D214: Capstone - Task 3

Scott Sullivan | 01054753 | Aug 9, 2023

A1. Executive Summary

Problem Statement

Our company's streaming platform is growing in popularity and with these additional users, we are seeing a strong demand to grow user engagement. Previously our data team had shown a correlation between additional minutes watched by users to a reduction in churn, and increase in profitability for the company.

Our goal is to determine if a movie recommendation engine for our customers, based on a regression model, can predict average viewer ratings of a film in order for our company to gain insights on which films to feature to the customers. The goal is that if we can accurately predict user ratings of a film, we will see a correlation to viewership of that film and more engagement and higher viewer satisfaction.

Hypothesis

The data analytics team has chosen as part of the methodology, to use a Random Forest Regression in order to verify rejection of our null hypothesis. Specifically, we have several attributes associated with each film in our database, such as the film's genre, revenue, and ratings, with which we will train our model.

Specifically, we have narrowed our null-hypothesis (H0) and alternate-hypothesis (H1) to the following:

- **Hypothesis: Null hypothesis (H0)** - The accuracy of a Random Forest Regression-based movie recommendation system does not significantly predict the average viewing rating of users on our streaming service.
- **Alternate Hypothesis (H1)** - The accuracy of a Random Forest Regression-based movie recommendation system significantly predicts the average viewing rating of users on our streaming service.

We will measure the accuracy of our model with the Mean Squared Error (MSE) so that we can statistically verify the performance of the model (as compared to a test set of actual values).

A2. Summary of Data-Analysis Process

Data Preliminary Examination and Cleaning

The data analytics team began the collection process by searching the data analytics website, kaggle.com, in order to find the appropriate data set.

The datasource was downloaded from kaggle and placed in a subfolder for our project in preparation. The data source (*https://www.kaggle.com/datasets/utsh0dey/25k-movie-dataset*) for the full movie dataset was saved to our local folder for analysis.

Once we found this data set, we conducted a preliminary examination of the data and found that it had the appropriate data needed for the analysis we were looking to conduct. However, it did need some cleaning as some of the data values were in the wrong columns or incorrectly labeled, along with some null values in the data set. We dropped any null value rows, and used Pandas in order to examine the data, format it to the proper data types, and drop any unneeded attributes.

Data Extraction and Preparation

We began by importing the required python frameworks into Jupyter Notebook, and then importing the raw data as a Pandas dataframe, followed by the cleaning mentioned above.

During our framework import, frameworks such as *pandas, numpy, tensorflow, keras*, and *nltk* are imported for each part of the cleaning and classification analysis.

We chose several frameworks for their strength in working with Random Tree Regression models, such as Sci-kit Learn. Additionally, we preferred having the ability to tweak many of the model's parameters within Sci-kit Learn for our specific use case. Unfortunately, we did notice that due to the limitations of the data team's computer, running the model training on such data sets can be taxing on the computer. However, this did not prevent us from generating a successful model and being able to conclusively make some recommendations. Future model training would benefit, however, from taking advantage of higher end GPUs to train models of significant depth.

At this point, we will now walk through each step of the data extraction and preparation process.

Drop Null Values

When looking at the data as part of the exploratory phase, the author detected several movies had zero user ratings. One reason was determined that this was because the film had not been released yet. Any zero values were dropped to avoid unreleased, non-reviewed films in the analysis.

Drop Any Rows with Wrong Data Type

During the expiratory phase, the author also found that some of the 'Run Time' attributes had either a 'not-released' instead of run time value, or had a revenue value instead of run

time. To fix these incorrect data types from affecting our data, we dropped those values. Additionally, many of the lists in Genres came in as strings and not lists. That will be fixed later, as will the mis-spellings in the column names from the original data source.

This was all accomplished using the built in Pandas functions (such as ***str.contains()*** and ***drop()*** for the cleaning).

Convert Run Time and User Ratings

Once this was accomplished, the author took the 'run time' attribute, which was a string that was in the format "X hours and Y minutes", and split the string to get just the hour and minute values, converted to integers to show the total running time in minutes.

Additionally, to reflect this in the data, the column name was renamed to clarify that the number is minutes. The author added the extra "if" statement to check in case the conversion had already been done and there was not column named 'Run Time' (since he had gone back a few times to rerun the code).

User ratings also were listed with a "K" instead of thousands digits. Logic was added to reformat the user ratings into numerical values. This was all accomplished using standard python and pandas framework calls.

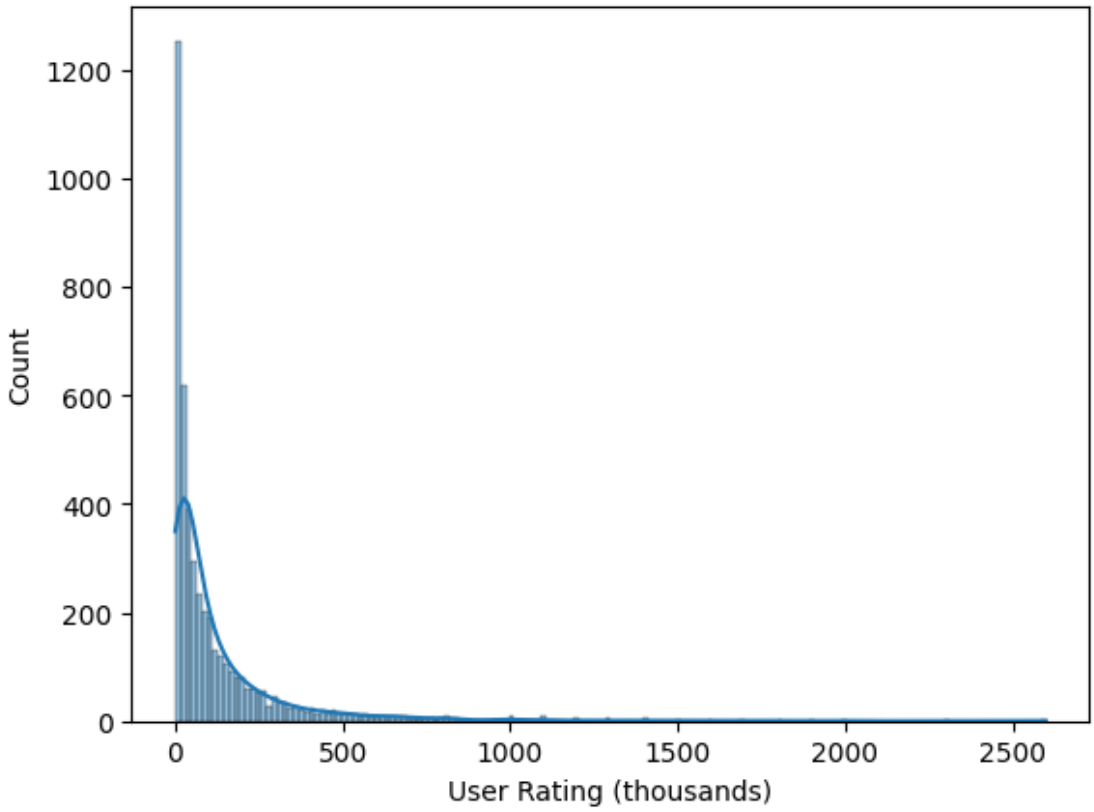
Adjust for Outliers in 'User Rating (thousands)'

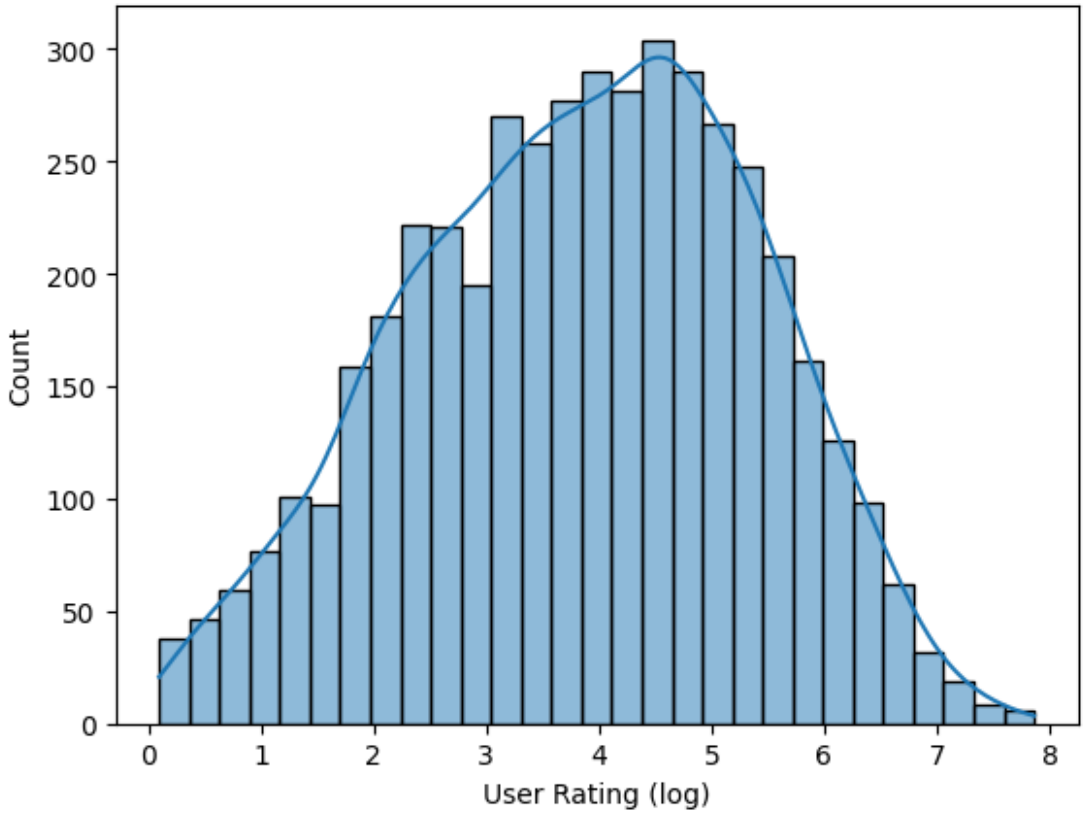
One of the key elements, the "User Rating (thousands)" attribute, appears to have significant outliers. The author concluded this was due to the results of summer 'blockbuster tentpole' films that grossly outperform many other films. We adjusted our values to account for this.

The author first graphed this attribute to verify the outliers with Seaborn Plot and saw a distict highly right skewed distribution, showing a very long tail of outliers. Examining the scale of the graph also confirms that we have a large number of small numbers and a very small number of high volume user ratings, consistent with the idea of a 'summer blockbuster' film having a giant number of user ratings compared to smaller budget films.

With this confirmed via a visual plot, the author then performed a log transformation in order to deal with the skewed data. This will help to normalize the data. Using a log normalization helped to compress outliers (Source: Janeway).

The following two graphs illustrate the before and after of the log transformations of the user ratings.


Count
1200
1000
800
600
400
200
0
0
500
1000
1500
2000
2500
User Rating (thousands)


Count
300
250
200
150
100
50
0
0
1
2
3
4
5
6
7
8
User Rating (log)

Convert Genres Using One Hot Encoding

The values in the "Genres" attribute column were discovered earlier, during the exploratory phase, to not actually being lists, but rather a single string. Therefore, the author had to split the strings to create lists, and then use those to base our one hot encoding of the values. This was accomplished using Sci-Kit Learn and the MultiLabelBinarizer function, in addition to dropping the original column after concatenating the One Hot Encoded data.

Below are the before and after results of our one hot encoding of the Genre attribute:

```
from sklearn.preprocessing import MultiLabelBinarizer
df_raw['Genres'] = df_raw['Genres'].apply(lambda x: [i.strip() for i in x.strip("[]").
df_raw.head(3)
```

	Title	Revenue	Rating	Genres	User Rating (thousands)	User Rating (log)
0	Top Gun: Maverick	170.0	8.6	[Action, Drama]	187.0	5.236442
2	Top Gun	15.0	6.9	[Action, Drama]	380.0	5.942799
3	Lightyear	71.0	5.2	[Animation, Action, Adventure]	32.0	3.496508

```
mlb = MultiLabelBinarizer()
one_hot_labels_genres = mlb.fit_transform(df_raw['Genres'])
df_encoded_genres = pd.DataFrame(one_hot_labels_genres, columns=mlb.classes_)

df_rawOneHot = pd.concat([df_raw, df_encoded_genres], axis=1)
df_rawOneHot.drop(['Genres'], axis=1, inplace=True)
```

```
df_rawOneHot.head(3)
```

	Title	Revenue	Rating	User Rating (thousands)	User Rating (log)	Action	Adventure	Animation	Biography
0	Top Gun: Maverick	170.0	8.6	187.0	5.236442	1.0	0.0	0.0	0.0
2	Top Gun	15.0	6.9	380.0	5.942799	1.0	1.0	1.0	0.0
3	Lightyear	71.0	5.2	32.0	3.496508	1.0	1.0	0.0	0.0

Analysis

The author decided to use a Random Forest Regressor model for several reasons. First, he has had good results using this model in a previous course as part of this program. But more importantly, the Random Forest Regressor has a large advantage over other methods of analysis with this specific data set because some of the data has such a wide distribution. This advantage is that a Random Forest is better on working through non-normalized data due to the way it returns average values of the individual trees (Source: Fernandez).

We just have to be careful to not overfit our data with a Random Forest Regressor (Source: Ellis).

We chose, as part of our original hypothesis, to use the User Rating as the target variable. We did this because viewers were much more likely to watch a film if it is rated higher. The author contemplated using running time, but after reflection, decided people are not going to the movies based on length, but on how the film is rated.

Preliminary Random Forest

Because of the large number of parameter adjustments that can be made, the author made use of a parameter grid in order to iterate through the parameters to find the best fit without overfitting. The author began with making educated guesses for the model to act as an initial point. He then set up the grid optimizer to get the final values in which to train the Random Forest Regressor. Below are the parameters we settled on for the model.

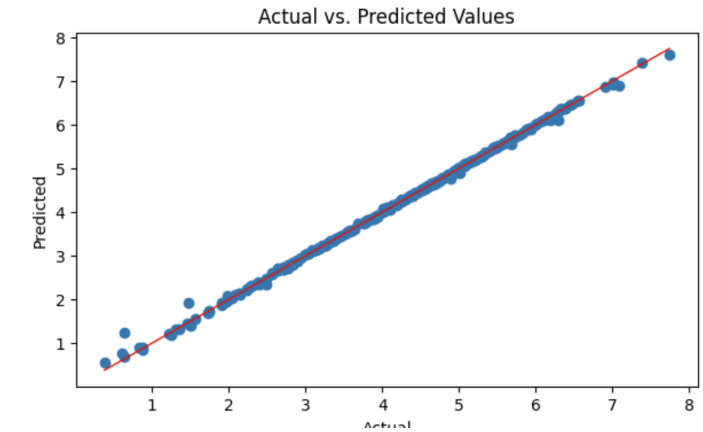
```
pipeline = Pipeline(steps=[('preprocessor', preprocessor),
                           ('regressor', RandomForestRegressor(n_estimators=200,
                                                               max_depth=10,
                                                               min_samples_split=2,
                                                               min_samples_leaf=1,
                                                               max_features=15,
                                                               bootstrap=True,
                                                               random_state=3))])
```

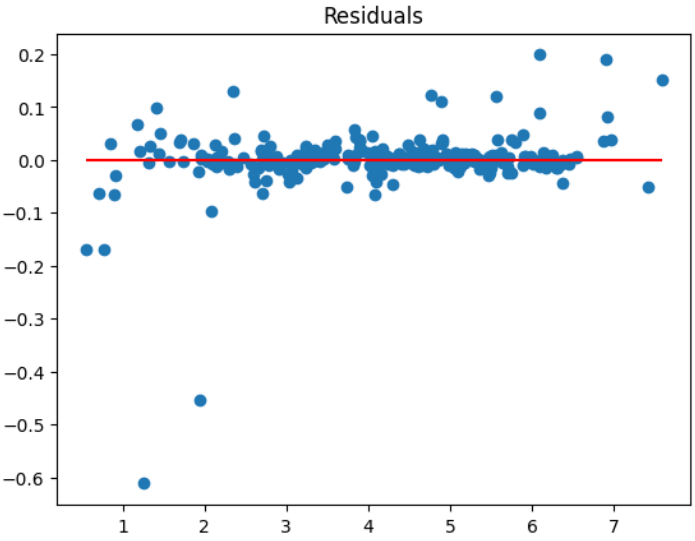
A3. Outline of Findings

To extract the accuracy of our model, we will compute the Mean Squared Error. We go into further detail on the meaning of the MSE in our summary in section E. But at a high level, the lower the value, the closer our predicted values match the actual values.

```
# Mean Squared Error Evaluation of Model
mse = mean_squared_error(y_test, y_pred)
print("Mean Squared Error: ", mse)
```

```
Mean Squared Error:  0.0034877868080804655
```





Having a MSE of this value is very good and the author confirmed the accuracy by running a secondary calculation to measure the accuracy of the model as a scatter plot, shown below.

Additionally, running a Residuals Plot allowed us to visualize the accuracy as well. Since it visualizes the error between the observed and predicted values, the plot of an accurate model will be scattered evenly on the horizontal axis. And this is exactly the plot that we see in the results to the left.

A4. Limitations of the Techniques and Tools Used

One limitation with this analysis conducted in our analysis was that of possible overfitting. As can be seen in the actual versus predicted graph, the predictions fall very close to the actual values. There exists a possibility that this model has been overfitted with the data, but additional calculations and analysis of the model will need to be conducted in order to verify if this is true, and if so, to what extent.

Also, running model creation tools like this on large data sets can be taxing on the computer and the author's computer is an older laptop. As such, he is unable to take advantage of any high end GPUs to train models of significant depth without locking up the computer (which he did in an earlier iteration).

A5. Summary of Proposed Actions

We can summarize our findings with the results of our final several calculations above. Specifically, the Mean Squared Error is **0.00349** for our data set's model. This is an indication of our model's performance compared to actual values (outcomes). We take the actual minus the predicted values, then square that result. We then sum those values and take the average. The closer the actual and predicted values, the lower the difference, and, thus, a lower MSE means a better performing model.

Since the result we see with our model is close to zero, we can conclude that the model performed relatively well at predicting the outcomes from our data set.

The author **proposes two actions for future study** of the data set. First, the data analysis should continue with newer data as it is added to the original data set and compared to see if it maintains the accuracy. This will ensure our data set continues to grow (hopefully becoming more accurate with additional data) and also stay fresh. The second area for future study that should be examined is blending in additional data sets that would complement the existing data. There are perhaps relationships that are not being extracted to their fullest extent and having additional demographic or larger economic factors that affect the results.

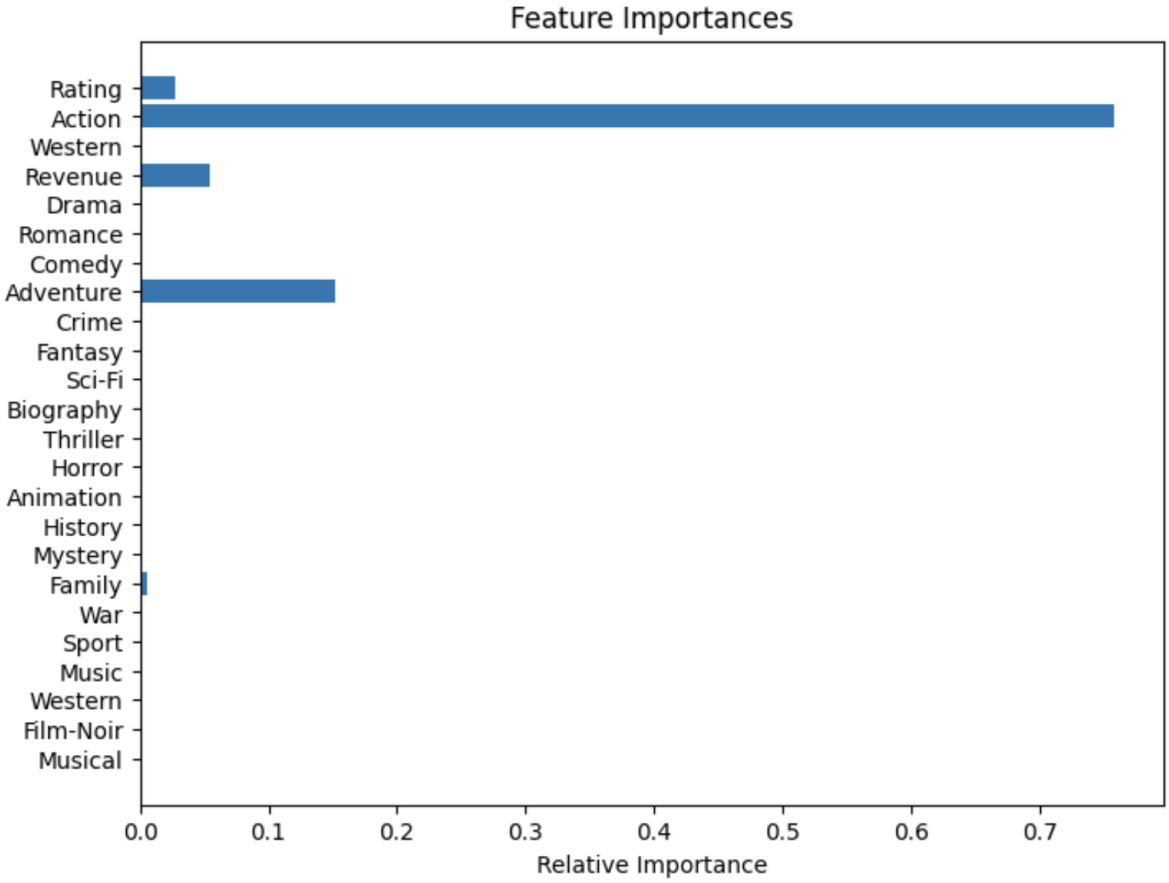
A6. Expected Benefits of the Study

The author found that by using a Random Forest Regression model, we were able to create a model that predicts user ratings with a MSE of **0.00349.** This is a very accurate model that will accurately predict user rating for films to be released based on specific metrics of Genre and Revenue, for example.

Our team was also able to determine which genres were more popular and more correlated with the user ratings to allow a faster way for us to get to our prediction. For example, we ran a feature_importances_() function on our data set and found that by far, the 'Action'

genre had more say in User Rating than any other single attribute. ‘Adventure’ and ‘Revenue’ were second and third, respectively.

Below is a full breakdown of our feature_importances_ graph:



As a result, we see that by rejecting our null hypothesis with a model that can accurately predict User Ratings based on the given parameters, that we can confidently use this model to assist the team that acquires film streaming licenses for future films to offer on our service. We recommend creating a model in the form of a product that is accessible via an intranet web page that is updated frequently as new films are released to create a curated list of films that would positively impact our user base and grow customer satisfaction.

Sources

- Fernandez, Javier. "How Data Normalization Affects Your Random Forest Algorithm." https://towardsdatascience.com/how-data-normalization-affects-your-random-forest-algorithm-fbc6753b4ddf
- Ellis, Christina. "Random Forest Overfitting." https://crunchingthedata.com/random-forest-overfitting/
- Janeway, Nicole. "Handle Outliers with Log-Based Normalization." https://www.nicolejaneway.com/python/outliers-with-log/