Music Trending Analysis Based on Spotify Data

1st Trent Zhang
dept. Statistics
University of Illinois at Urbana-Champaign
Urbana, IL
zz90@illinois.edu

***Abstract*—First, we tried few statistical model to know what are the largest influencers on a song's success based on the features extracted from spotify. For top 20 % popular musics, our regression model gives reasonable out-of-sample predictions: ±4.86 based on root-mean-squared-error. Then, we tried multinomial logistic regression, linear discriminative analysis, quadratic discriminative analysis, k-nearest neighbors and decision tree to classify whither a song is popular or not. The best classification model is KNN (K=2) gives out-of-sample predictive over all error rate: 0.391 based on 0-1-error, and error rate of 0.3953 on popular songs.**

***Index Terms*—Trend Prediction, Music Analysis, Regression Analysis**

I. Introduction

A. Background

In 2017, the music industry generated $8.72 billion in the United States alone. Thanks to growing streaming services (Spotify, Apple Music, etc) the industry continues to flourish. Popular songs secure the lion's share of revenue. The top 10 artists in 2016 generated a combined $362.5 million in revenue. The question of what makes a song popular has been studied before with varying degrees of success. Every song has key characteristics including lyrics, duration, artist information, temp, beat, loudness, chord, etc.

B. Problem Description

We decided to further investigate by asking three key questions:

1) Are there certain characteristics for hit songs?
2) What are the largest influencers on a song's success?
3) Can we predict the popularity of new songs?

II. Data Preparation

A. Data Source

Predicting how popular one track will be a hard task. To answer these questions, we made use of Spotify Audio Features (Audio features for 115k tracks collected from the official Spotify Web API) provided by tomigelo from kaggle [1], which covered the tracks that released between January 2018 to April 2019.

University of Central Florida

B. Data Content

This dataset has altogether 130633 row and 17 column. Each track (row) has values for artist name, track name, track id and the audio features [2] itself. Additionally, there is also a popularity feature which is changing over time so it might not be up-to-date when we access the data. All features are listed as TABLE I

TABLE I
Features of original data

	Features		Features
1	artist_name	11	loudness
2	track_id	12	mode
3	track_name	13	speechiness
4	acousticness	14	tempo
5	danceability	15	time_signature
6	duration_ms	16	valence
7	energy	17	popularity
8	instrumentalness		
9	key		
10	liveness		

C. Data Split

The total data is randomly split into train data (70%) and validation data (30%).

III. Descriptive Analysis

We depict scatter, density and correlation plot of our numeric variables as Fig. 1, and have some insights from it as follows:

- The more energy, the more loudness;
- Most of popular tracks are loud;
- Most of tracks are non-live, loud, non-speech, 4/4 time signature;
- The popularity data is significantly right skewed.

IV. Regressive Model Analysis of Top 20% Popular Song

In this section, in order to know what are the largest influencers on a song's success. We discuss the results of the multiple regression model, ridge model, Lasso model and elastic-net model based on Top 20% Popular Song. Cross validation is used to evaluate the result of model.

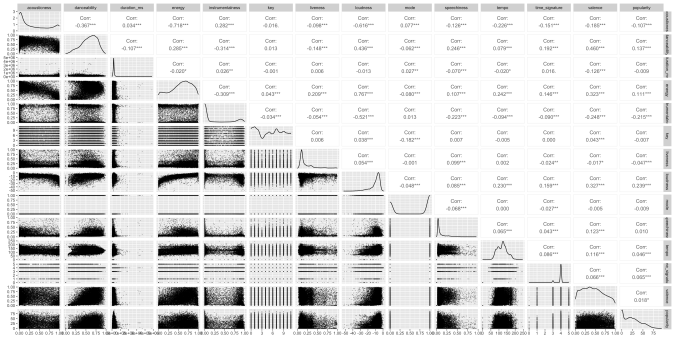

Fig. 1. Scatter, density and correlation plots

A. The Multiple Regression Model

We use linear regression with all variable and get a model with RMSE as TABLE III. Fig. 2 shows the prediction versus the observation when we use all train data to fit the model. Fig. 3 shows the prediction versus the observation when we use the model to test data.

From the figure, test error and we can know that Linear regression is not good at predicting popularity of one track.

TABLE II
COEFFICIENT OF LINEAR REGRESSION

	coefficients
(Intercept)	85.10
acousticness	-0.14
danceability	4.26
duration_ms	-0.00
energy	-2.05
instrumentalness	-1.26
key	-0.07
liveness	1.50
loudness	0.01
mode	0.10
speechiness	0.78
tempo	-0.02
time_signature	0.53
valence	-0.00

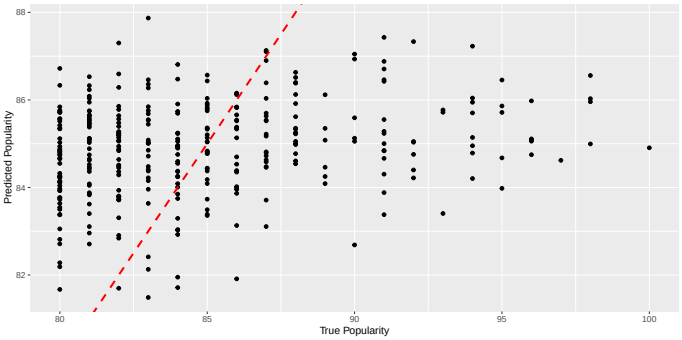

Fig. 2. Prediction Versus The Observation in train

B. Ridge with Cross Validation

But we noticed that there are many variables in this model and the some of the variable is statistically insignificant.

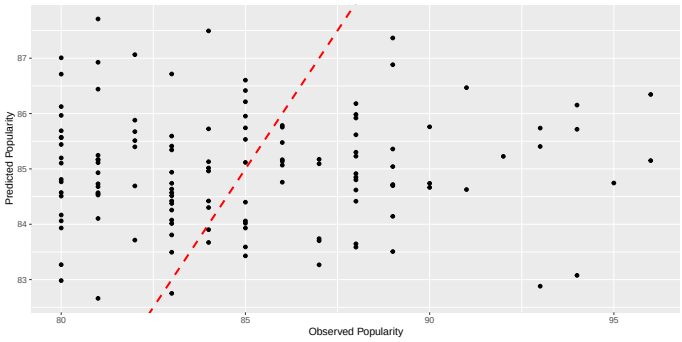

Fig. 3. Prediction Versus The Observation in test

TABLE III
RMSE TABLE OF LINEAR REGRESSION

rmse.lm.train	rmse.lm.test
4.04	4.86

Therefore, variable selection is needed.

We first use Ridge regression. The result is shown as Fig. 4. From the result, we can see that the error bar range is Large and the minimum RMSE is at $\log \lambda = -1.65$. Obviously, the ridge regression makes substantial improvements in improving test RMSE compared to the linear regression model which mainly because of the penalty.

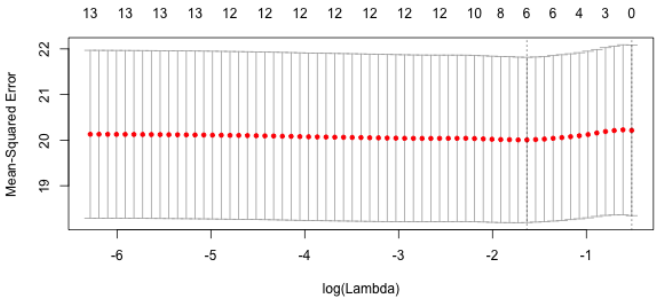

Fig. 4. Total Number of Coefficient and RMSE in Ridge[1]

C. LASSO with Cross Validation

LASSO is a regression analysis method which performs both variable selection and regularization in order to improve prediction accuracy and interpretability of the statistical model it produces [3]. By LASSO we can greatly shrink the number of variable and balance the accuracy of the model.However, there is a parameter λ needed to tune, in order to make the objectivity of choosing λ, here we use 10th Cross Validation [4].

[1]Red point is the mean RMSE of 10 Cross Validations, gery error bar is the uppermost and lowermost RMSE of 10 fold Cross Validation, the two dashed lines are value of lambda that gives minimum RMSE and largest value of lambda such that error is within 1 standard error of the minimum. The following similar figures' elements are the same as this figure.

TABLE IV
COEFFICIENT OF RIDGE REGRESSION

	s0
acousticness	-0.06
danceability	1.86
duration_ms	-0.00
energy	-0.73
instrumentalness	-0.87
key	-0.02
liveness	0.54
loudness	-0.03
mode	-0.01
speechiness	0.35
tempo	-0.01
time_signature	0.27
valence	0.10

The result is shown as Fig. 5. From the result, we can see that the error bar range is Large and the minimum RMSE is at $\log \lambda = -1.65$. Obviously, the LASSO makes substantial improvements in improving test RMSE compared to the linear regression model which mainly because of the penalty. Besides, LASSO can also make some coefficient to 0. From the coefficient TABLE V we can see the following imformation:

1) The higher danceability, the higher popularity;
2) The lower energy, the higher popularity.

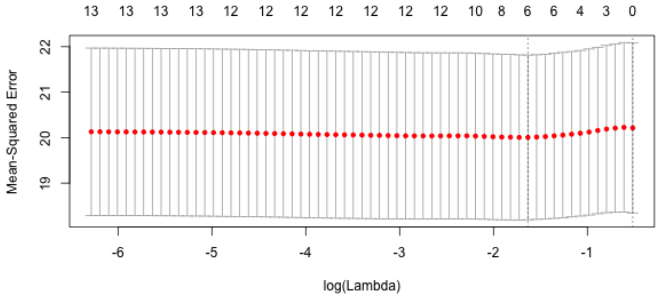

Fig. 5. Total Number of Coefficient and RMSE in Lasso

TABLE V
COEFFICIENT OF LASSO REGRESSION

	s0
acousticness	0.00
danceability	2.90
duration_ms	-0.00
energy	-0.58
instrumentalness	0.00
key	-0.00
liveness	0.00
loudness	0.00
mode	0.00
speechiness	0.00
tempo	-0.02
time_signature	0.00
valence	0.00

D. Elastic-net with Cross Validation

Elastic Net first emerged as a result of critique on lasso, whose variable selection can be too dependent on data and thus unstable. The solution is to combine the penalties of ridge regression and lasso to get the best of both worlds. Elastic Net aims at minimizing the following loss function:

$$\frac{\sum_{i-1}^{n}\left(y_i - x_i'\hat{\beta}\right)^2}{2n} + \lambda\left(\frac{1-\alpha}{2}\sum_{j=1}^{m}\hat{\beta}_j^2 + \alpha\sum_{j=1}^{m}\left|\hat{\beta}_j\right|\right) \quad (1)$$

where α is the mixing parameter between ridge ($\alpha = 0$) and lasso ($\alpha = 1$).

The result is shown as Fig. 6. From the result, we can see that the best Elastic-net is similar to LASSO model.

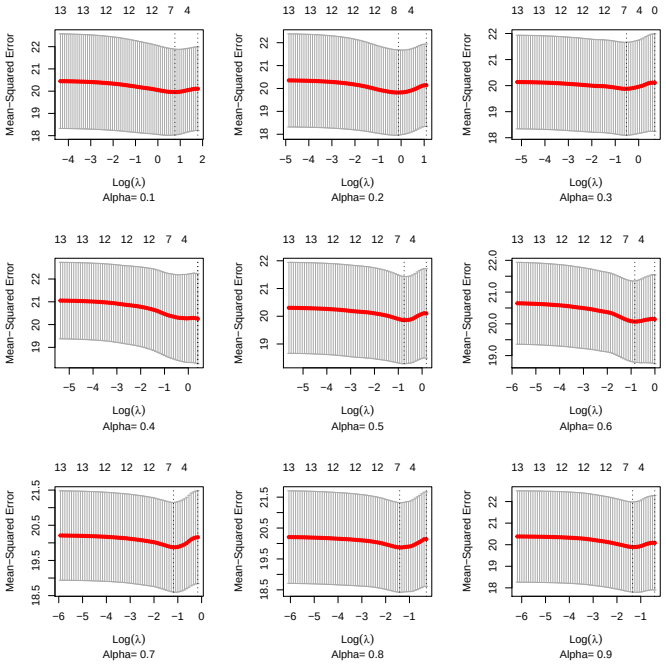

Fig. 6. Total Number of Coefficient and RMSE in Elastic-net

TABLE VI
COEFFICIENT OF ELASTIC-NET REGRESSION

	Coefficient
acousticness	0.00
danceability	2.34
duration_ms	-0.00
energy	-0.25
instrumentalness	0.00
key	0.00
liveness	0.00
loudness	0.00
mode	0.00
speechiness	0.00
tempo	-0.01
time_signature	0.00
valence	0.00

V. Classificative Model Analysis

In this chapter, we use multinomial logistic regression, linear discriminative analysis, quadratic discriminative analysis, tree method, random forest and k-nearest neighbors [5] to classify our data. All model result is trained by 10-fold cross validation within the train data. Before classification, we divide the popularity data to 3 classes as TABLE VII.

TABLE VII
DIVID THE DATA TO 3 CLASSES

	Unpopular	Normal	Popular
Popularity	0-40	40-70	70-100
Train Count	26621	63406	1437

In this table, we noticed that our data is highly imbalanced, if we use this data directly, our classification model might be highly biased towards the category which has the highest samples. Hence, we balanced the data, the way we use to balance the data subsample.

TABLE VIII
BALANCED DATA

	Unpopular	Normal	Popular
Popularity	0-40	40-70	70-100
Train Count	1437	1437	1437

A. Multinomial Logistic Regression

Logistic regression is commonly used in classic classification analysis, since we have multiple class in the data, multinomial logistic regression is introduced here.

The error analysis plot is depict as Fig.7. From error analysis plot we can see that the picked model's error rate is 0.54. The popular song is well predicted, however, normal song and unpopular song are bad predicted.

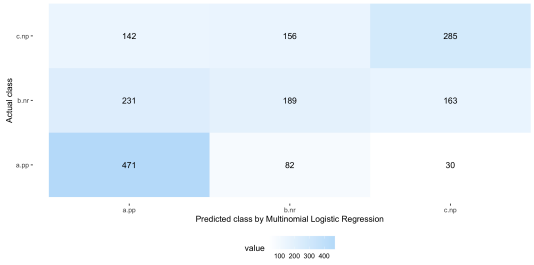

Fig. 7. Error Analysis[2]Of Multinomial Logistic Regression

B. Linear Discriminative Analysis (LDA)

After modeling LDA, we depicted and error analysis plot as Fig.8. From error analysis plot we can see that the picked model's error rate is 0.516 which is almost the same as multinomial logistic regression. The prediction is almost no improvement, which means that both multinomial logistic regression and LDA can't greatly classify the data.

As we all know, these two way generates a linear boundary of classification, however, when the real boundary is non-linear these 2 ways can't produce a great result. So we next try some non-linear boundary models.

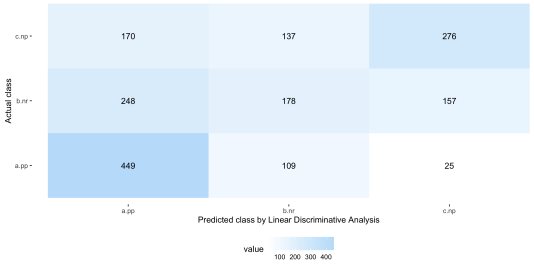

Fig. 8. Error Analysis Of Linear Discriminative Analysis

C. Quadratic Discriminative Analysis (QDA)

In order to model a non-linear boundary, naturally, QDA comes to our mind. Since QDA assumes a quadratic decision boundary, it can accurately model a wider range of problems than the linear methods.

We train the data with QDA model, and depicted error analysis plot as Fig.9. From error analysis plot we can see that the picked model's error rate is 0.516. The overall prediction is the same as LDA which means that QDA is no better than previous ways.

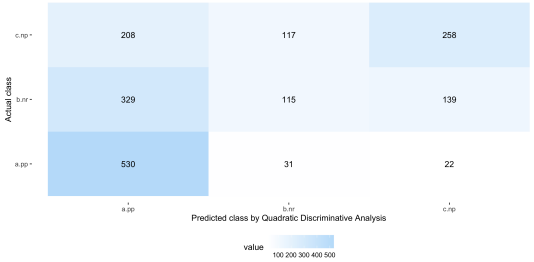

Fig. 9. Error Analysis Of Quadratic Discriminative Analysis

D. Decision Tree

Besides, we train the data with Decision Tree model. Then we depict error analysis plot as Fig.10. From error analysis plot we can see that the picked model's error rate is 0.509 and it only predicted popular class greatly. Obviously, Decision Tree have no significant improvement than previous ways.

E. K-Nearest Neighbors (KNN)

What if we use an even more flexible method than QDA? Will the result be better? Hence, KNN is introduced as our last classification method. We train the data with KNN model, get the best $k = 2$ by cross validation. Then we depict error analysis plot as Fig.11. From error analysis plot we can see that the picked model's error is 0.391, and error rate on popular

[2]The black line in the left figure is the average line of 10 accuracies in cross validation, and the overall accuracy in the right figure is from the best model picked from cross validation. All following similar figure are the same.

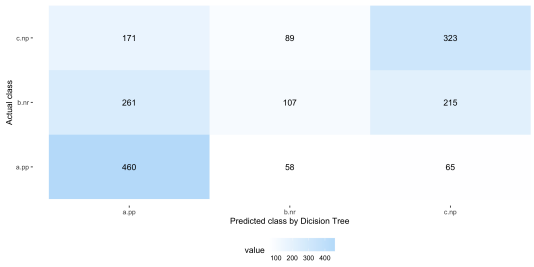

Fig. 10. Error Analysis Of Decision Tree Analysis

songs is 0.3953. The error rate has been decreased by 10% than prior methods, however, the accuracy is still very low.

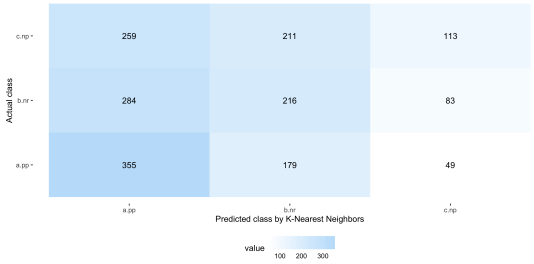

Fig. 11. Error Analysis Of 2-Nearest Neighbors Analysis

VI. Conclusion

Going into this endeavour, we were uncertain if it is even possible to predict, better than random, if a song will be popular or not. After testing our model on new songs pulling from Spotify, we observe that it is significantly simpler to correctly predict a bad song rather than a hit. It may have been easier to predict non hit songs because our data was skewed, with limited hit songs. Besides, We didn't consider lyrics, to predict a song's popularity.

From the result, we can see that all regression model are not so good. The error bar range in Fig. 4, Fig. 5 and Fig. 6 is large.

VII. Futrue works

Moving forward, we would like to explore how additional features such as lyrics, artist popularity, artist location or release date can influence a song's popularity. In addition, we may consider using the full dataset to see if we can improve our models. Another alternative is to use Spotify API to collect our own data.

References

[1] Tomigelo, "Spotify audio features," Dec 2018. [Online]. Available: https://www.kaggle.com/tomigelo/spotify-audio-features/version/1

[2] "Get audio features for a track." [Online]. Available: https://developer.spotify.com/documentation/web-api/reference/tracks/get-audio-features/

[3] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 58, no. 1, pp. 267–288, 1996.

[4] R. Kohavi *et al.*, "A study of cross-validation and bootstrap for accuracy estimation and model selection," in *Ijcai*, vol. 14, no. 2. Montreal, Canada, 1995, pp. 1137–1145.

[5] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An introduction to statistical learning*. Springer, 2013, vol. 112.

Appendix A
R Source Code

```
library(glmnet)
library(ggplot2)
library(Metrics)
library(FNN)
library(caret)
library(gridExtra)
library(xtable)

rm(list = ls())
set.seed(5665)

error_analysis <- function(pred,actual, plot_name) {
  # pred = predict(model, DT.test)
  error = ifelse(pred == actual, 1, 0)
  table.error = table(data.frame(pred, actual))
  table.error.rate = table.error
  errorrate = sum(error) / length(error)
  t = reshape2::melt(table.error.rate)
  ep = ggplot(t, aes(t[, 1], t[, 2], fill = value, label = round(value, 3))) +
    # x and y axes => Var1 and Var2
    geom_tile() + # background colours are mapped according to the
      value column
    geom_text() +
    scale_fill_continuous(high = "#c0e1fa", low = "#ffffff") +
    theme(legend.position = "bottom",
          panel.background = element_rect(fill = "white")) +
    scale_x_discrete(label = abbreviate) + scale_y_discrete(label =
      abbreviate) +
    xlab(paste("Predicted class by", plot_name)) + ylab("Actual class")
  # +ggtitle(paste("Overall Accuracy=", round(model$results$Accuracy,
    4)))
  # cvep = ggplot(model$resample, mapping = aes(Resample, Accuracy)
    ) +
  # geom_point() +
  # geom_hline(yintercept = mean(model$resample$Accuracy))
  summa = list()
  summa[["error"]] = error
  summa[["table.error"]] = table.error
  summa[["table.error.rate"]] = table.error.rate
  summa[["errorrate"]] = errorrate
  ggsave(
    file = paste("error", gsub(" ", "", plot_name), '.pdf', sep = ''),
    path = "Thesis/figure",
    # grid.arrange(cvep, ep, ncol = 2),
    ep,
    width = 10,
    height = 5
  )
  return(summa)
}


# r.dt=data.table::fread("/Users/Trent/Downloads/track_features/tf_
    000000000000.csv", header = T, sep = ',')
# select_index=sample(nrow(r.dt),100000)
# dt = as.data.frame(r.dt[select_index,])
# rownames(dt.train) <- NULL

# dt=read.csv("data/track_features/tf_mini.csv")
# dt=dt[,2:30]
# index.pop=which(dt[,3]>98)
# dt=dt[index.pop,]
# index.year=which(dt[,2]==2018)
# dt=dt[index.year,c(1,3:29)]

t = read.csv("data/SpotifyAudioFeaturesApril2019.csv")
data.nnum = t[, 1:3]
data = t[, 4:17]

train = sample(nrow(data), 0.7 * nrow(data))
data.train = data[train,]
data.test = data[-train,]

ggplot(data, aes(popularity)) + geom_histogram(bins=40)

library(GGally)
temp=ggpairs(data[sample(nrow(data), 0.1 * nrow(data)), ], lower = list(
    continuous = wrap(
  "points", alpha = 0.3, size = 0.1
)))
ggsave(file = 'ggpairs.png',
       path = "Thesis/figure",
       temp,
       width = 20,
       height = 10
)

#regression of popular type
dt.train = data.train[which(data.train$popularity>79),]
dt.test = data.test[which(data.test$popularity>79),]

lm1 = lm(popularity ~ ., data = dt.train)
summary(lm1)
x = dt.train$popularity
y = lm1[["fitted.values"]]
predict_and_observe_plot_data = data.frame(x, y)
p = ggplot(data = predict_and_observe_plot_data, aes(x, y)) +
  geom_point() +
  geom_abline(
    intercept = 0,
    slope = 1,
    color = "red",
    linetype = "dashed",
    size = 1
  ) +
  xlab("True Popularity") +
  ylab("Predicted Popularity")
ggsave(
  file = 'predict_and_observe_plot.pdf',
  path = "Thesis/figure",
  gridExtra::arrangeGrob(p),
  width = 10,
  height = 5
)
rmse.lm.train = rmse(dt.train$popularity, lm1[["fitted.values"]])
error.lm.train = dt.train$popularity - lm1[["fitted.values"]]


pred.lm = predict(lm1, dt.test[, -14])
x = dt.test$popularity
y = pred.lm
predict_and_observe_plot_data = data.frame(x, y)
p = ggplot(data = predict_and_observe_plot_data, aes(x, y)) +
  geom_point() +
  geom_abline(
    intercept = 0,
    slope = 1,
    color = "red",
    linetype = "dashed",
    size = 1
  ) +
  xlab("Observed Popularity") +
  ylab("Predicted Popularity")
ggsave(
  file = 'testerror.pdf',
  path = "Thesis/figure",
  gridExtra::arrangeGrob(p),
  width = 10,
  height = 5
)
rmse.lm.test = rmse(dt.test$popularity, pred.lm)
error.lm.test = dt.test$popularity - pred.lm


```

```
x = as.matrix(dt.train[,-14])
y = as.matrix(dt.train$popularity)
x.t = as.matrix(dt.test[,-14])
y.t = as.matrix(dt.test$popularity)

cvfit.ridge = cv.glmnet(x, y, alpha = 0)
pdf('Thesis/figure/cvfitridge.pdf',width = 600, height = 300)
plot(cvfit.ridge)
dev.off()
xtable(as.array(glmnet(x, y, alpha = 0, lambda = cvfit.ridge$lambda.min)$beta))

cvfit.lasso = cv.glmnet(x, y, alpha = 1)
pdf('Thesis/figure/cvfitlasso.pdf',width = 600, height = 300)
plot(cvfit.lasso)
dev.off()
xtable(as.array(glmnet(x, y, alpha = 1, lambda = cvfit.lasso$lambda.min)$beta))


pdf('Thesis/figure/cvfitpla_net.pdf')
par(mfrow=c(3,3))
pla_net=function(k){
  cvfit.net = cv.glmnet(x, y, alpha=k)
  plot(cvfit.net, sub=paste("Alpha=",k))
}
for (k in 0.1*1:9) {
  pla_net(k)
}
dev.off()
xtable(as.array(glmnet(x, y, alpha = 0.5, lambda = 0.5)$beta))

f_knn_tst_rmse = function(k) {
  rmse(y.t, knn.reg(x, x.t, y, k)$pred)
}
f_knn_trn_rmse = function(k) {
  rmse(y, knn.reg(x, x, y, k)$pred)
}
# k = c(1, 5, 10, 25, 50, 250)
k = c(1, 3:10)
knn_tst_rmse = sapply(k, f_knn_tst_rmse)
knn_trn_rmse = sapply(k, f_knn_trn_rmse)
# determine "best" k
best_k = k[which.min(knn_tst_rmse)]
knn_results = data.frame(k,
                         round(knn_trn_rmse, 2),
                         round(knn_tst_rmse, 2))
colnames(knn_results) = c("k", "Train RMSE", "Test RMSE")

pred.best.knn = knn.reg(x, x.t, y, 25)$pred
error.best.knn = y.t - pred.best.knn
plot(error.best.knn)




########################classification
DT.train = data.train
DT.test = data.test
DT.train$popularity = as.factor(ifelse(
  DT.train$popularity < 10,
  "c.unpopular",
  ifelse(
    DT.train$popularity < 40,
    "b.normal",
    ifelse(DT.train$popularity < 70, "b.normal", "a.popular")
  )
))
DT.test$popularity = as.factor(ifelse(
  DT.test$popularity < 10,
  "c.unpopular",
  ifelse(
    DT.test$popularity < 40,
    "b.normal",
    ifelse(DT.test$popularity < 70, "b.normal", "a.popular")
  )
))
#c.unpopular
table(DT.test$popularity)
table(DT.train$popularity)
# data is not balanced, we first balance our train data
DT.test=caret::downSample(x=DT.test[, -ncol(DT.test)],y=DT.test$popularity,yname="popularity")
DT.train=caret::downSample(x=DT.train[, -ncol(DT.train)],y=DT.train$popularity,yname="popularity")

table(DT.test$popularity)
table(DT.train$popularity)
#############Logistic Regression
ctrl = trainControl(method = "cv", 3)
model.multinom = train(
  popularity ~ .,
  data = DT.train,
  method = "multinom",
  trControl = ctrl,
  preProcess = c("center", "scale")
)
pred.multinom = predict(model.multinom, DT.test, type = "prob")
error.multinom = error_analysis(predict(model.multinom, DT.test),DT.test[, 14], plot_name = "Multinomial Logistic Regression")
error.multinom
#############LDA
ctrl = trainControl(method = "cv", 10)
model.lda = train(
  popularity ~ .,
  data = DT.train,
  method = "lda",
  trControl = ctrl,
  preProcess = c("center", "scale")
)
pred.lda = predict(model.lda, DT.test, type = "prob")
error.lda = error_analysis(predict(model.lda, DT.test),DT.test[, 14], plot_name = "Linear Discriminative Analysis")
error.lda
#############QDA
ctrl = trainControl(method = "cv", 10)
model.qda = train(
  popularity ~ .,
  data = DT.train,
  method = "qda",
  trControl = ctrl,
  preProcess = c("center", "scale")
)
pred.qda = predict(model.qda, DT.test, type = "prob")
error.qda = error_analysis(predict(model.qda, DT.test),DT.test[, 14], plot_name = "Quadratic Discriminative Analysis")
error.qda
#############KNN
pred.knn = knn(DT.train[, 1:13], DT.test[, 1:13], DT.train[, 14], k = 2, prob = TRUE)
error.knn = error_analysis(as.vector(pred.knn),DT.test[, 14], plot_name = "K-Nearest Neighbors")
error.knn

error.knn$table.error

f_knn = function(k) {
  error.knn =error_analysis(as.vector(knn(DT.train[, 1:13], DT.test[, 1:13], DT.train[, 14], k, prob = TRUE)),DT.test[, 14], plot_name = "K-Nearest Neighbors")
  print(error.knn$table.error)
}
k = c(1:10)
sapply(k, f_knn)

View(t[row.names(DT.test[which(as.vector(pred.knn) != DT.test$popularity & DT.test$popularity == "c.unpopular"), ]),c(1,3,17)
```

```
    ↪ ])
############Dicision tree
model.tree = rpart::rpart(popularity ~ ., data = DT.train)
pred.tree = predict(model.tree, DT.test[, 1:13], type = "class")
error.tree = error_analysis(as.vector(pred.tree),DT.test[, 14], plot_name =
    ↪ "Dicision Tree")
error.tree
############Random forest
model.forest= randomForest::randomForest(popularity ~., data=DT.train)
pred.forest= predict(model.forest, DT.test[,1:13], type = "class")
error.forest=error_analysis(as.vector(pred.tree),DT.test[, 14],plot_name="
    ↪ Random Forest")
error.forest
```

../Code/Final project.R