TITLE: IMDb Score Prediction using Data Science

PHASE 5: project documentation and submission

Introduction :

IMDb scores are determined by user ratings and can change overtime as more users rate the movie or show.

The problem is to develop a machine learning model to predict the IMDb scores of movies available on Films based on their genre, premiere date, runtime, and language.

This project involves data collection, data prepossessing, feature engineering, clustering algorithms, visualization, and interpretation of results.

The model aims to accurately estimate the popularity of movies to assist users in discovering highly rated films that align with their preferences.

Problem definition :

The goal of this problem is to develop a predictive model that can accurately estimate the IMDb (Internet Movie Database) scores for a given set of movies. IMDb scores, often referred to as IMDb ratings, are a measure of the perceived quality and popularity of movies. These scores are typically on a scale of 1 to 10, with higher scores indicating better reception

Design thinking process:

1.Empathize:

- Understand the Users: Begin by understanding the needs and expectations of users who rely on IMDb scores, such as movie enthusiasts, film studios, and streaming platforms.
- Gather User Stories: Conduct interviews, surveys, and observations to collect user stories and pain points related to IMDb scores and movie recommendations.

2.Define:

- Problem Statement: Clearly define the problem you want to solve, such as "How might we predict IMDb scores more accurately to improve movie recommendations?"

- User Personas: Create user personas to represent different user groups and their specific needs

3.Ideate:

- Generate Feature Ideas: Identify potential features and data sources that could influence IMDb scores, including movie metadata, user reviews, and critic reviews.

4.Prototype:

- Create Data Pipelines: Develop data pipelines to collect, clean, and preprocess relevant data from diverse sources. This may involve scraping movie details, collecting user reviews, and aggregating critic reviews.
- Feature Engineering: Design features that capture key aspects of movies, such as genre, director, cast, release date, budget, and historical IMDb scores.
- Model Selection: Experiment with different machine learning algorithms, regression models, and deep learning approaches to predict IMDb scores. Create prototypes of these models.
- Visualization: Develop data visualizations to explore the relationships between features and IMDb scores, which can help in feature selection.

5.Test:

- Model Evaluation: Use a portion of the data to train and validate your IMDb score prediction models. Evaluate the models using metrics like MAE, MSE, RMSE, and R2.
- User Testing: Get feedback from users to understand how well the IMDb score predictions align with their expectations and preferences.
- Iterate: Make improvements based on the feedback and continue to refine the model and data pipelines

6.Implement:

- Full-Scale Deployment: Deploy the IMDb score prediction model in a real-world environment, such as a movie recommendation system.
- Monitor and Maintain: Continuously monitor the model's performance, update it with new data, and ensure it remains accurate over time.

7.Learn:

- Gather Feedback: Continue to collect feedback from users and stakeholders to identify areas for improvement.
- Adapt to Changes: Be prepared to adapt the model as movie preferences and user behavior change.

8.Scale and Iterate:

- As the IMDb score prediction system gains traction, consider scaling it to accommodate a larger user base.

- Iterate on the design thinking process to address new challenges and opportunities that may arise.

Loading the dataset:

For loading the data ,we use the dataset using the given dataset link

Dataset link: : **https://www.kaggle.com/datasets/luiscorter/netflix-original-films-imdb-scores**

Predicting IMDB Scores.

Loading the libraries

The aim of this project is to come up with a model for predicting the IMDB scores. We load the libraries we need for this.

library(ggplot2)

library(GGally)

library(dplyr)

```
##
## Attaching package: 'dplyr'
## The following object is masked from 'package:GGally':
##
##     nasa
## The following objects are masked from 'package:stats':
##
##     filter, lag
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(tree)
library(rpart)
library(rpart.plot)
```

Reading the data

We import the CSV file that has been given.

```
movie <- read.csv('../input/movie_metadata.csv',header = T,strin
gsAsFactors = F)
```

The data contains columns that are of numerical and character types. We will just concentrate on extracting the numerical variables. We store the resulting data in temp.

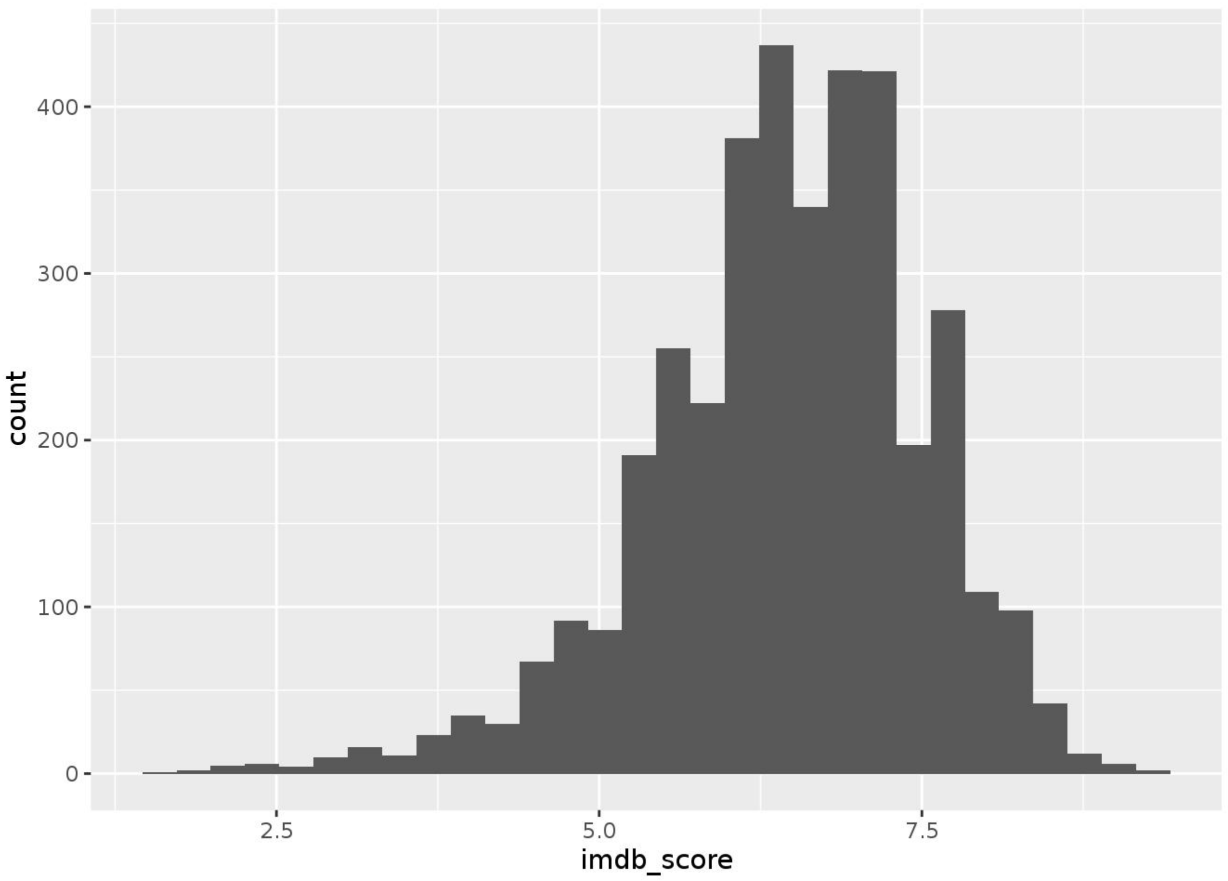
```
columns <- c()
for(i in 1:dim(movie)[2])
{
  if(is.numeric(movie[,i])|| is.integer(movie[,i]))
  {
    columns[i]=T
  }
  else
  {
    columns[i]=F
  }
}
temp <- na.omit(movie[,columns])
```

How is the response variable distributed?

This would give us an idea of the method we will need to predict our IMDB scores.

```
ggplot(temp, aes(x=imdb_score)) + geom_histogram()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidt
h`.
```



From the above plot, we see that the response variable is real valued (they are not made of whole numbers.). This would then eliminate the need for classification.

Simple Linear Model

We will start by building a simple linear model for our response variable. We do need to ask a few questions. How are the variables correlated to the response variable?

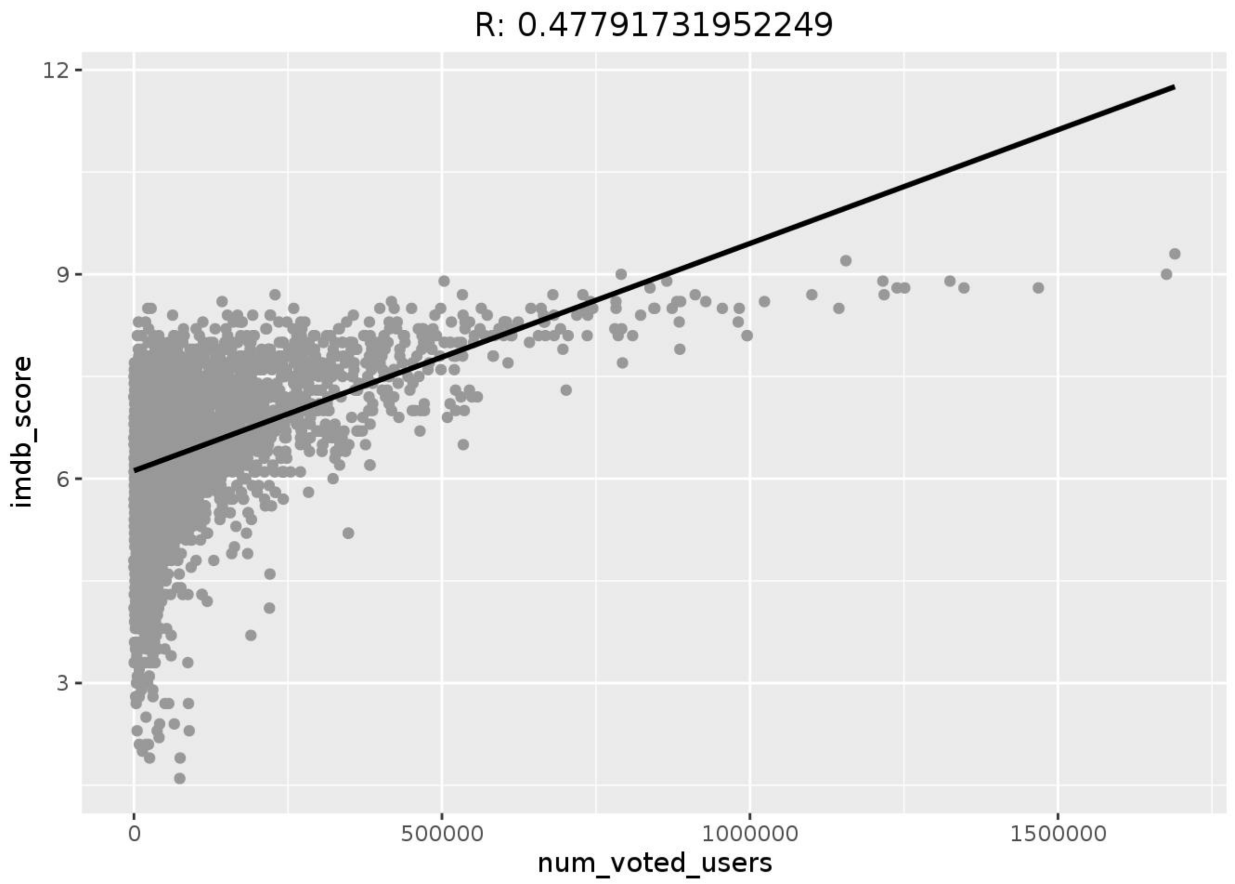
```
correlation <- c()

for(i in 1:dim(temp)[2])

{

  correlation[i] <- cor(temp[,i],temp[,'imdb_score'])
}
```

```
correlation

## [1] 0.34388077 0.36612369 0.19083814 0.06497354 0.09313142
## [6] 0.21212439 0.47791732 0.10625870 -0.06429247 0.32252237
## [11] 0.02904057 -0.12926516 0.10206038 1.00000000 0.02845372
## [16] 0.27947774
```

What are the top two variables with respect to correlation? We see that the second and the seventh variables have ‘fairly’ high correlations with our response variable. Using all the variables to build our linear model would increase the error terms.

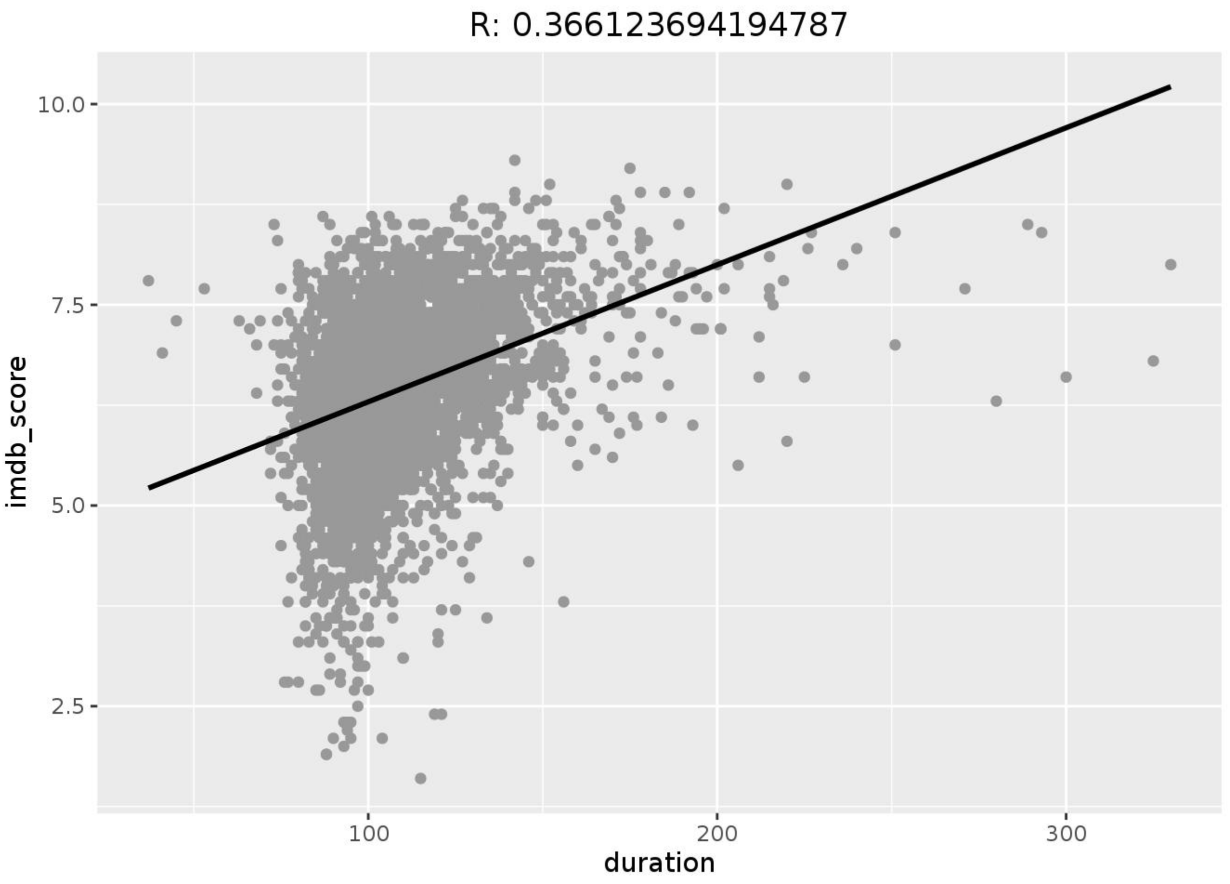
Scatter Plot to visualize correlation

```
ggplot(temp, aes(x=num_voted_users, y=imdb_score)) + geom_point(colour="grey60") +
  stat_smooth(method=lm, se=FALSE, colour="black")+ggtitle(paste('R:',correlation[7]))
```



ggplot(temp, aes(x=duration, y=imdb_score)) + geom_point(colour="grey60") +

stat_smooth(method=lm, se=FALSE, colour="black")+ggtitle(paste('R:',correlation[2]))



Linear Model

Using the number of voted users variable and the duration variable , we build our model. We split our data to a test and training set

set.seed(2)

train <- sample(dim(temp)[1],dim(temp)[1]*0.9)

temp_train <- temp[train,]

temp_test <- temp[-train,]

lmfit = lm(imdb_score~num_voted_users+duration,data=temp_train)

```
summary(lmfit)
##
## Call:
## lm(formula = imdb_score ~ num_voted_users + duration, dat
a = temp_train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.8430 -0.5103  0.1009  0.6245  2.4636
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)     5.009e+00  7.771e-02   64.46   <2e-16 ***
## num_voted_users 2.780e-06  1.078e-07   25.80   <2e-16 ***
## duration        1.067e-02  7.179e-04   14.86   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.8991 on 3417 degrees of freedom
## Multiple R-squared:  0.2739, Adjusted R-squared:  0.2734
## F-statistic: 644.4 on 2 and 3417 DF,  p-value: < 2.2e-16
```

```
pred <- predict(lmfit,temp_test)
```

The R squared metric should be close to one for this linear model to be of good quality.The R squared metric helps us assess how the model explains the variability in the data.A perfect model has an R square of 1. The model we have built has an R squared of 0.2757,which is really poor. We would expect this as the correlations of the explanatory variables were less than 0.5.

MSE

We calulate the mean squared error pertaining to this model.

```
mean((temp_test$imdb_score-pred)^2)
```

```
## [1] 0.8331187
```

Regression trees

Lets try another method to predict the IMDB scores. We will use the regression tree method here.The regression trees method , unlike linear regression do not assume linearity in the data.Visualizing the decision tree is really simple as it follows logic. We seperate our data into the training and test data sets. We apply the train set to the rpart() function.

```
library(rpart)


set.seed(3)
```

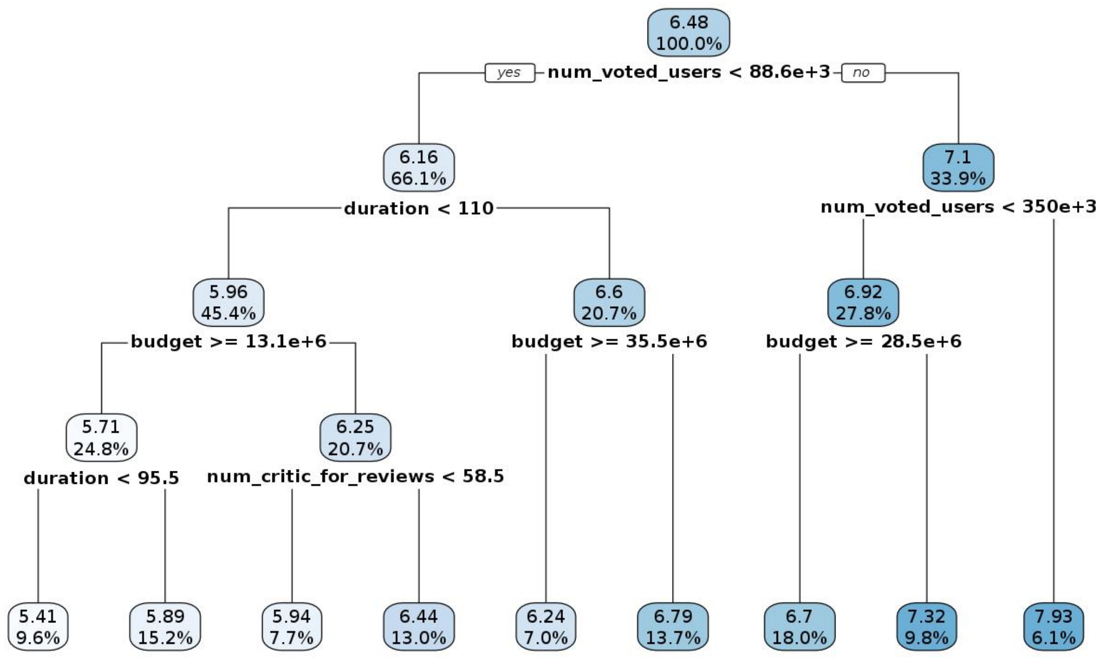
```
m.rpart <- rpart(imdb_score~.,data=temp_train)
m.rpart
## n= 3420
##
## node), split, n, deviance, yval
##       * denotes terminal node
##
##  1) root 3420 3803.67300 6.478596
##    2) num_voted_users< 88566 2261 2348.41200 6.158425
##      4) duration< 110.5 1554 1637.39400 5.955727
##        8) budget>=1.31e+07 847  787.23160 5.707556
##         16) duration< 95.5 327  368.09250 5.410703 *
##         17) duration>=95.5 520  372.20270 5.894231 *
##        9) budget< 1.31e+07 707  735.50100 6.253041
##         18) num_critic_for_reviews< 58.5 262  328.25990 5.93
5496 *
##         19) num_critic_for_reviews>=58.5 445  365.26800 6.44
0000 *
##      5) duration>=110.5 707  506.82890 6.603960
##       10) budget>=3.55e+07 239  145.68320 6.242259 *
##       11) budget< 3.55e+07 468  313.91000 6.788675 *
##    3) num_voted_users>=88566 1159  771.33820 7.103192
##      6) num_voted_users< 349779.5 950  532.38460 6.921895

##       12) budget>=2.85e+07 615  325.97350 6.703252 *
##       13) budget< 2.85e+07 335  123.03840 7.323284 *
##      7) num_voted_users>=349779.5 209   65.79455 7.927273
 *
```

From the structure we see that the variable pertaining to number of voted users is the most important followed by duration.

Visualizing the tree

The tree structure is just like a flowchart. Let' see the rules pertaining to predicting the IMDB scores.

```
rpart.plot(m.rpart,digits = 3)
```



Testing this model

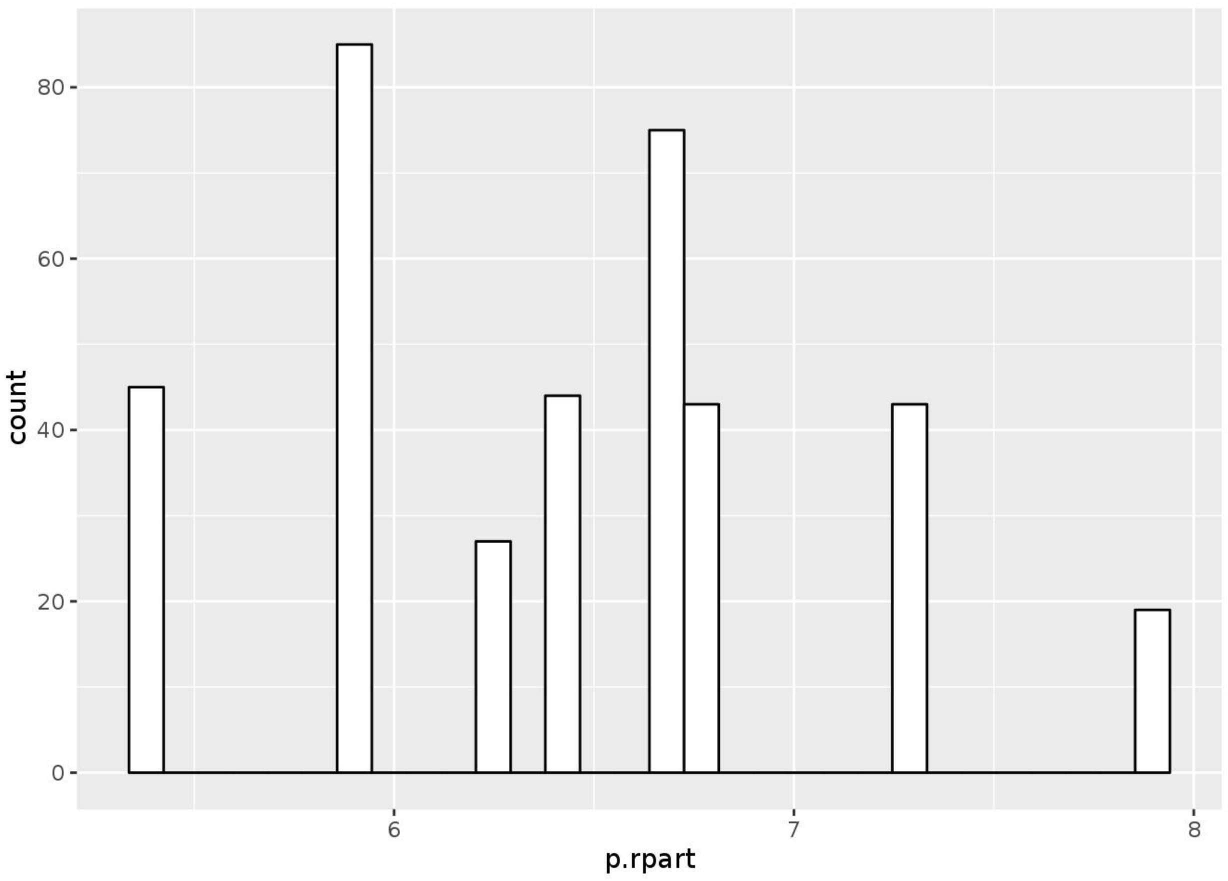
We apply this model to the test data.

```
p.rpart <- predict(m.rpart,temp_test)
```

How do the actual and predicted models differ? We visualize this by the use of histograms.

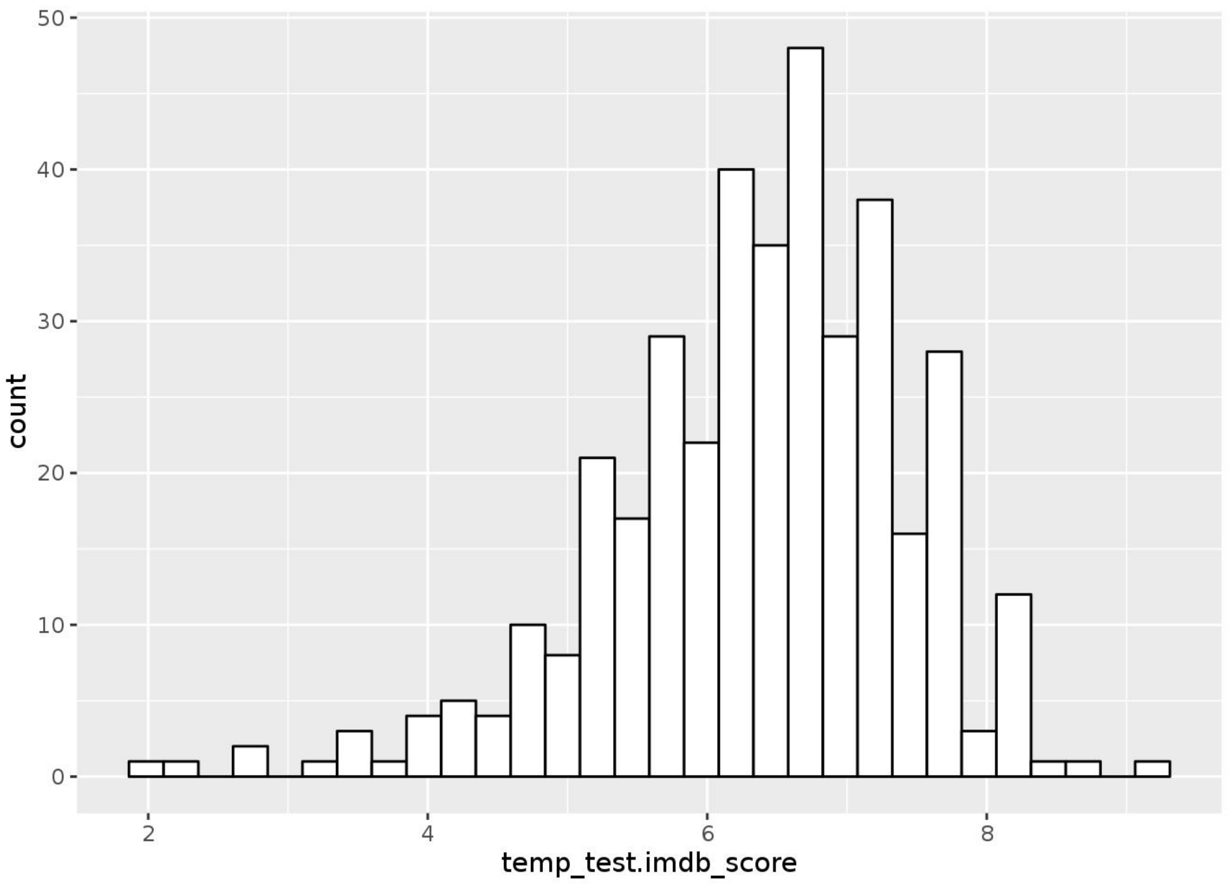
```
tree_dataframe <- data.frame(p.rpart,temp_test$imdb_score)

ggplot(tree_dataframe, aes(x=p.rpart)) + geom_histogram(fill="
white", colour="black")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidt
h`.
```



```
ggplot(tree_dataframe, aes(x=temp_test.imdb_score)) + geom_h
istogram(fill="white", colour="black")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidt
h`.
```



We see that the distributions of the predictions are in no way close to that of the distributions of the real values. What is the correlation between the predicted and the actual values?

```
cor(p.rpart,temp_test$imdb_score)
```

```
## [1] 0.5631952
```

This is a good correleation score. But it goes not give a measure of how the predicted values deviate from the actual values.

MSE

We therefore calculate the mean squared error

```
mean((p.rpart-temp_test$imdb_score)^2)
```

```
## [1] 0.8025349
```

We see a slight improvement in performance by using regression trees.

Random Forests

We use Random Forests regression to predict our IMDB score. The random forest is an ensemble based learner which uses random feature selection to add diversity to the decision tree models.

```
set.seed(5)
library(randomForest)
```

```
## randomForest 4.6-12
## Type rfNews() to see new features/changes/bug fixes.
##
## Attaching package: 'randomForest'
## The following object is masked from 'package:dplyr':
##
##     combine
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
rf <- randomForest(imdb_score~.,data=temp[train,],ntree=500,m
try=floor(dim(temp)[2]/3))

pred_rf <- predict(rf,temp[-train,])

mean((pred_rf-temp[-train,]$imdb_score)^2)

## [1] 0.4976074
```

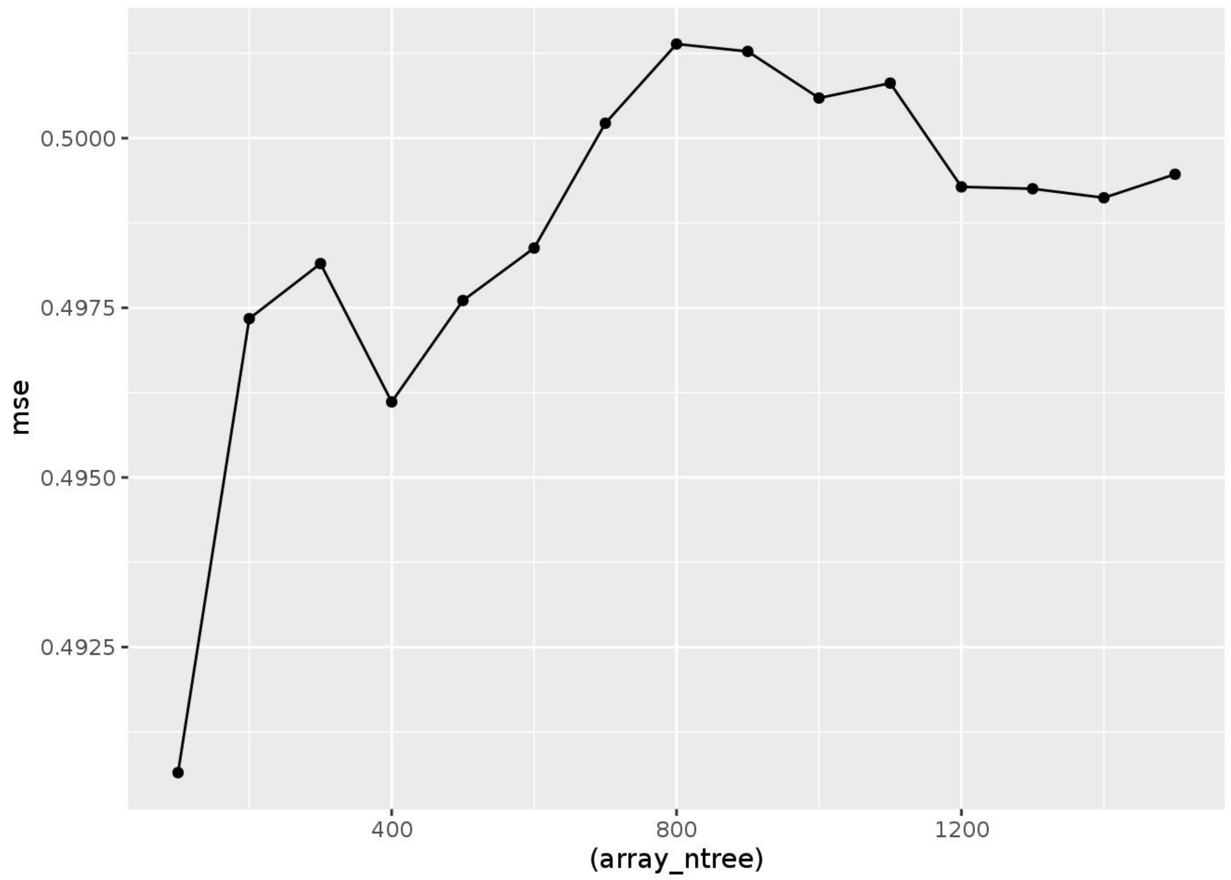
Tuning the parameter ‘ntree’

We change the ntree parameter to check which value of ntree results in a smaller MSE.

```
array_ntree<- c(100,200,300,400,500,600,700,800,900,1000,110
0,1200,1300,1400,1500)

mse <- c()

j<-1

for(i in array_ntree)

{ set.seed(5)

  rf <- randomForest(imdb_score~.,data=temp[train,],ntree=i,mtr
y=floor(dim(temp)[2]/3))

  pred_rf <- predict(rf,temp[-train,])

  mse[j]<-mean((pred_rf-temp[-train,]$imdb_score)^2)

  j=j+1


}
```

We then plot the MSE against the number of trees. We will use the value of ntree with the minimum MSE

```
data_mse <- data.frame(array_ntree,mse)

ggplot(data_mse, aes(x=(array_ntree), y=mse)) + geom_line() +
geom_point()
```



```
set.seed(5)

rf <- randomForest(imdb_score~.,data=temp[train,],ntree=data_mse$array_ntree[data_mse$mse==min(data_mse$mse)],mtry=floor(dim(temp)[2]/3))

pred_rf <- predict(rf,temp[-train,])

mean((pred_rf-temp[-train,]$imdb_score)^2)
```

```
## [1] 0.4906501
```

We see some improvement in the model by changing ntree.

Conclusion :

In conclusion, predicting IMDb scores is a complex task that involves various factors and challenges.IMDb scores are influenced by a multitude of subjective and contextual factors, and no model can perfectly capture all of these nuances.

To improve IMDb score predictions, it's crucial to consider factors such as user reviews, genre, director, actors, and release date, among others. However, it's essential to remember that IMDb scores are ultimately a reflection of audience opinions, and these opinions can change over time.

Therefore, any prediction model should be periodically updated and validated against new data.