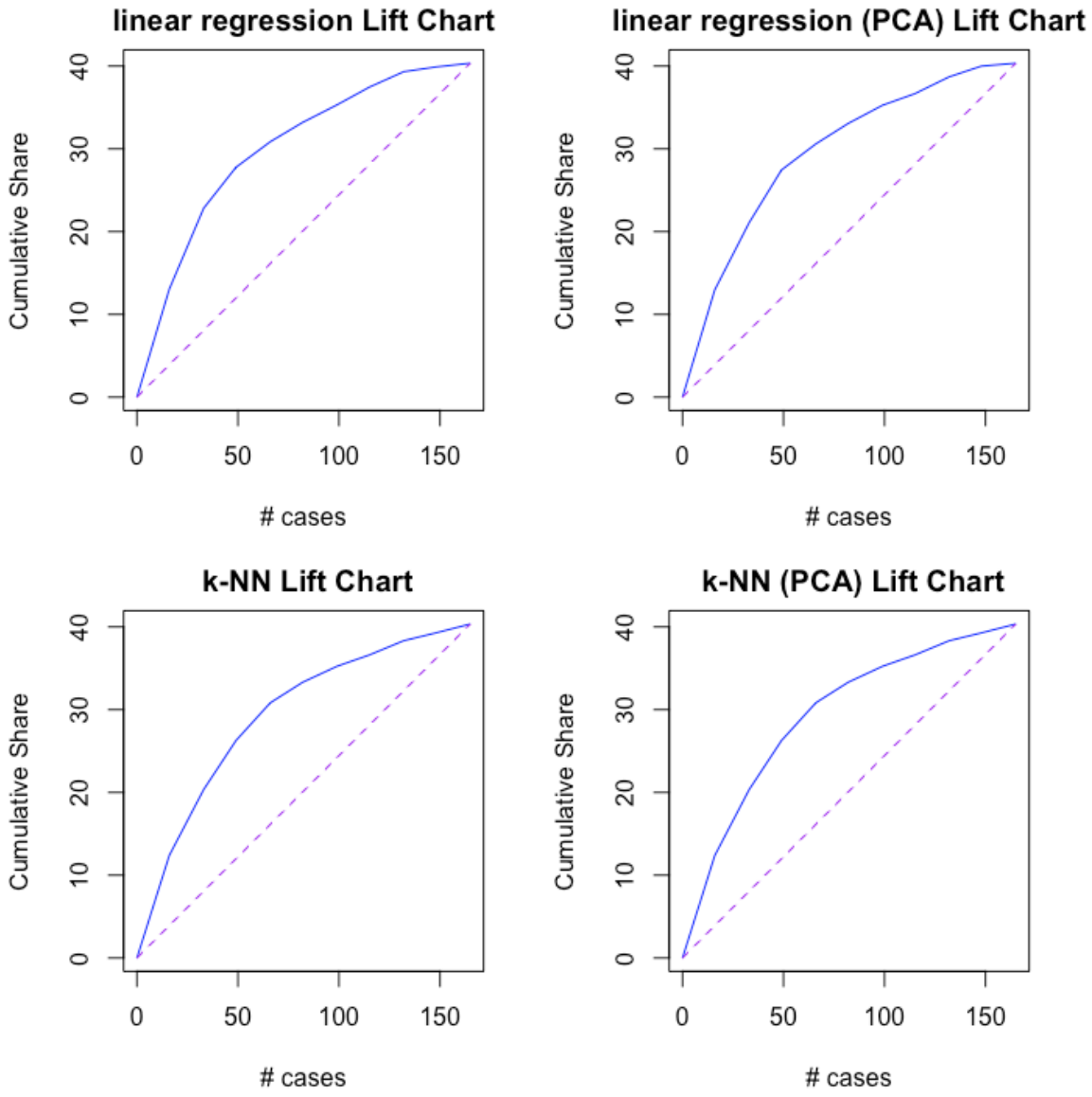
When we implement k-NN method, we select the best k between 1-20 that has the lowest RMSE error. According to the table 1 as follows, one dataset we choose k=9, the other PCA dataset we choose k=8.

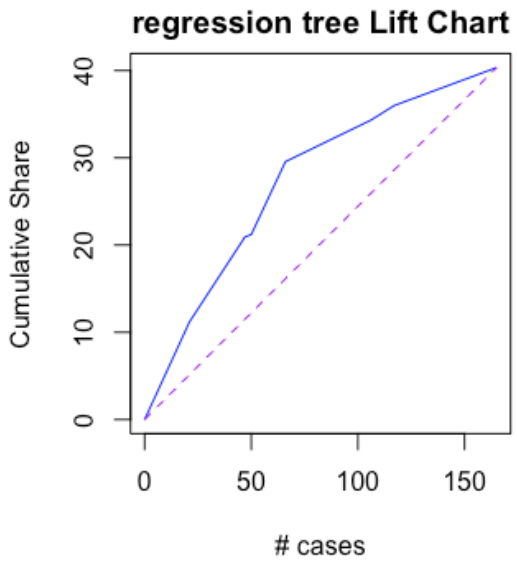
k	RMSE	k	RMSE
1	0.2629739	1	0.2799422
2	0.2378650	2	0.2392293
3	0.2151462	3	0.2183900
4	0.2086571	4	0.2111955
5	0.2058592	5	0.1984408
6	0.2058254	6	0.1940595
7	0.2005838	7	0.1909620
8	0.1940672	8	0.1905104
9	0.1918758	9	0.1975574
10	0.1946977	10	0.1993703
11	0.1964732	11	0.2011736
12	0.1979958	12	0.1981330
13	0.1982914	13	0.1994200
14	0.1996551	14	0.1994079
15	0.2002380	15	0.1996147
16	0.2001329	16	0.1998933
17	0.2004358	17	0.1992181
18	0.2029834	18	0.2004470
19	0.2041559	19	0.2007683
20	0.2051008	20	0.2022607

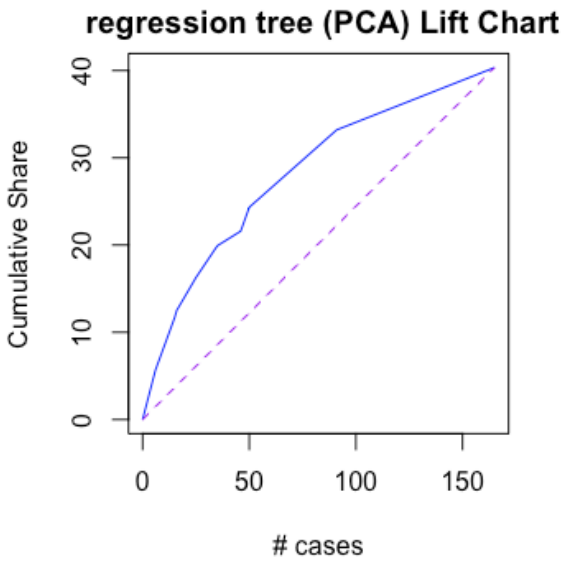
Table 1: k-NN RMSE list

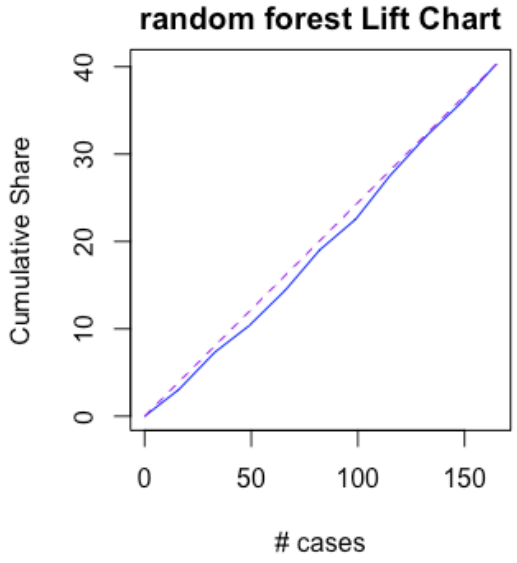
V. Performance Evaluation

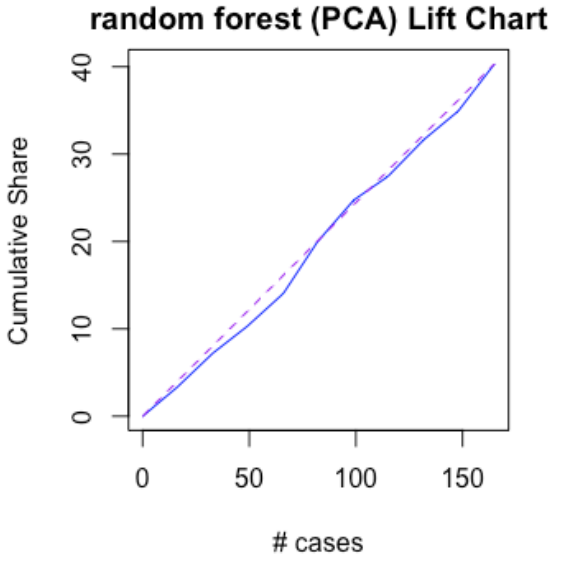
After we implement all algorithms on the training dataset and obtain the model respectively, we apply all models on the validation dataset to evaluate the performance by using lift chart and our accuracy list including RMSE and MAE, shown as figure 15 and table 2.

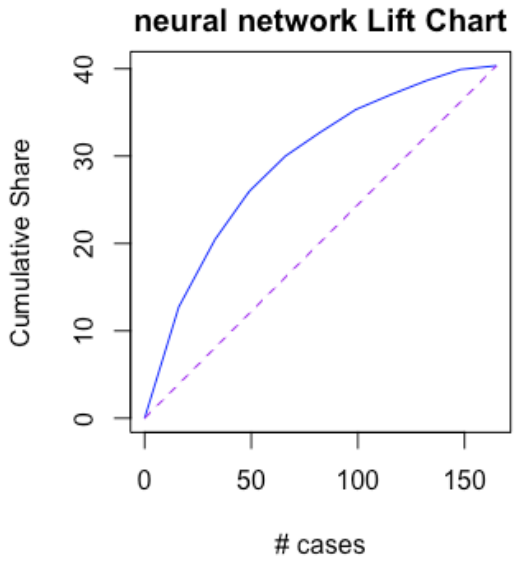


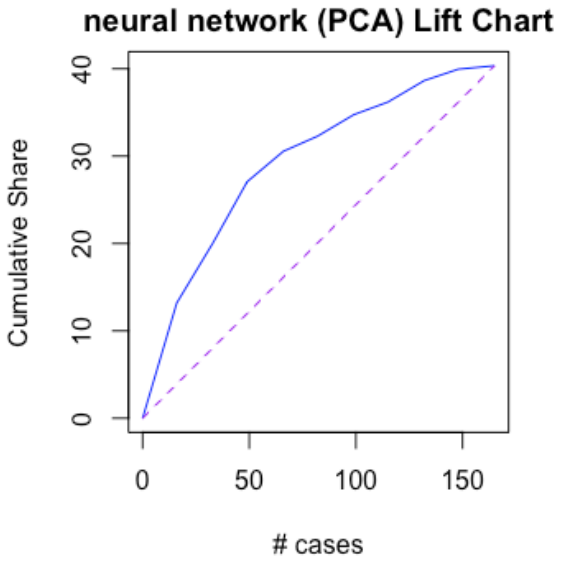

regression tree Lift Chart
Cumulative Share
0
10
20
30
40
0
50
100
150
cases


regression tree (PCA) Lift Chart
Cumulative Share
0
10
20
30
40
0
50
100
150
cases


random forest Lift Chart
Cumulative Share
0
10
20
30
40
0
50
100
150
cases


random forest (PCA) Lift Chart
Cumulative Share
0
10
20
30
40
0
50
100
150
cases


neural network Lift Chart
Cumulative Share
0
10
20
30
40
0
50
100
150
cases


neural network (PCA) Lift Chart
Cumulative Share
0
10
20
30
40
0
50
100
150
cases

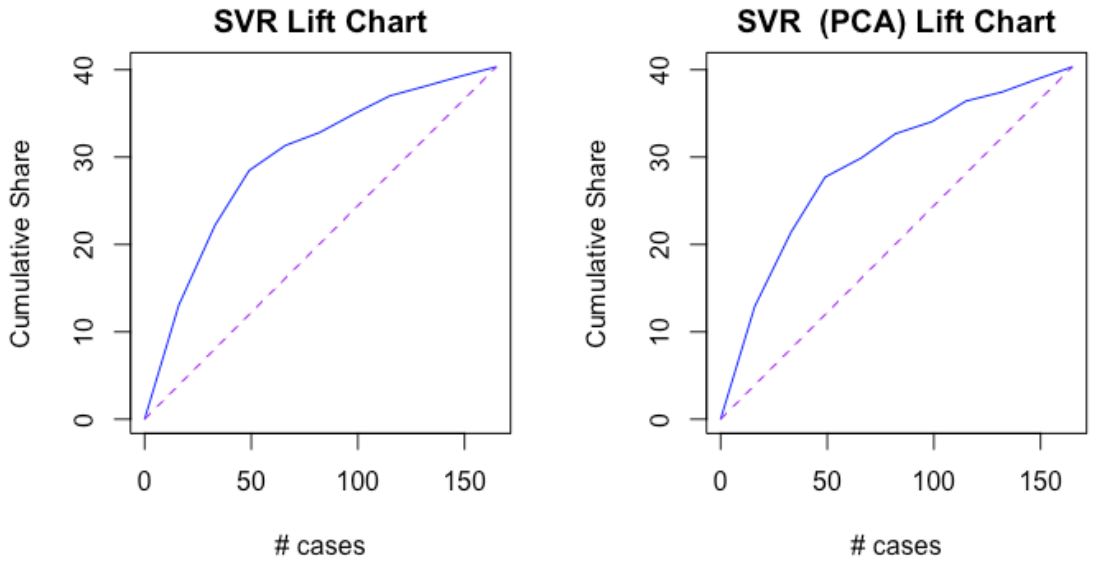


Figure 15: lift chart for all methods

Model	RMSE	MAE
Linear Regression	0.1881442	0.1494219
Linear Regression (PCA)	0.1993308	0.1533005
k-NN Regression	0.1918758	0.1458108
k-NN Regression (PCA)	0.1905104	0.1446912
Regression Tree	0.246735	0.1771493
Regression Tree (PCA)	0.2152023	0.1557992
Random Forests	0.3529442	0.2712877
Random Forests (PCA)	0.3371799	0.2560216
Neural Network	0.194352	0.1393411
Neural Network (PCA)	0.1833397	0.1347395
Support Vector Regression	0.1724506	0.1268904
Support Vector Regression (PCA)	0.1819132	0.1280873

Table 2: accuracy list

As we can see from the table 2 above, linear regression, k-NN regression, neural network, and support vector regression performs better than the tree model. Among them, support vector regression which achieves the lowest RMSE, and the lowest MAE has the best performance. Therefore, we select the support vector regression (SVR) as our best method. In addition, we don't need to extract components by PCA because the SVR on the dataset on which we select the variable by correlation analysis performs higher accuracy than the PCA dataset.

VI. Discussion and Recommendation

According to the accuracy list and lift chart shown as figure 15 and table 2, we can find that linear regression, k-NN regression, neural network, and support vector regression performs better than the tree model.

Regression tree model performs worse, for one thing regression tree need a large-scale dataset to fit model better but our dataset is not large enough for the regression tree. For the other, regression tree can do the variable selection and reduction automatically, so our data preprocessing will affect regression tree performance.

The method which has the best performance is support vector regression (SVR) that is the kind of support vector machine (SVM) in prediction problem, it is reasonable because SVR combines k-NN and regression method to model highly complex relationship. However, this method become worse when the dataset is in larger scale because the space complexity and time complexity will increase.

Finally, we will recommend that we can use the SVR method to predict the MVP for the current NBA season.

VII. Summary

In this study, we apply 6 machine learning algorithms respectively on the datasets we obtained from correlation analysis and PCA analysis. Then, we evaluate the performance of these 12 kinds of models. Finally, in terms of our performance evaluation, the best case is that we use support vector regression (SVR) to train our model on the dataset obtained from correlation analysis, which performs the highest accuracy.

Appendix: R Code for use case study

Import Data Set

```
MVP_df <- read.csv("mvp.csv", header=TRUE)
MVP_df <- MVP_df[,c(-24:-27)]
str(MVP_df)
```

Data Exploration and Visualization

```
library(ggplot2)
library(tidyverse)

# age histogram
ggplot(MVP_df, aes(x=Age)) +
  geom_histogram(color="black", fill="sky blue", binwidth = 1)

# 3d scatter plot
library(scatterplot3d)
attach(MVP_df)
scatterplot3d(PTS, VORP, Share,
              highlight.3d = TRUE,
              pch = 16,
              main = "3D scatter plot")
detach(MVP_df)

# bar chart: count of MVP nominations
MVP_df_count <- group_by(MVP_df, Player) %>% summarise(count = n())
MVP_df_count1<-subset(MVP_df_count,count>5,select=c(Player,count))
ggplot(data=MVP_df_count1)+
  geom_bar(mapping = aes(x=Player,y=count), stat='identity',width=0.5)+
ylab("count of MVP nominations")+ coord_flip()
```

Data Preparation and Preprocessing

```
library(psych)
library(tidyverse)
library(corrplot)

# data summary
summary(MVP_df)

# using domain knowledge: remove irrelevant variables
MVP_df1 <- MVP_df[,c(-1:-6)]

# original
MVP_df2 <- MVP_df1[,c(-3,-12,-13,-15,-17)]
```

```
# normalization
MVP_norm_df <- cbind(MVP_df1$Share, as.data.frame(scale(MVP_df1[,2:1
7]))) %>% rename(Share = "MVP_df1$Share")

# 1. correlation analysis
MVP_norm_df1 <- MVP_norm_df[c(2:17)]
corrplot(cor(MVP_norm_df1))
pairs.panels(MVP_norm_df1 , cor = TRUE)

# variable selection: removing variables that are strongly correlated t
o others is useful for avoiding multicollinearity problems
MVP_norm_df2 <- MVP_norm_df1[,c(-2,-11,-12,-14,-16)]
corrplot(cor(MVP_norm_df2))


# normal dataset
MVP_norm_df3 <- cbind(MVP_norm_df$Share, MVP_norm_df2) %>% rename(Share
 = "MVP_norm_df$Share")



# 2. PCA analysis

# input the correlation matrix to fa.parallel() function to determine t
he number of components to extract
fa.parallel(cor(MVP_norm_df1), n.obs = 412 , fa = "pc", n.iter = 100, s
how.legend = TRUE, main = "Scree plot with parallel analysis")

MVP_pca <- principal(MVP_norm_df1, nfactors = 5, rotate = "none", score
s = TRUE)
MVP_pca
MVP_pca_df <- as.data.frame(MVP_pca$scores)

factor.plot(MVP_pca)
fa.diagram(MVP_pca, simple = FALSE)
corrplot(cor(MVP_pca_df))
pairs.panels(MVP_pca_df , cor = TRUE)


# PCA dataset
MVP_pca_df1 <- cbind(MVP_norm_df$Share, MVP_pca_df) %>% rename(Share =
"MVP_norm_df$Share")
```

Data Mining Techniques and Implementation

Performance Evaluation

training and validation data set

```
# partition the data into training (60%) and validation (40%) sets
set.seed(100)
train_index <- sample(rownames(MVP_norm_df3 ), dim(MVP_norm_df3 )[1]*0.
6)
valid_index <- setdiff(rownames(MVP_norm_df3 ),train_index)
# dataset
MVP_train_df <- MVP_norm_df3[train_index,]
MVP_valid_df <- MVP_norm_df3[valid_index,]
# PCA dataset
MVP_train_pca_df <- MVP_pca_df1[train_index,]
MVP_valid_pca_df <- MVP_pca_df1[valid_index,]
```

linear regression

```
# linear regression
linear_model1 <- lm(Share ~ ., data = MVP_train_df)
summary(linear_model1)

lr_predict1 <- predict(linear_model1, MVP_valid_df)

# evaluating predictive performance
library(gains)
lr_gain1 <- gains(MVP_valid_df$Share, lr_predict1)
options(scipen=999)
lr_share1 <- (MVP_valid_df$Share)

# Lift chart
par(pty="s")
plot(c(0,lr_gain1$cume.pct.of.total*sum(lr_share1))~c(0,lr_gain1$cume.o
bs),
     xlab = "# cases", ylab = "Cumulative Share", main = "linear regres
sion Lift Chart", type = "l", col = "blue")
# baseline
lines(c(0,sum(lr_share1))~c(0,dim(MVP_valid_df)[1]), col = "purple", lt
y = 2)

# Decile-wise lift chart
barplot(lr_gain1$mean.resp/mean(lr_share1), names.arg = lr_gain1$depth,

        xlab="Percentile", ylab = "Mean Share", main = "linear regressi
on Decile-wise Lift Chart", col="pink")
```

```
# RMSE
library(Metrics)
RMSE <- rmse(MVP_valid_df$Share, lr_predict1)
RMSE
# MAE
MAE <- mae(MVP_valid_df$Share, lr_predict1)
MAE
# MAPE
library(MLmetrics)
MAPE <- MAPE(y_pred = MVP_valid_df$Share, y_true = lr_predict1)
MAPE
```

linear regression (PCA)

```
# linear regression PCA
linear_model2 <- lm(Share ~ ., data = MVP_train_pca_df)
summary(linear_model2)

lr_predict2 <- predict(linear_model2, MVP_valid_pca_df)

# evaluating predictive performance
library(gains)
lr_gain2 <- gains(MVP_valid_pca_df$Share, lr_predict2)
options(scipen=999)
lr_share2 <- (MVP_valid_pca_df$Share)

# Lift chart
par(pty="s")
plot(c(0,lr_gain2$cume.pct.of.total*sum(lr_share2))~c(0,lr_gain2$cume.o
bs),
     xlab = "# cases", ylab = "Cumulative Share", main = "linear regres
sion (PCA) Lift Chart", type = "l", col = "blue")
# baseline
lines(c(0,sum(lr_share2))~c(0,dim(MVP_valid_pca_df)[1]), col = "purple
", lty = 2)

# Decie-wise lift chart
barplot(lr_gain2$mean.resp/mean(lr_share2), names.arg = lr_gain2$depth,

        xlab="Percentile", ylab = "Mean Share", main = "linear regressi
on (PCA) Decile-wise Lift Chart", col="pink")

# RMSE
RMSE <- rmse(MVP_valid_pca_df$Share, lr_predict2)
RMSE
# MAE
MAE <- mae(MVP_valid_df$Share, lr_predict2)
MAE
```

```
# MAPE
MAPE <- MAPE(y_pred = MVP_valid_pca_df$Share, y_true = lr_predict2)
MAPE
```

k-NN

```
library(caret)
# compute k-NN for different k from 1 to 20 on validation set.
knn_rmse_df <- data.frame(k = seq(1, 20, 1), RMSE = 0)
for(i in 1:20){
  knn <- knnreg(Share~.,data=MVP_train_df, k=i)
  knn_pred <- predict(knn, MVP_valid_df)
  knn_rmse_df[i,2] <- rmse(MVP_valid_df$Share, knn_pred)
}
knn_rmse_df

# select k = 9
knn1 <- knnreg(Share~.,data=MVP_train_df, k=9)
knn_predict1 <- predict(knn1, MVP_valid_df)


# evaluating predictive performance
library(gains)
knn_gain1 <- gains(MVP_valid_df$Share, knn_predict1)
options(scipen=999)
knn_share1 <- (MVP_valid_df$Share)

# Lift chart
par(pty="s")
plot(c(0,knn_gain1$cume.pct.of.total*sum(knn_share1))~c(0,knn_gain1$cum
e.obs),
     xlab = "# cases", ylab = "Cumulative Share", main = " k-NN Lift Ch
art", type = "l", col = "blue")
# baseline
lines(c(0,sum(knn_share1))~c(0,dim(MVP_valid_df)[1]), col = "purple", l
ty = 2)

# Decile-wise lift chart
barplot(knn_gain1$mean.resp/mean(knn_share1), names.arg = knn_gain1$dep
th,
        xlab="Percentile", ylab = "Mean Share", main = " k-NN  Decile-w
ise Lift Chart", col="pink")

# rmse
library(Metrics)
RMSE <- rmse(MVP_valid_df$Share, knn_predict1)
RMSE
# MAE
MAE <- mae(MVP_valid_pca_df$Share, knn_predict1)
```

```
MAE
# MAPE
library(MLmetrics)
MAPE <- MAPE(y_pred = MVP_valid_df$Share, y_true = knn_predict1)
MAPE
```

k-NN (PCA)

```
# compute k-NN for different k from 1 to 20 on validation set.
knn_rmse_df <- data.frame(k = seq(1, 20, 1), RMSE = 0)
for(i in 1:20){
  knn <- knnreg(Share~.,data=MVP_train_pca_df, k=i)
  knn_pred <- predict(knn, MVP_valid_pca_df)
  knn_rmse_df[i,2] <- rmse(MVP_valid_pca_df$Share, knn_pred)
}
knn_rmse_df

# select k = 8
knn2 <- knnreg(Share~.,data=MVP_train_pca_df, k=8)
knn_predict2 <- predict(knn2, MVP_valid_pca_df)

# evaluating predictive performance
library(gains)
knn_gain2 <- gains(MVP_valid_pca_df$Share, knn_predict1)
options(scipen=999)
knn_share2 <- (MVP_valid_pca_df$Share)

# Lift chart
par(pty="s")
plot(c(0,knn_gain2$cume.pct.of.total*sum(knn_share2))~c(0,knn_gain2$cum
e.obs),
     xlab = "# cases", ylab = "Cumulative Share", main = " k-NN (PCA) L
ift Chart", type = "l", col = "blue")
# baseline
lines(c(0,sum(knn_share2))~c(0,dim(MVP_valid_pca_df)[1]), col = "purple
", lty = 2)

# Decile-wise lift chart
barplot(knn_gain2$mean.resp/mean(knn_share2), names.arg = knn_gain2$dep
th,
        xlab="Percentile", ylab = "Mean Share", main = " k-NN (PCA) Dec
ile-wise Lift Chart", col="pink")

# rmse
library(Metrics)
RMSE <- rmse(MVP_valid_pca_df$Share, knn_predict2)
RMSE
# MAE
MAE <- mae(MVP_valid_pca_df$Share, knn_predict2)
```

```
MAE
# MAPE
library(MLmetrics)
MAPE <- MAPE(y_pred = MVP_valid_pca_df$Share, y_true = knn_predict2)
MAPE
```

regression tree

```
library(rpart)
library(rpart.plot)
# run a regression tree
regression_tree1 <- rpart(Share~., data = MVP_train_df, method = "anova
")

prp(regression_tree1, type = 1, extra = 1, split.font = 1, varlen = -1
0, under = TRUE)

rt_predict1 <- predict(regression_tree1, MVP_valid_df)

# evaluating predictive performance
library(gains)
rt_gain1 <- gains(MVP_valid_df$Share, rt_predict1)
options(scipen=999)
rt_share1 <- (MVP_valid_df$Share)

# Lift chart
par(pty="s")
plot(c(0,rt_gain1$cume.pct.of.total*sum(rt_share1))~c(0,rt_gain1$cume.o
bs),
     xlab = "# cases", ylab = "Cumulative Share", main = "regression tr
ee Lift Chart", type = "l", col = "blue")
# baseline
lines(c(0,sum(rt_share1))~c(0,dim(MVP_valid_df)[1]), col = "purple", lt
y = 2)

# Decile-wise lift chart
barplot(rt_gain1$mean.resp/mean(rt_share1), names.arg = rt_gain1$depth,

        xlab="Percentile", ylab = "Mean Share", main = " regression tre
e Decile-wise Lift Chart", col="pink")

# rmse
library(Metrics)
RMSE <- rmse(MVP_valid_df$Share, rt_predict1)
RMSE
MAE <- mae(MVP_valid_df$Share, rt_predict1)
MAE
# MAPE
library(MLmetrics)
```

```
MAPE <- MAPE(y_pred = MVP_valid_df$Share, y_true = rt_predict1)
MAPE
```

regression tree (PCA)

```
# run a regression tree
regression_tree2 <- rpart(Share~., data = MVP_train_pca_df, method = "a
nova")

prp(regression_tree2, type = 1, extra = 1, split.font = 1, varlen = -1
0, under = TRUE)

rt_predict2 <- predict(regression_tree2, MVP_valid_pca_df)

# evaluating predictive performance
library(gains)
rt_gain2 <- gains(MVP_valid_pca_df$Share, rt_predict2)
options(scipen=999)
rt_share2 <- (MVP_valid_pca_df$Share)

# Lift chart
par(pty="s")
plot(c(0,rt_gain2$cume.pct.of.total*sum(rt_share2))~c(0,rt_gain2$cume.o
bs),
     xlab = "# cases", ylab = "Cumulative Share", main = " regression t
ree (PCA) Lift Chart", type = "l", col = "blue")
# baseline
lines(c(0,sum(rt_share2))~c(0,dim(MVP_valid_pca_df)[1]), col = "purple
", lty = 2)

# Decile-wise lift chart
barplot(rt_gain2$mean.resp/mean(rt_share2), names.arg = rt_gain2$depth,

        xlab="Percentile", ylab = "Mean Share", main = " regression tre
e (PCA) Decile-wise Lift Chart", col="pink")

# rmse
library(Metrics)
RMSE <- rmse(MVP_valid_pca_df$Share, rt_predict2)
RMSE
MAE <- mae(MVP_valid_df$Share, rt_predict2)
MAE
# MAPE
library(MLmetrics)
MAPE <- MAPE(y_pred = MVP_valid_pca_df$Share, y_true = rt_predict2)
MAPE
```

random forest

```
# random forest regression
MVP_train_RFdf1 <- MVP_train_df
MVP_valid_RFdf1 <- MVP_valid_df
set.seed(100)
library(randomForest)
RF_model1<-randomForest(Share ~ ., data = MVP_train_RFdf1)
importance(RF_model1)
varImpPlot(RF_model1)

pred1=predict(RF_model1,data =MVP_valid_RFdf1)
MVP_valid_RFdf1$rf_predict1 <- 0
MVP_valid_RFdf1$rf_predict1[which(MVP_valid_RFdf1$rf_predict1==0)] <- p
red1



# evaluating predictive performance
library(gains)
rf_gain1 <- gains(MVP_valid_RFdf1$Share, MVP_valid_RFdf1$rf_predict1)
options(scipen=999)
rf_share1 <- (MVP_valid_RFdf1$Share)

# Lift chart
par(pty="s")
plot(c(0,rf_gain1$cume.pct.of.total*sum(rf_share1))~c(0,rf_gain1$cume.o
bs),
     xlab = "# cases", ylab = "Cumulative Share", main = "random forest
 Lift Chart", type = "l", col = "blue")
# baseline
lines(c(0,sum(rf_share1))~c(0,dim(MVP_valid_RFdf1)[1]), col = "purple",
 lty = 2)

# Decile-wise lift chart
barplot(rf_gain1$mean.resp/mean(rf_share1), names.arg = rf_gain1$depth,

        xlab="Percentile", ylab = "Mean Share", main = "random forest D
ecile-wise Lift Chart", col="pink")

# rmse
library(Metrics)
RMSE <- rmse(MVP_valid_RFdf1$Share, MVP_valid_RFdf1$rf_predict1)
RMSE
# MAE
MAE <- mae(MVP_valid_RFdf1$Share, MVP_valid_RFdf1$rf_predict1)
MAE
# MAPE
library(MLmetrics)
```

```
MAPE <- MAPE(y_pred = MVP_valid_RFdf1$Share, y_true = MVP_valid_RFdf1$r
f_predict1)
MAPE
```

random forest (PCA)

```
MVP_train_RFdf2 <- MVP_train_pca_df
MVP_valid_RFdf2 <- MVP_valid_pca_df
set.seed(100)
# random forest regression PCA
RF_model2<-randomForest(Share ~ ., data = MVP_train_RFdf2)
importance(RF_model2)
varImpPlot(RF_model2)

pred2=predict(RF_model2,data =MVP_valid_RFdf2)
#library(ROCR)
MVP_valid_RFdf2$rf_predict2 <- 0
MVP_valid_RFdf2$rf_predict2[which(MVP_valid_RFdf2$rf_predict2==0)] <- p
red2



# evaluating predictive performance
library(gains)
rf_gain2 <- gains(MVP_valid_RFdf2$Share, MVP_valid_RFdf2$rf_predict2)
options(scipen=999)
rf_share2 <- (MVP_valid_RFdf2$Share)

# Lift chart
par(pty="s")
plot(c(0,rf_gain2$cume.pct.of.total*sum(rf_share2))~c(0,rf_gain2$cume.o
bs),
     xlab = "# cases", ylab = "Cumulative Share", main = "random forest
 (PCA) Lift Chart", type = "l", col = "blue")
# baseline
lines(c(0,sum(rf_share2))~c(0,dim(MVP_valid_RFdf2)[1]), col = "purple",
 lty = 2)

# Decile-wise lift chart
barplot(rf_gain2$mean.resp/mean(rf_share2), names.arg = rf_gain2$depth,

        xlab="Percentile", ylab = "Mean Share", main = "random forest
(PCA) Decile-wise Lift Chart", col="pink")

# rmse
library(Metrics)
RMSE <- rmse(MVP_valid_RFdf2$Share, MVP_valid_RFdf2$rf_predict2)
RMSE
# MAE
```

```
MAE <- mae(MVP_valid_RFdf2$Share, MVP_valid_RFdf2$rf_predict2)
MAE
# MAPE
library(MLmetrics)
MAPE <- MAPE(y_pred = MVP_valid_RFdf2$Share, y_true = MVP_valid_RFdf2$r
f_predict2)
MAPE
```

neural network

```
MVP_train_nndf1 <- MVP_train_df
MVP_valid_nndf1 <- MVP_valid_df

set.seed(100)
# neural network
library("neuralnet")
NN_model1<-neuralnet(Share ~ ., data = MVP_train_nndf1,hidden = 3, thre
shold = 0.01)
pred1 <- compute(NN_model1,covariate = MVP_valid_nndf1[,-1] )
MVP_valid_nndf1$nn_predict1 <- 0
MVP_valid_nndf1$nn_predict1[which(MVP_valid_nndf1$nn_predict1==0)] <- p
red1$net.result

plot(NN_model1, main = " Neural Network")

# evaluating predictive performance
library(gains)
nn_gain1 <- gains(MVP_valid_nndf1$Share, MVP_valid_nndf1$nn_predict1)
options(scipen=999)
nn_share1 <- (MVP_valid_nndf1$Share)

# Lift chart
par(pty="s")
plot(c(0,nn_gain1$cume.pct.of.total*sum(nn_share1))~c(0,nn_gain1$cume.o
bs),
     xlab = "# cases", ylab = "Cumulative Share", main = "neural networ
k Lift Chart", type = "l", col = "blue")
# baseline
lines(c(0,sum(nn_share1))~c(0,dim(MVP_valid_nndf1)[1]), col = "purple",
 lty = 2)

# Decile-wise lift chart
barplot(nn_gain1$mean.resp/mean(nn_share1), names.arg = nn_gain1$depth,

        xlab="Percentile", ylab = "Mean Share", main = "neural network
Decile-wise Lift Chart", col="pink")

# rmse
library(Metrics)
```

```
RMSE <- rmse(MVP_valid_nndf1$Share, MVP_valid_nndf1$nn_predict1)
RMSE
# MAE
MAE <- mae(MVP_valid_nndf1$Share, MVP_valid_nndf1$nn_predict1)
MAE

library(MLmetrics)
# MAPE
MAPE <- MAPE(y_pred = MVP_valid_nndf1$Share, y_true = MVP_valid_nndf1$n
n_predict1)
MAPE
```

neural network (PCA)

```
MVP_train_nndf2 <- MVP_train_pca_df
MVP_valid_nndf2 <- MVP_valid_pca_df

set.seed(100)
# neural network(PCA)
NN_model2<-neuralnet(Share ~ ., data = MVP_train_nndf2,hidden = 3, thre
shold = 0.01)
pred2 <- compute(NN_model2,covariate = MVP_valid_nndf2[,-1] )
MVP_valid_nndf2$nn_predict2 <- 0
MVP_valid_nndf2$nn_predict2[which(MVP_valid_nndf2$nn_predict2==0)] <- p
red2$net.result

plot(NN_model2, main = " Neural Network (PCA)")
# evaluating predictive performance
library(gains)
nn_gain2 <- gains(MVP_valid_nndf2$Share, MVP_valid_nndf2$nn_predict2)
options(scipen=999)
nn_share2 <- (MVP_valid_nndf2$Share)

# Lift chart
par(pty="s")
plot(c(0,nn_gain2$cume.pct.of.total*sum(nn_share2))~c(0,nn_gain2$cume.o
bs),
     xlab = "# cases", ylab = "Cumulative Share", main = "neural networ
k (PCA) Lift Chart", type = "l", col = "blue")
# baseline
lines(c(0,sum(nn_share2))~c(0,dim(MVP_valid_nndf2)[1]), col = "purple",
 lty = 2)

# Decile-wise lift chart
barplot(nn_gain2$mean.resp/mean(nn_share2), names.arg = nn_gain2$depth,

        xlab="Percentile", ylab = "Mean Share", main = "neural network
(PCA) Decile-wise Lift Chart", col="pink")
```

```
# rmse
library(Metrics)
RMSE <- rmse(MVP_valid_nndf2$Share, MVP_valid_nndf2$nn_predict2)
RMSE
# MAE
MAE <- mae(MVP_valid_nndf2$Share, MVP_valid_nndf2$nn_predict2)
MAE
# MAPE
library(MLmetrics)
MAPE <- MAPE(y_pred = MVP_valid_nndf2$Share, y_true = MVP_valid_nndf2$n
n_predict2)
MAPE
```

SVM (SVR)

```
library(e1071)
svm_model1 <- svm(MVP_train_df[,2:12], MVP_train_df[,1])
summary(svm_model1)
svm_predict1 <- predict(svm_model1, MVP_valid_df[,2:12])

# evaluating predictive performance
library(gains)
svm_gain1 <- gains(MVP_valid_df$Share, svm_predict1)
options(scipen=999)
svm_share1 <- (MVP_valid_df$Share)

# Lift chart
par(pty="s")
plot(c(0,svm_gain1$cume.pct.of.total*sum(svm_share1))~c(0,svm_gain1$cum
e.obs),
     xlab = "# cases", ylab = "Cumulative Share", main = "SVR Lift Char
t", type = "l", col = "blue")
# baseline
lines(c(0,sum(svm_share1))~c(0,dim(MVP_valid_df)[1]), col = "purple", l
ty = 2)

# Decile-wise lift chart
barplot(svm_gain1$mean.resp/mean(svm_share1), names.arg = svm_gain1$dep
th,
        xlab="Percentile", ylab = "Mean Share", main = "SVR Decile-wise
 Lift Chart", col="pink")

# RMSE
library(Metrics)
RMSE <- rmse(MVP_valid_df$Share, svm_predict1)
RMSE
# MAE
MAE <- mae(MVP_valid_df$Share, svm_predict1)
MAE
```

```
# MAPE
library(MLmetrics)
MAPE <- MAPE(y_pred = MVP_valid_df$Share, y_true = svm_predict1)
MAPE
```

SVM (SVR) (PCA)

```
library(e1071)
svm_model2 <- svm(MVP_train_pca_df[,2:6], MVP_train_pca_df[,1])
summary(svm_model2)
svm_predict2 <- predict(svm_model2, MVP_valid_pca_df[,2:6])

# evaluating predictive performance
library(gains)
svm_gain2 <- gains(MVP_valid_pca_df$Share, svm_predict2)
options(scipen=999)
svm_share2 <- (MVP_valid_pca_df$Share)

# Lift chart
par(pty="s")
plot(c(0,svm_gain2$cume.pct.of.total*sum(svm_share2))~c(0,svm_gain2$cum
e.obs),
     xlab = "# cases", ylab = "Cumulative Share", main = "SVR  (PCA) Li
ft Chart", type = "l", col = "blue")
# baseline
lines(c(0,sum(svm_share2))~c(0,dim(MVP_valid_df)[1]), col = "purple", l
ty = 2)

# Decile-wise lift chart
barplot(svm_gain2$mean.resp/mean(svm_share2), names.arg = svm_gain2$dep
th,
        xlab="Percentile", ylab = "Mean Share", main = "SVR (PCA) Decil
e-wise Lift Chart", col="pink")

# RMSE
library(Metrics)
RMSE <- rmse(MVP_valid_pca_df$Share, svm_predict2)
RMSE
# MAE
MAE <- mae(MVP_valid_df$Share, svm_predict2)
MAE
# MAPE
library(MLmetrics)
MAPE <- MAPE(y_pred = MVP_valid_pca_df$Share, y_true = svm_predict2)
MAPE
```