Anime Score Prediction

Zacharias Daum

2022-11-21

Introduction and Overview

The website MyAnimeList (https://myanimelist.net/) provides information about various anime. In addition to that, it allows its users to add anime to their personal lists, indicating they have seen or want to see them. The users can also give a rating to each anime, which is represented by an integer in the range 1-10.
Based on this rating from the users, the website provides a rating score for the anime, given enough users did give a rating to a certain anime.

The goal of this project is to use data available on MyAnimeList to predict which future anime will probably receive a high score from the users.

As a basis, the dataset "Anime Database 2022" that was provided by Harits Fadlilah on kaggle (https://www.kaggle.com/datasets/harits/anime-database-2022?resource=download) was used. The data for this dataset was retrieved by web-scraping data from MyAnimeList, and contains the data available on September 20, 2022.

For this project, several different algorithms are used to predict the expected scores for anime.
To simulate doing these predictions for future anime, a validation dataset is built from all anime that were released in the years 2020-2022. The remaining anime are used as training and test data.

The models shall be trained on the training data and then a performance comparison will be made based on the test data.
Afterwards, the models will try to predict the data from the validation dataset, this is to observe if there are any issues encountered with "future" data, for example if certain algorithms perform comparatively worse on the "future" validation data than on the test data.

Methods and Analysis

Exploratory Data Analysis

Basic Data Exploration

```
head(dataset)
```

```
## # A tibble: 6 x 28
##      ID Title           Synonyms Japanese English Synopsis Type  Episodes Status
##   <dbl> <chr>           <chr>    <chr>    <chr>   <chr>    <chr>    <dbl> <chr>
## 1 16498 Shingeki no Ky~ AoT, SnK <U+9032><U+6483><U+306E>~  Attack~ "Centur~ TV         25 Finis~
## 2  1535 Death Note      DN       <U+30C7><U+30B9><U+30CE>~  Death ~ "Brutal~ TV         37 Finis~
## 3  5114 Fullmetal Alch~ Hagane ~ <U+92FC><U+306E><U+932C>~  Fullme~ "After ~ TV         64 Finis~
```

```
## 4 30276 One Punch Man   One Pun~ <U+30EF><U+30F3><U+30D1>~  One Pu~ "The se~ TV          12 Finis~
## 5 11757 Sword Art Onli~ S.A.O, ~ <U+30BD><U+30FC><U+30C9>~  Sword ~ "Ever s~ TV          25 Finis~
## 6 31964 Boku no Hero A~ Unknown  <U+50D5><U+306E><U+30D2>~  My Her~ "The ap~ TV          13 Finis~
## # ... with 19 more variables: Start_Aired <chr>, End_Aired <chr>,
## #   Premiered <chr>, Broadcast <chr>, Producers <chr>, Licensors <chr>,
## #   Studios <chr>, Source <chr>, Genres <chr>, Themes <chr>,
## #   Demographics <chr>, Duration_Minutes <dbl>, Rating <chr>, Score <dbl>,
## #   Scored_Users <dbl>, Ranked <dbl>, Popularity <dbl>, Members <dbl>,
## #   Favorites <dbl>
```

```
str(dataset)
```

```
## spec_tbl_df [21,460 x 28] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ ID              : num [1:21460] 16498 1535 5114 30276 11757 ...
##  $ Title           : chr [1:21460] "Shingeki no Kyojin" "Death Note" "Fullmetal Alchemist: Brotherhoo
##  $ Synonyms        : chr [1:21460] "AoT, SnK" "DN" "Hagane no Renkinjutsushi Fullmetal Alchemist, Ful
##  $ Japanese        : chr [1:21460] "<U+9032><U+6483><U+306E><U+5DE8><U+4EBA>" "<U+30C7><U+30B9><U+30C
##  $ English         : chr [1:21460] "Attack on Titan" "Death Note" "Fullmetal Alchemist Brotherhood" "
##  $ Synopsis        : chr [1:21460] "Centuries ago, mankind was slaughtered to near extinction by mons
##  $ Type            : chr [1:21460] "TV" "TV" "TV" "TV" ...
##  $ Episodes        : num [1:21460] 25 37 64 12 25 13 220 12 26 148 ...
##  $ Status          : chr [1:21460] "Finished Airing" "Finished Airing" "Finished Airing" "Finished Ai
##  $ Start_Aired     : chr [1:21460] "Apr 7, 2013" "Oct 4, 2006" "Apr 5, 2009" "Oct 5, 2015" ...
##  $ End_Aired       : chr [1:21460] "Sep 29, 2013" "Jun 27, 2007" "Jul 4, 2010" "Dec 21, 2015" ...
##  $ Premiered       : chr [1:21460] "Spring 2013" "Fall 2006" "Spring 2009" "Fall 2015" ...
##  $ Broadcast       : chr [1:21460] "Sundays at 0158 (JST)" "Wednesdays at 0056 (JST)" "Sundays at 170
##  $ Producers       : chr [1:21460] "Production I.G, Dentsu, Mainichi Broadcasting System, Pony Canyon
##  $ Licensors       : chr [1:21460] "Funimation" "VIZ Media" "Funimation, Aniplex of America" "VIZ Med
##  $ Studios         : chr [1:21460] "Wit Studio" "Madhouse" "Bones" "Madhouse" ...
##  $ Source          : chr [1:21460] "Manga" "Manga" "Manga" "Web manga" ...
##  $ Genres          : chr [1:21460] "Action, Drama" "Supernatural, Suspense" "Action, Adventure, Drama
##  $ Themes          : chr [1:21460] "Gore, Military, Survival" "Psychological" "Military" "Parody, Sup
##  $ Demographics    : chr [1:21460] "Shounen" "Shounen" "Shounen" "Seinen" ...
##  $ Duration_Minutes: num [1:21460] 24 23 24 24 23 24 23 24 23 23 ...
##  $ Rating          : chr [1:21460] "R - 17+ (violence & profanity)" "R - 17+ (violence & profanity)"
##  $ Score           : num [1:21460] 8.53 8.62 9.13 8.51 7.2 ...
##  $ Scored_Users    : num [1:21460] 519803 485487 900398 19066 990254 ...
##  $ Ranked          : num [1:21460] 1002 732 12 1112 29562 ...
##  $ Popularity      : num [1:21460] 1 2 3 4 5 6 7 8 9 10 ...
##  $ Members         : num [1:21460] 3524109 3504535 2978455 2879907 2813565 ...
##  $ Favorites       : num [1:21460] 155695 159701 207772 59651 64997 ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   ID = col_double(),
##   ..   Title = col_character(),
##   ..   Synonyms = col_character(),
##   ..   Japanese = col_character(),
##   ..   English = col_character(),
##   ..   Synopsis = col_character(),
##   ..   Type = col_character(),
##   ..   Episodes = col_double(),
##   ..   Status = col_character(),
##   ..   Start_Aired = col_character(),
##   ..   End_Aired = col_character(),
```

```
##   ..   Premiered = col_character(),
##   ..   Broadcast = col_character(),
##   ..   Producers = col_character(),
##   ..   Licensors = col_character(),
##   ..   Studios = col_character(),
##   ..   Source = col_character(),
##   ..   Genres = col_character(),
##   ..   Themes = col_character(),
##   ..   Demographics = col_character(),
##   ..   Duration_Minutes = col_double(),
##   ..   Rating = col_character(),
##   ..   Score = col_double(),
##   ..   Scored_Users = col_double(),
##   ..   Ranked = col_double(),
##   ..   Popularity = col_double(),
##   ..   Members = col_double(),
##   ..   Favorites = col_double()
##   .. )
##  - attr(*, "problems")=<externalptr>
```

The dataset contains 28 columns and 21460 rows.
The following columns are present:

- ID (numeric): unique identifier of an anime
- Title (character string): title of the anime
- Synonyms (character string): comma separated list of alternative titles and abbreviations of the title
- Japanese (character string): Japanese title of the anime
- English (character string): English title of the anime
- Synopsis (character string): Text describing what the anime is about, in a varyingly detailed manner
- Status (character string): String indicating whether the anime has finished airing, is currently airing, or has not yet aired
- Type (character string): type of anime, e.g. “TV”, “Movie” etc.
- Episodes (numeric): number of episodes
- Start_Aired (character string): The date when the anime first aired
- End_Aired (character string): The date when the anime ended
- Premiered (character string): Season (Spring/Summer/Fall/Winter - anime are usually released within one of these cycles) and year the anime first aired.
- Broadcast (character string): The weekday and time at which the anime was initially broadcast.
- Producers (character string): Comma separated list of producer(s)
- Licensors (character string): Comma separated list of licensor(s)
- Studios (character string): Comma separated list of animation studio(s)
- Source (character string): String representing the source material the anime is based on, e.g. “Manga”, or “Original” if there is no source material
- Genres (character string): Comma separated list of genres
- Themes (character string): Comma separated list of themes
- Demographics (character string): Target demographic
- Duration_Minutes (numeric): Episode duration in minutes
- Rating (character): Recommended age rating, e.g. “R - 17+ (violence & profanity)”
- Score (numeric): The rating score based on the scores given by all users that rated the respective anime
- Scored_Users (numeric): The number of users that rated this anime
- Ranked (numeric): Rank of the anime based on its Score
- Popularity (numeric): Rank of the anime based on the number of users that have added it to their list
- Members (numeric): Number of users who have added this anime to their list
- Favorites (numeric): Number of users who have set this anime as one of their favorites

Unneeded columns

The columns that contain alternative titles ("English", "Japanese", "Synonyms") do not seem to provide useful information for this kind of prediction.

As the scenario assumes the predictions are made for new anime, the columns that represent user feedback ("Scored_Users", "Members", "Favorites") are considered unavailable. This also applies to "Score", which is the value the models should predict.

Also, the column "Synopsis" contains a text that varies strongly in terms of detail and writing style, and this is mainly due to being written by a group of (presumably) loosely cooperation people and at most vaguely related to the anime itself. Thus, it seems questionable if it would provide useful insights for this problem, even with an approach like sentiment analysis.

While the column "End_Aired" may provide minor additional information, in most cases it should be mostly redundant with the information available through "Start_Aired" and "Episodes", considering most anime usually air one episode per week.

NA values

```
# find columns with na
names(which(sapply(dataset, anyNA)))
```

```
## [1] "Synopsis"         "Episodes"         "Duration_Minutes" "Score"
## [5] "Scored_Users"     "Ranked"
```

The above columns contained some NA values, which is especially relevant for the more interesting "Episodes", "Score" and "Duration_Minutes" columns.

```
# find columns with "Unknown"
names(which(sapply(dataset, function(x) any(str_detect(x, "^Unknown$")))))
```

```
##  [1] "Synonyms"     "Japanese"     "English"      "Synopsis"     "Type"
##  [6] "Start_Aired"  "End_Aired"    "Premiered"    "Broadcast"    "Producers"
## [11] "Licensors"    "Studios"      "Source"       "Genres"       "Themes"
## [16] "Demographics"
```

In addition to that, infos for several text-based columns like "Premiered" are sometimes incomplete. In these cases, the string "Unknown" is present instead.

Dates

The columns "Start Aired" and "Premiered" both contain date information.

"Start Aired" contains data in varying formats and different amount of detail. Observed formats are:

- Month Day, Year
- Month Year
- Month-ShortYear
- Year
- "Unknown"

Premiered contains the season and year the anime was released, or "Unknown". The year information seemed redundant in both columns, but a comparison showed some differences:

```
# Find mismatches between Premiered year and Start_Aired year
dataset %>% filter(str_ends(Premiered, "[0-9]{4}")) %>%
  filter(str_extract(Start_Aired, "[0-9]{4}$") != str_extract(Premiered, "[0-9]{4}$")) %>%
  select("Title", "Start_Aired", "Premiered")
```

```
## # A tibble: 67 x 3
##    Title                                                Start_Aired  Premiered
##    <chr>                                                <chr>        <chr>
##  1 "Shingeki no Kyojin: The Final Season"               Dec 7, 2020  Winter 2021
##  2 "Kimetsu no Yaiba: Yuukaku-hen"                      Dec 5, 2021  Winter 2022
##  3 "Major S2"                                           Dec 10, 2005 Winter 2006
##  4 "Pokemon Housoukyoku"                                Dec 3, 2002  Winter 2003
##  5 "Ayakashi"                                           Dec 12, 2007 Winter 2008
##  6 "Haiyoru! Nyaruani: Remember My Love(craft-sensei)" Dec 11, 2010 Winter 2011
##  7 "Pupipo!"                                            Dec 21, 2013 Winter 2014
##  8 "Boku no Imouto wa \"Osaka Okan\""                   Dec 22, 2012 Winter 2013
##  9 "Mazinger Z"                                         Dec 3, 1972  Winter 1973
## 10 "Weiß Survive R"                                     Dec 5, 2009  Winter 2010
## # ... with 57 more rows
```

1. The Premiered info contained a lot of "Unknown" values, whereas the Start_Aired column was mostly complete.

2. The year in both columns matched most of the time, but there were a few exceptions. Manual spot-checks of 5 of the affected anime revealed that the year in Start_Aired was correct in each case. In addition to that, all these cases were for a Winter season and listed the Premiered year as one year later than the Start_Aired year. This suggests that this is due to inconsistent definition of which year the Winter season belongs to - e.g. it runs from late 1999 to early 2000, is it the Winter season 1999 or 2000? Since a clear majority of rows attached it to the year in which it started (in this example it would be 1999), this was considered the correct definition for this project.
   This meant the year in Start_Aired is the only relevant one and the year in the Premiered column is consequently irrelevant.

Comma-separated values

The following columns all contain comma-separated values:

- Demographics
- Producers
- Licensors
- Studios
- Genres
- Themes
- (Synonyms)

The vast majority of entries in each column only contain one or a few entries, the biggest outlier was a Producer entry with 20 comma-separated values.
Synonyms were not checked in detail, since they were deemed irrelevant early on.

Data preparation

Dropping columns

```
# drop columns that seem irrelevant
dataset <- dataset %>% select(-c("Synonyms", "Japanese", "English"))

# drop ranked, popularity as it is merely a representation of order for score
# and members, respectively
dataset <- dataset %>% select(-c("Ranked", "Popularity"))

# drop synopsis as the present text vary a lot in terms of content, length and
# writing style, and that presumably is mostly due to the fact that the texts
# are written by lots of different writers, which is unrelated to the anime itself
dataset <- dataset %>% select(-"Synopsis")

# remove end aired as it's redundant with start_aired and Episodes
dataset <- dataset %>% select(-"End_Aired")

# drop attributes that are considered not available for the prediction,
# excluding the score value that shall be predicted (as it's need for the test set)
dataset <- dataset %>%
  select(-c("Scored_Users", "Members", "Favorites"))

# drop attributes that were deemed of too minor relevance after initial experimenting with models
dataset <- dataset %>% select(-c("Broadcast", "Producers", "Licensors", "Studios", "Rating"))
```

All columns that were deemed unneeded/unavailable in the exploratory data analysis (except “Score”, which is still needed for training) were dropped:

- English
- Japanese
- Synonyms
- Synopsis
- End_Aired
- Scored_Users
- Ranked
- Popularity
- Members
- Favorites

Later on, it became clear that training the models with that many predictors was not feasible, so the following columns were dropped as well:

- Producers
- Licensors
- Studio
- Broadcast
- Rating

Producers, Licensors and Studios each not only expanded the number of rows in the dataset when their comma-separated values were spread over multiple rows, they also each contained hundreds of unique, categorical values, making them seem like less promising candidates for predictors.

Dropping rows

```
# filter out anime that are not yet fully released
dataset <- dataset %>%
  filter(Status == "Finished Airing") %>%
  select(-c("Status"))
```

First off, all anime that did not yet finish airing were dropped as the goal is to predict ratings after an anime has finished. As the columns "Status" now had the value "Finished Airing" for every entry, it was dropped as well.

```
# filter out entries that do not have a score, episode count or duration
dataset <- dataset %>% filter(!is.na(Score) & (!is.na(Episodes)) &
                                (!is.na(Duration_Minutes)))
```

There was a low number of rows with NA values in the columns "Score", "Episodes" and/or "Duration_Minutes". These rows were dropped from the dataset.

The rows containing "Unknown" values were kept and were not changed, as actually the majority of rows have some missing ("Unknown") values.

Restructuring Date information

```
any_month_regex <- "Jan|Feb|Mar|Apr|May|Jun|Jul|Aug|Sep|Oct|Nov|Dec"

# restructure and unify date info
dataset <- dataset %>% mutate(
  Start_Aired_Year = case_when(
    Start_Aired == "Unknown" ~ "Unknown",
    str_ends(Start_Aired, "[0-9]{4}") ~ str_extract(Start_Aired, "[0-9]{4}$"),
    as.integer(str_extract(Start_Aired, "[0-9]{2}$")) > 22 ~
      paste0("19", str_extract(Start_Aired, "[0-9]{2}$")),
    as.integer(str_extract(Start_Aired, "[0-9]{2}$")) <= 22 ~
      paste0("20", str_extract(Start_Aired, "[0-9]{2}$"))
  ),
  Start_Aired_Month = ifelse(
    str_detect(Start_Aired, any_month_regex),
    str_extract(Start_Aired, any_month_regex),
    "Unknown"
  ),
  Start_Aired_Day = ifelse(str_detect(Start_Aired, "[0-9]{1,2},"),
                           str_extract(Start_Aired, "[0-9]{1,2}(?=,)"),
                           "Unknown")
) %>% select(-Start_Aired, -Start_Aired_Day)

dataset <- dataset %>% mutate(
  Start_Aired_Season = ifelse(Premiered == "Unknown",
                              Premiered,
                              str_extract(Premiered, "Spring|Summer|Fall|Winter"))
) %>% select(-Premiered)
```

```r
# drop a few rows that do not have a Start_Aired_Year
dataset <- dataset %>% filter(Start_Aired_Year != "Unknown")
```

To be able to better work with the date data, the available information was split into multiple columns:

- Start_Aired_Year: based on Start_Aired only - "Unknown" if Start_Aired was "Unknown"
- Start_Aired_Month: based on Start_Aired - "Unknown" if Start_Aired was "Unknown" or did not contain month info
- Start_Aired_Day: based on Start_Aired - "Unknown" if Start_Aired was "Unknown" or did not contain day info
- Start_Aired_Season: based on Premiered - "Unknown" if Premiered was "Unknown"

During this transformation, the inconsistent format of the "Start_Aired" data was also taken care of. The few rows that had no known year associated with them were removed from the dataset.
Also, the columns "Start_Aired" and "Premiered" were now no longer needed and thus dropped. While developing the models, it became clear that the information in Start_Aired_Day didn't prove to be valuable, so this column was ultimately dropped as well.

Expanding comma-separated values

```r
# expand comma-separated values
dataset <- dataset %>% mutate(
  Demographics = str_split(Demographics, ", "),
  Genres = str_split(Genres, ", "),
  Themes = str_split(Themes, ", ")
) %>% unnest(Demographics) %>%
  unnest(Genres) %>%
  unnest(Themes)
```

Initially all comma-separated values were spread out over multiple rows. This resulted in a dataset of about 100,000 rows. As later on, the columns "Producers" and "Licensors" turned out to be unused and were dropped, only the columns "Demographics", "Genres" and "Themes" were spread out now, resulting in about 40,000 rows.

Type conversions

```r
# remove whitespaces from strings
dataset <- dataset %>% mutate_at(
  c("Type", "Source", "Genres", "Themes", "Demographics", "Start_Aired_Month",
    "Start_Aired_Season"), str_replace_all, " ", "_")

# convert columns to Factor or Integer where applicable
dataset <- dataset %>% mutate_at(
  c("Type", "Source", "Genres", "Themes", "Demographics", "Start_Aired_Month",
    "Start_Aired_Season"), as.factor)
dataset <- dataset %>% mutate_at(
  c("Episodes", "Duration_Minutes", "Start_Aired_Year"), as.integer)
```

The resulting categorical data was converted into Factors, while all numerical data was converted into an integer wherever possible. As one of the used models turned out to have issues with spaces in factor names, all spaces were replaced beforehand.

This resulted in the following columns:

- ID (numeric): unique identifier of an anime - not used for training models
- Title (character string): title of the anime - not used for training models
- Type (factor): type of anime, e.g. "TV", "Movie" etc.
- Episodes (integer): number of episodes
- Start_Aired_Year (integer): The year when the anime first started airing
- Start_Aired_Month (factor): The month when the anime first started airing
- Start_Aired_Season (factor): The season when the anime first started airing
- Source (factor): source material the anime is based on
- Genres (factor): genre of the anime
- Themes (factor): theme of the anime
- Demographics (factor): target demographic of the anime
- Duration_Minutes (integer): Episode duration in minutes
- Score (numeric): The rating of the anime - should be predicted

Create datasets

```
# Create datasets
validation_set <- dataset %>% filter(as.integer(Start_Aired_Year) >= 2020)
data_for_training <- dataset %>% filter(as.integer(Start_Aired_Year) >= 2020)

set.seed(1, sample.kind="Rounding")
test_index <- createDataPartition(y = data_for_training$Score, times = 1,
                                  p = 0.1, list = FALSE)
training_set <- data_for_training[-test_index,]
test_set <- data_for_training[test_index,]
```

Now that the data preparation is finished, the validation, training and test datasets can be created.

As discussed in the introduction section, all anime from 2020 or newer form the validation dataset, whereas the rest is split 90/10 into training and test data.

Training models

```
naive <- mean(training_set$Score)
train_lm <- train(Score ~ Type + Episodes + Start_Aired_Year +
                    Start_Aired_Season + Source + Genres + Themes +
                    Demographics + Duration_Minutes,
                  method = "lm", data = training_set)
train_rf <- train(Score ~ Type + Episodes + Source + Demographics,
                  method = "ranger", data = training_set, num.trees = 50)
train_gl <- train(Score ~ Type + Episodes + Source + Demographics +
                    Duration_Minutes,
                  method = "gamLoess", data = training_set)
```

As a baseline, a naive approach is used that always just predicts the mean Score from the training set. Then, three models (standard linear regression, random forest and loess) are trained to see if they surpass the naive approach, and to determine which of them is best.

For the random forest, during the testing phase, it became clear that this model cannot handle instances of factors that were not present in the training dataset. This was hard to avoid with several factors that contained dozens of different values each, some of which were quite rare, especially considering that actual new data might contain new genres, themes etc. To get it to work at all, the predictors were changed to just numeric inputs as well as the season, which has only 5 possible values, all of which are common, and it is reasonable to assume that no new values will be added in the future either.

Also, the loess model was unable to run with as many predictors as the standard linear regression model, so the number of predictors was reduced, removing those that were expected to be of comparatively lower importance.

Results

```
calculate_RMSE <- function(true_ratings, predicted_ratings){
  sqrt(mean((true_ratings - predicted_ratings)^2))
}

names = c("naive", "linear regression", "random forest", "loess")
test_results = c(calculate_RMSE(test_set$Score, naive),
                 calculate_RMSE(test_set$Score, predict(train_lm, test_set)),
                 calculate_RMSE(test_set$Score, predict(train_rf, test_set)),
                 calculate_RMSE(test_set$Score, predict(train_gl, test_set))
)

validation_results = c(calculate_RMSE(validation_set$Score, naive),
      calculate_RMSE(validation_set$Score, predict(train_lm, validation_set)),
      calculate_RMSE(validation_set$Score, predict(train_rf, validation_set)),
      calculate_RMSE(validation_set$Score, predict(train_gl, validation_set))
)
```

```
names(test_results) <- names
test_results
```

```
##             naive linear regression     random forest             loess
##         0.7932278         0.6898125         0.6252831         0.7082614
```

```
names(validation_results) <- names
validation_results
```

```
##             naive linear regression     random forest             loess
##         0.8318322         0.6946566         0.6228289         0.7328884
```

The results show that all models surpassed the baseline approach, however the random forest provided the best results, both on the test and validation data.

The standard linear regression performs decently on both datasets. The loess model provides similar results on the test set, but for the validation set, the gap is considerably larger due to loess's weaker results.

However, even with these issues all three models still managed to improve upon the naive approach by a good margin.

Conclusion

Data preparation

The preparation of the data worked well and eliminated several issues with the original data, like NA values, inconsistent and incomplete date entries, and irrelevant information.
Given the problem with the random forest later on, it might be worthwhile to consider grouping rarely occuring instances of a factor into a generic combined instance like “Other”, which could then also be used to handle unknown instances in the validation set (or actual future data).

Dataset creation

The approach to create both a test and validation set seemed to work well and indicated weaknesses of the random forest model that were not expected purely based on the test data. An improvement to this setup would be to split the validation set off early on, as this could potentially simulate the addition of new Factor values like a new genre, which was impossible to occur in the validation set using the approach used in this project.

Training and results

While the issue of the loess model with the factors containing spaces could be solved properly, the approach to mitigating the issues with the lack of performance and errors when using too many predictors and the issues specific to the random forest were rather improvised and could probably be resolved better.

The results indicate the random forest model is best suited for this problem, which is somewhat surprising considering it works with the least amount of predictors. Possibly, with a solution to allow it to use other predictors even better results could be achieved.
All in all, considering that there still seems to be plenty of potential for optimization of the models, rather than the resulting RMSE comparison I would consider the lessons learned the actual most valuable results from this project.