Machine Learning in Python - Project 1

Friday, March 18th

Anoushka Ghosh, Keith Tung, Jonathan Hoover, Rebekah Kiner

1. Introduction

NBC Universal has decided to produce the highly anticipated reunion episode of The Office. However, they are in need of insight of how to best produce the highest rated reunion episode possible. To assist NBC in this, we need to make an interpretable and predictive model that captures the underlying relationships between the features and the audience ratings for The Office. From this model, we will be able to advise NBC on features to include in the reunion episode that causes high ratings.

Data used to find these features were theoffice.csv and the show's text transcripts. The additional data sourced from the text transcripts for all the episodes in the show were downloaded from 'schruptepy.' The features from the transcripts included in our dataset were:

- `season` - Season number of the episode
- `episode` - Episode number within a season
- `episode_name` - Episode name
- `director` - Episode director(s)
- `writer` - Episode writer(s)
- `character` - Character who have spoken the corresponding lines
- `text` - Dialogues delivered by the corresponding characters
- `text_w_direction` - Dialogues delivered in a stage direction
- `season_episode` - Episode writer(s)

Before looking at models, some Data Cleaning and Data Exploration on the complete dataset was needed to learn more about our features and their relationship to the IMDb ratings. We also take into consideration the interaction between the main characters as features. To produce a model, we need to make sure the data is suitable and the features being used in the model appear to have some predictive power on the ratings.

Then, we fit different models for our data to check which one is the best predictive model. We examined the models of K-Nearest Neighbors Regression, Lasso Regression, Ridge Regression, and Random Forest Regression.

The model that worked best was Random Forest Regression with main character interactions, with R^2 score around 0.46-0.48 and Root Mean squared error around 0.356. The features that turn out to be a significant influence on the model, consist mostly of characters, some character interactions, some writers, directors, and the number of words and lines.

2. Exploratory Data Analysis and Feature Engineering

Data Cleaning

Looking at the director's names, we noticed that some of the directors' name occured twice because of mispellings in their names. We fixed this prior to any analysis. The mispelled directors corrected are:

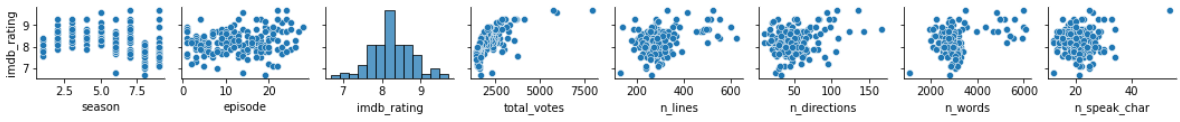
- Greg Daneils to Greg Daniels
- Charles McDougal to Charles McDougall
- Claire Scanlong to Claire Scanlon

The data has 185 entries/rows and 13 columns in total, without any missing values. Our features/columns for The Office data consists of

Col#	Column	Non-Null Count	Dtype	Description
0	season	186 non-null	int64	season
1	episode	186 non-null	int64	episode number
2	episode_name	186 non-null	object	episode names
3	director	186 non-null	object	directors
4	writer	186 non-null	object	writers
5	imdb_rating	186 non-null	float64	IMDB ratings of each episode
6	total_votes	186 non-null	int64	total votes of each episode
7	air_date	186 non-null	object	air date of each episode
8	n_lines	186 non-null	int64	number of lines in each episode
9	n_directions	186 non-null	int64	number of lines in a stage direction
10	n_words	186 non-null	int64	number of words in each episode
11	n_speak_char	186 non-null	int64	number of different characters with spoken lines in episode
12	main_chars	186 non-null	object	main characters appearing in episode

Out[10]:

```
<seaborn.axisgrid.PairGrid at 0x7f593275ac90>
```



We have created a pairplot looking only at the relationship between the variables and IMDB rating.

The plot suggests that IMDB rating follows a normal distribution. The columns season, total votes, episode variables show some potential as predictors in our initial exploration (we drop these later; see Feature Engineering for reasons).

We decided to show only the pairsplots comparing the variables to IMDB rating (instead of each variable comparison) because the remaining comparisons did not factor into our model creation. Note that there is a slight correlation between n_words and n_lines (which makes sense since number of lines are dependant on the number of words spoken), but dropping n_words did not improve the model performance, so we kept both n_words and n_lines.

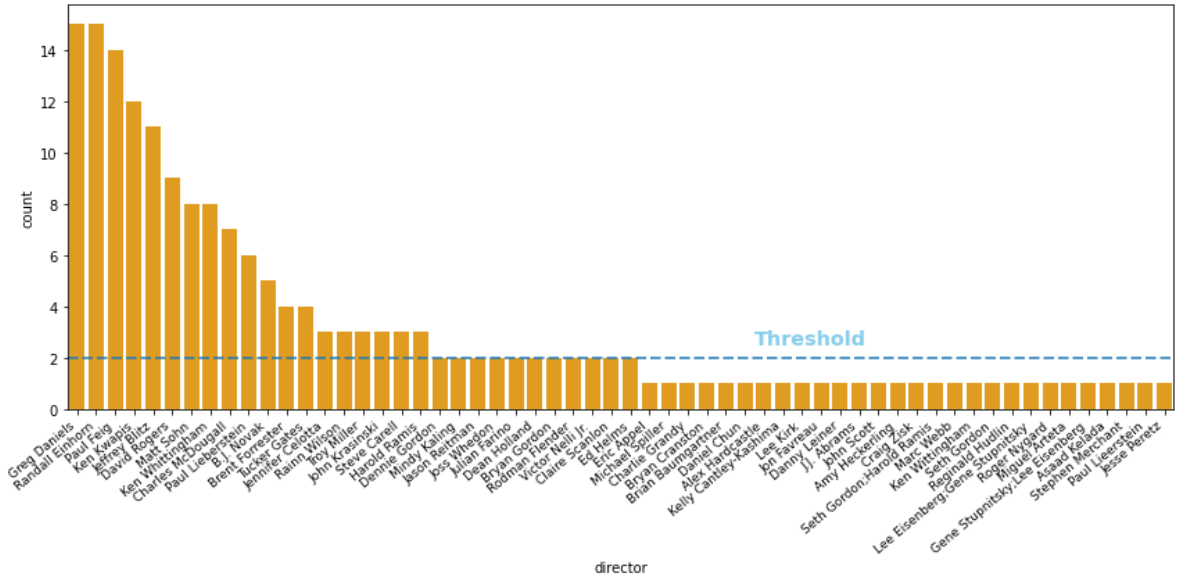
In the pairplot above IMDb ratings look normally distributed. To confirm we will run a Shapiro-Wilk test, which tests a null hypothesis that the data comes from a normal distribution against the alternative that it does not

come from a normal distribution.

```
Shapiro-Wilk Test statistic: 0.988 p-value: 0.12
```

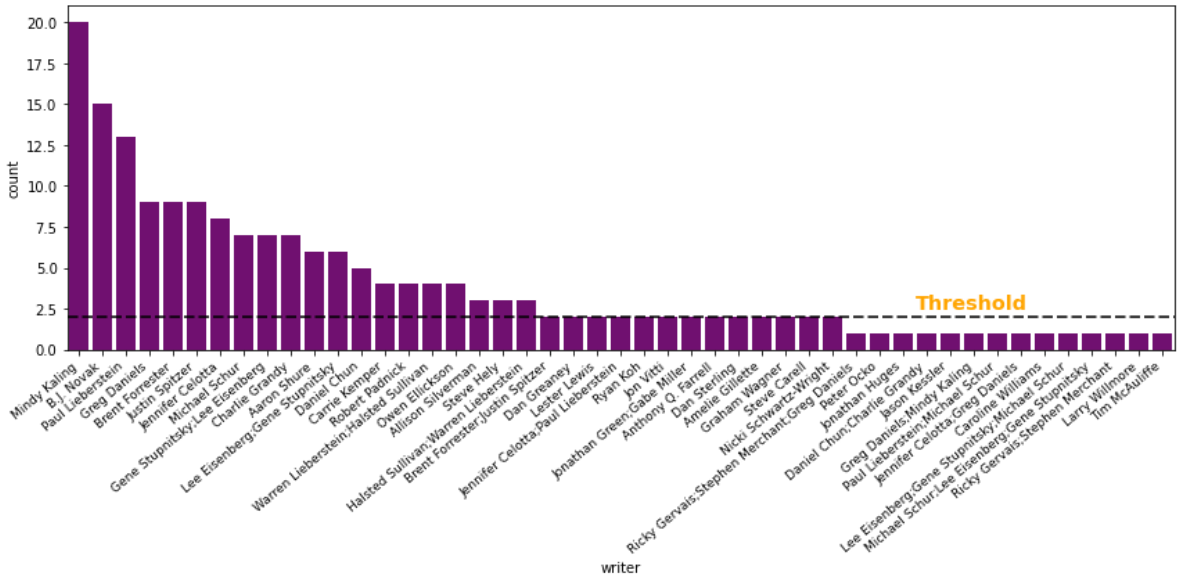
The Shapiro Wilk test suggests the IMDb ratings data is normal, since the p-value is not less than $\alpha = 0.05$. Thus, we fail to reject the null hypothesis, which says the data follows a normal distribution. This conclusion lets us consider traditional regression models for our prediction (however, we do not end up using the traditional regressions for different reasons; see the Model Fitting and Tuning Section for more details).

We assume that writers and directors have a strong influence on the ratings of an episode. Given this, we've looked at the distribution of both writers and directors along with the highest rated (and lowest) writers and directors throughout the show.



Out[13]:

```
<function matplotlib.pyplot.show>
```



The first barplot displays how many episodes a director directed. The main directors who have directed more than 6 episodes are:
-Ken Kwapis -Ken Whittingham -Greg Daniels

-Paul Feig -Charles McDougall - Randall Einhorn
-Jeffrey Blitz -David Rogers -Matt Sohn

The second barplot displays how many episodes each writer has written. The main writers who have written more than 10 episodes are
-B.J.Novak,
-Paul Lieberstein and
-Mindy Kaling.

Since there are a lot of directors and writers for this show, we want to reduce the number of features to consider. We assume those directors and writers with few appearances will have low impacts on IMDB rating predictions, so we will want to drop the low appearance writers and directors during feature selection. We've set a threshold of 2 episodes, below which we will drop. This threshold is shown on the plots. For more info, read the feature engineering section.

We are interested in which writers and directors might have the most predictive value for IMDB rating. As such, we look at which directors got the highest and lowest average IMDb ratings to explore if there is a relationship between writers/directors and IMDb rating.

Highest Rated Directors

	director	imdb_rating
5	Harold Ramis	8.866667
16	Steve Carell	8.733333
12	Paul Feig	8.685714
18	Tucker Gates	8.625000
9	Ken Kwapis	8.541667

Lowest Rated Directors

	director	imdb_rating
8	John Krasinski	7.633333
11	Matt Sohn	7.850000
14	Rainn Wilson	7.933333
1	Brent Forrester	7.950000
3	David Rogers	7.955556

Highest Rated Writers

	writer	imdb_rating
8	Greg Daniels	8.744444
7	Gene Stupnitsky;Lee Eisenberg	8.514286
13	Michael Schur	8.485714
16	Paul Lieberstein	8.400000
2	B.J. Novak	8.373333

Lowest Rated Writers

	writer	imdb_rating
1	Allison Silverman	7.433333
15	Owen Ellickson	7.450000
18	Steve Hely	7.600000
4	Carrie Kemper	7.825000
5	Charlie Grandy	7.900000

The writers and directors with highest average IMDb rating could indicate whom we should hire for the reuinion episode. Conversely, those with the lowest average scores will probably be avoided. The results will depend on the final model.

Categorical variables will need to be turned into dummy variables. The Categorical Variables are (5):
-episode_name -director
-main_chars -air_date
-writer

Some episodes have multiple writers and directors. Here we check how many episodes have more than one director or writer. We will need to handle this in the feature engineering section.

```
Director cells with more than 1 element: 3
Writer cells with more than 1 element: 34
```

When looking through the episode names, we identified some episodes that are multi-part episodes. However, some of the two-part episode names are stored as "Parts 1&2" while some are stored seperately as "Part 1" and "Part 2". These will have to be handled in feature engineering.

Out[20]:

	season	episode	episode_name	director
37	3	10	A Benihana Christmas (Parts 1&2)	Harold Ramis
50	3	24	The Job (Parts 1&2)	Ken Kwapis
79	5	17	Lecture Circuit (Part 2)	Ken Kwapis

Numerical columns will need to be scaled for a machine learning model. We found the following numerical columns in the dataset:
-season -episode -imdb_rating
-total_votes -n_lines -n_directions
-n_speak_char -n_words

We are interested in the effect of character lines per episode on IMDb rating (as well as character presence in episode), but this data is not included in the Kaggle dataset provided. Fortunately the schrutepy package contains the transcripts for each episode, so we will mine the transcripts for the total number of characters per line to do some initial exploratory data analysis.

Note that technically this data mining is considered feature engineering, so it could go in the following section; however, to do any exploratory data analysis we need to include it here.

The transcripts contain all speaking characters that appear in the show. However, some characters appear very few times throughout the series. Given that the total number of episodes is 186, we do not need the data from each individual character. This would likely lead to overfitting. As such, we needed to define a threshold for characters to include. We will call these characters the main characters.

To define the main characters we 1) calculate the total number of lines spoken by each character throughout the whole series, 2) Find the percentage of the total lines spoken throughout the series for each character, and 3) set a threshold beyond which a character is considered a main character. In this case, we have set the threshold to 0.005 (0.5% of total lines spoken).

The main characters we then get are :

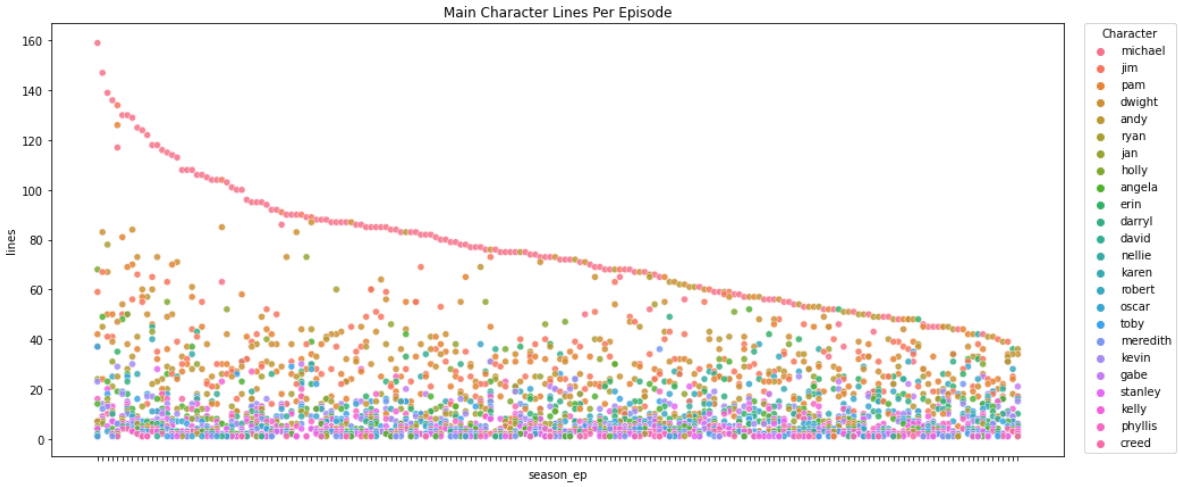
Character names	Character Names
Micheal	Jim
Pam	Dwight
Andy	Ryan
Jan	Holly

Character names	Character Names
Angela	Erin
Darryl	David
Nellie	Karen
Robert	Oscar
Toby	Meredith
Kevin	Gabe
Stanley	Kelly
Phyllis	Creed

We would imagine that character lines per episode might play a role in IMDb rating. We have plotted this below.

Out[23]:

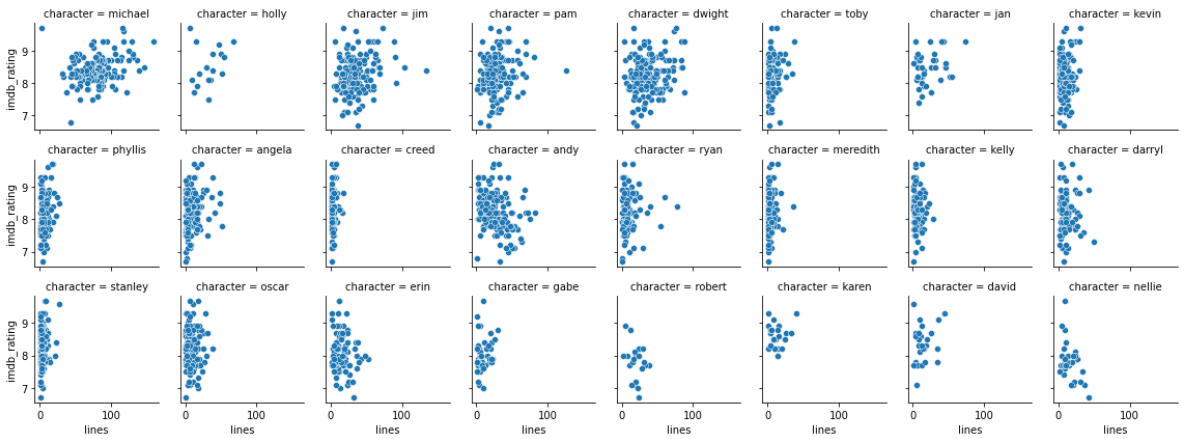
```
Text(0.5, 1.0, 'Main Character Lines Per Episode')
```



It is clear that certain characters dominate each episode (Particularly michael, jim, and dwight), but the patterns change over time. Interestingly, the total number of lines per episode declines over time, which, according to the pair plots above, might be correlated with IMDb rating.

We would, however, like to see if there is any correlation between total lines per character and IMDb rating. To assess this, we have merged line data with the IMDb data and plotted lines vs IMDb rating (grouped by main character).

```
<Figure size 1080x504 with 0 Axes>
```



While not all of the main character's lines appear to have an association with rating, it is clear that some do. In particular, the lines for Michael, Holly, Jim, Pam, and Dwight appear to show some correlation with IMDb rating, so these will likely be good predictors in an ML model.

Note that we have also calculated each characters' percentage of lines per episode and presence per episode (at least one line) for downstream machine learning. These features might be correlated with lines, but they do tell a slightly different story, so they have been included. The plots are not shown here.

Feature Engineering

A critical component to any machine learning analysis is Feature Engineering. Features define the model, so any transformations, additions, or reductions we perform on the feature space will affect the final model performance.

Given the exploratory data analysis, it is clear that we need to transform the writer and director columns into a categorical variable type that a machine learning model can handle. As noted before, some writers and directors have the names paired together with ";". We could either separate them or treat each unique combination as it's own category. We decided to treat each as its own category and create dummy variables that identify the combination of directors or writers present in an episode.

We will use one-hot encoding where each unique level of the categorical variable (minus one to ensure rank is sufficient) is given it's own binary column defining if that level is present of not. We use the pandas function get_dummies to do so.

Note that we defined the main characters as dummy variables above with the transcript data.

As we found in the exploratory data section, there are many writers and directors that appear only once or twice throughout the entire show. These columns do not add much information and only make the data more complex, so we combine these writers and directors into a new column called "low_appearance_writers" (or directors) and drop the original columns, as displayed below. This column is a dummy variable that defines when at least one of these low appearance writers or directors is present in the data.

As mentioned before the threshold for a "low_appearance" writer or director is one that appears in 1 or 2 episodes.

Out[27]:

	low_appearance_writer	low_appearance_director
0	1	0
1	0	0
2	0	0
3	0	1
4	0	0

As mentioned before, there are some two-part episodes. We created a column called "multi_part" to label these episodes as multi_part episodes with the expectation that they might yield different IMDb ratings since they are a unique episode type.

However, we also noted before that some of the two-part episode names are stored as "Parts 1&2" while some are stored seperately as "Part 1" and "Part 2". This could potentially cause bad IMDb predictions. Since separating is not an option, we decided to drop the observations for the two-part episodes labeled with "Parts 1&2". There are only two observations like this, so dropping should not have much impact.

In addition to the transformations and feature additions we performed above, there are some features we must remove. There are three categories for removal: 1) The data is not available before an episode is aired (and therefore cannot be used to predict). 2) The data will lack predictive power, and 3) The data is reflected in other columns.

Total votes falls into the first category. Total votes is the total number of IMDb votes for the episode, which will not be available before airing. Thus total votes is removed.

Season, season_ep, episode, episode_name, and air_date fall into the second category. Season, episode, and air_date will be known for new episodes, but the new episodes will have values for these features not contained within the training data, so they won't have predictive value. Thus, we drop these columns.

Episode name alone is not likely to have any predictive value. It might be correlated with episode content, which would have predictive value, but any prediction from it would not be interpretable, so we drop this feature as well.

Lastly, main_characters (the main characters provided in the original data) falls into the third category. We created variables for main character lines, line percentage, and presence during the feature addition phase, so the original main characters feature is redundant and thus dropped.

Next, we want to consider some interactions of our features. We cannot consider every interaction for all our features since it would yield over 7000 features and our dataset only has 184 observations (after removing 2 multi-part episodes). Thus, we only consider interactions between character lines since viewers enjoy some character combinations more than others so these interactions have the potential for predictive power. The model should detect which of the interactions are most important for IMDB ratings.

Standardisation of the data

The last feature engineering step is to standardize the numeric columns. Standardization is a critical step when the support of the individual features is different (https://medium.com/@urvashilluniyawhy-data-normalization-is-necessary-for-machine-learning-models-681b65a05029 (https://medium.com/@urvashilluniyawhy-data-normalization-is-necessary-for-machine-learning-models-681b65a05029)). This is important for regression models since the model requires differentiation and gradient calculation. This insures the model is optimized and coefficients converge since the features are all in the same range.

The standardization is applied only to the predictor variables (not response). However, the standardization is purely based on the training set mean divided by the standard deviation to ensure we do not overfit the test data.

Both a training set and a test set standardized by the training set values are created. In these datasets we split the response variable from the predictors (X from y) and then split both X and y into training and test sets with a 80/20 split (training/test).

Feature Reduction Side Note:

We attempted to reduce the feature set by removing highly correlated features; however, all resulting models had worse evaluation metrics (root mean-squared error, R^2 for regressors and F score, precision, and recall for classifiers). As such that analysis is not included.

3. Model Fitting and Tuning

Chosen Model: Random Forest

We would like a model that is highly interpretable and can inform the choices for a reunion episode of the office. Our response variable, IMDb rating, is continuous and normally distributed, so a regression approach (instead of classification) is the most viable option.

Our dataset contains more features than observations with 184 observations and 394 features (after dummy variables and interactions are created and before splitting into training and test sets). The training set (80% of data) has 147 observations and 394 features. Traditional linear regressions ML models are highly interpretable; however, they struggle with datasets containing more features than observations, so we need a regression approach that can handle this data structure and is still interpretable.

Random Forest regressors are a non-parametric, ensemble decision-tree method for regression that work well with datasets containing more features than observations [1, https://www.tandfonline.com/doi/pdf/10.1198/tast.2009.08199?needAccess=true] (https://www.tandfonline.com/doi/pdf/10.1198/tast.2009.08199?needAccess=true%5D). Additionally, this model can easily find the features that define the splits in the forest, which lends itself to an interpretable model.

After analyzing many different models, we have decided on the Random Forest model as it provided the best results and is interpretable. These results will be discussed in following sections.

Random Forest Background

Before getting into the details of our analysis, it is important to understand what random forests are and how they yield regression results.

Random Forests are a collection of trees built such that each tree contains a bootstrapped subsample of the data. Each tree contains only a subset of the total features, which reduces correlation in the trees. By reducing the total correlation, we reduce the effects of sampling error that would be present in a single tree.

Within each tree, the algorithm searches through the features to identify which is "most important" for splitting the data at a given node. In the scikit-learn package, the feature importance is defined by the gini importance, which calculates the mean decrease in impurity for each feature and selects the feature that will decrease the impurity the most at a given node (See [https://medium.com/the-artificial-impostor/feature-importance-measures-for-tree-models-part-i-47f187c1a2c3] (https://medium.com/the-artificial-impostor/feature-importance-measures-for-tree-models-part-i-47f187c1a2c3%5D) for more information). **So the higher the gini importance, the better the split**. This gini importance index will be important for identifying the features most likely to give a higher IMDb rating in a reunion episode.

For a random forest, the prediction is made by dropping the new observation into each tree of the forest, finding the predicted value for each tree, and averaging the values of each tree in the forest.

It is our hope that the Random Forest algorithm's ability to perform well on high feature, low observation data (while yielding feature information data), will help us identify features that are likely to yield a well-rated reunion episode.

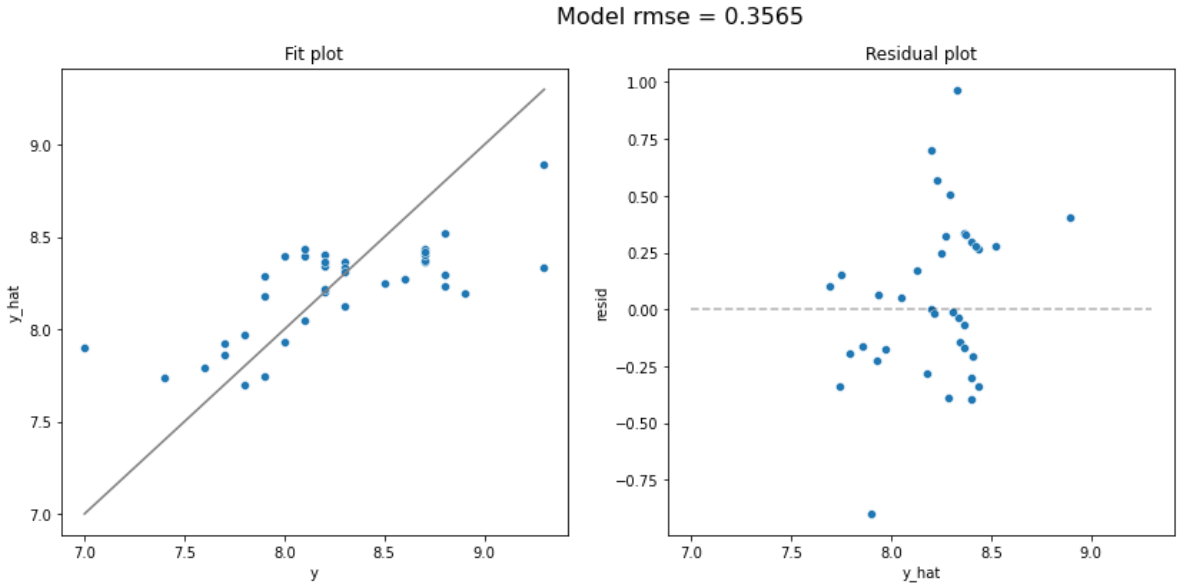
Model Tuning

We performed a cross-validated grid search (5-fold) on the following parameters in sklearn's RandomForestRegressor algorithm: n_estimator, min_impurity_decrease, min_samples_leaf, and max_depth to find the optimal parameters for our model. The optimal random forest based on the grid search is a random forest with 280 trees, a max of 7 nodes per tree, a minimum of 1 sample per terminal node, and a minimal impurity decrease of 0 (no minimum).

The true response vs predicted values and residuals plots from the test set are shown below.

Out[37]:

```
RandomForestRegressor(max_depth=7, max_features='sqrt', n_estimators=280,
                      random_state=22)
```



Out[38]:

```
0.35649703404486344


Training Set Results:
R^2 Train
0.8552456786412643
MSE Train
0.04327570743339031
RMSE Train
0.20802814096508748
--------------------------------------------------------------------------------
Test Set Results:
R^2 Test
0.479082649095413
MSE Test
0.12709013528278454
RMSE Test
0.35649703404486344
```

The results above indicate that the RF regressor explains 85.5% of the variation (R^2 value) in the training data and 47.9% of the variation in the test data. The root mean-squared errors are 0.21 and 0.36 in the training and test set, respectively. The model fits the training set better than the test data; however, the difference between

the two is not high enough to be concerned about overfitting.

The low R^2 value for the test set actually reflects the nature of the data. IMDb ratings are inherently subjective. They are influenced by the opinions of individuals, so there is intrinsically a high level of variability. As such, we can not expect a simple model to fully capture the variability in a noisy dataset.

While the test data set R^2 value of 0.479 means the model is not the best predictor, it still is a good model. The residuals show the error is randomly distributed about 0, so there is no association between the predicted IMDb rating and the error. Additionally, the rank of the true high values appears to be correlated with the rank of the predicted y-values (i.e. the highest true values are the highest predicted values even though the prediction is not 100% accurate). Therefore, while the model might not predict well, it is still a good explanatory model to understand the variables that most influence the rating. This will be explored next.

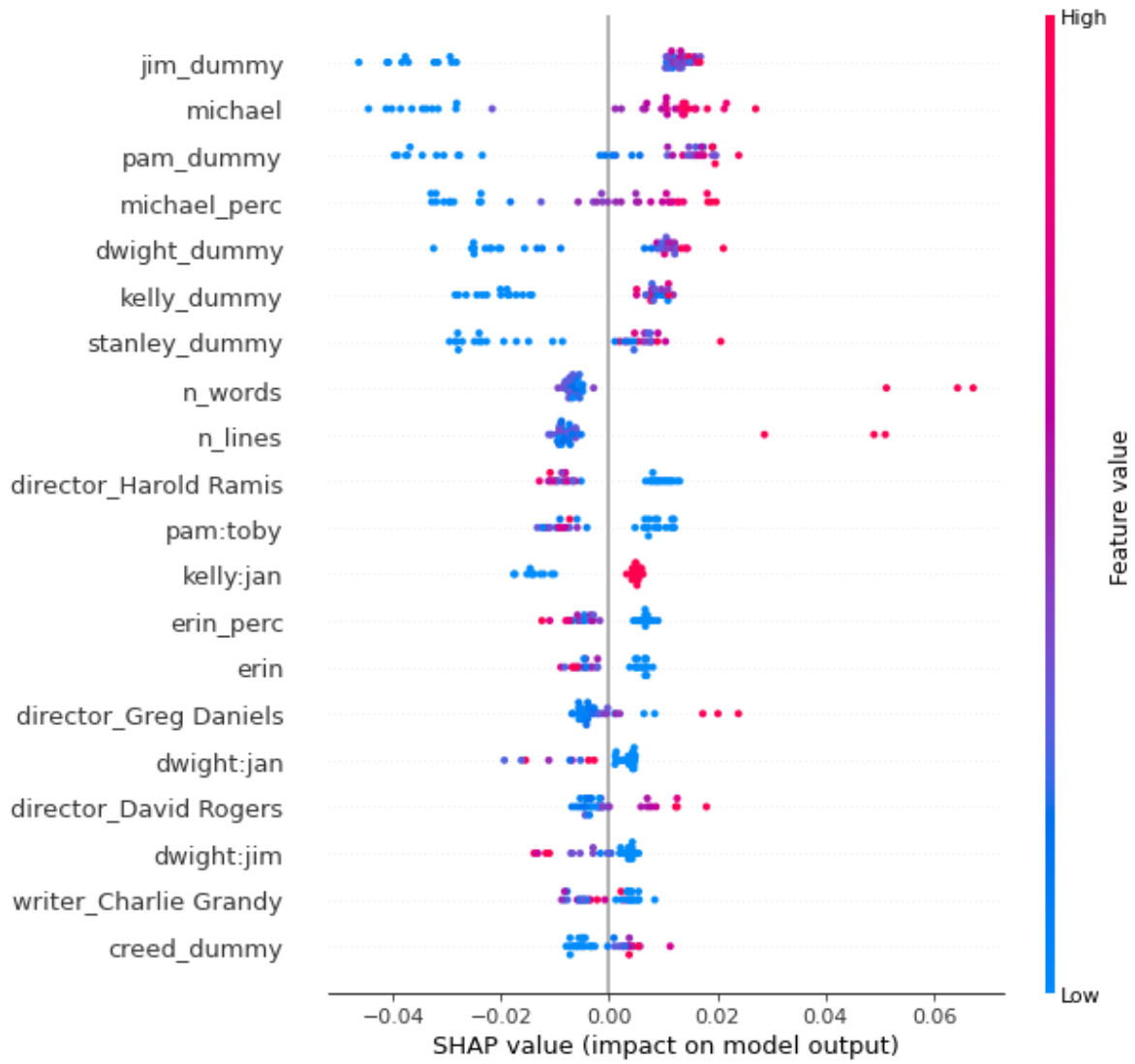
Feature Importance

To identify the important features could look at the gini importance (explained above); however, the gini importance only gives the absolute importance of a feature not the direction of effect. To get direction of effect, we will look at SHAP values. SHAP stands for SHapley Additive ExPlanations, coming from game theory. The SHAP values indicate the difference between the predictive model and the null model. Higher SHAP values are associated with a higher predictive power and lower SHAP values are associated with lower. For more details please see the following article https://towardsdatascience.com/explain-your-model-with-the-shap-values-bc36aac4de3d (https://towardsdatascience.com/explain-your-model-with-the-shap-values-bc36aac4de3d)

We use SHAP values to identify the direction of effect for a given feature by plotting the SHAP values for each feature and coloring by the value of the feature (plots shown below).

In the plot below, the x-axis stands for SHAP value. The y-axis has all the features. Each point on the chart indicates a SHAP value for the respective feature.

Negative SHAP values are associated with lower IMDb ratings and positive SHAP values are associated with higher IMDb ratings. Red indicates high values for the feature and blue indicates a lower value for the feature, so if a feature has all red points with positive SHAP values, that feature is positively associated with IMDb ratings. Conversely if all red points are associated with negative SHAP values, the feature is negatively associated with IMDb ratings.



The plot above, shows the top 20 features (according to gini importance), our model predicts the following features will have a positive impact on IMDb ratings:

- The presence of Jim, Michael, Pam, Dwight, Kelly, Stanley, and Creed.

- The character interactions of Kelly & Jan
- A high number of words and lines (including stage direction).
- The directors Greg Daniels and David Rogers

Our model predicts the following features will have a notable negative impact on IMDb ratings:

- The character interactions of Pam & Toby, Dwight & Jan, and Dwight & Jim
- The director Harold Ramis.
- The writer Charlie Grandy.

Note that these are just the top 20 features. For more nuanced info, we could look at more features, but these 20 are fairly informative.

What other models we tried

K-Nearest Neighbors

- A brief literature search on algorithms used to predict movie ratings showed KNN ML algorithms used occassionally, so we tried both a KNN regression and classification. Both regression and classification results were poor for all attempts at the algorithm (Regression: R^2 < 0.25, RMSE > 0.42; Classification: Avg Precision < 0.3, Avg Recall < 0.3 F-metric < 0.2), so we decided against this model.

Ridge Regression

- Considering the response variable, IMDb rating, is normally distributed and continuous nature, some form of regression makes sense. Ridge regression is suitable when a dataset contains a higher number of predictor variables, thus tested this model on the data. Ridge regression with an alpha of ~40.703(found using GridSearchCV) produced poor R^2 (<0.36) and RMSE results (>0.431) for all attempts, so we discarded it.

LASSO Regression

- Given the high number of features and low number of observations, we decided to try a LASSO regression in an attempt at feature reduction (Note that if a tuned value of alpha is 0, this yields a traditional regression, so we did not attempt a traditional regression). The LASSO regression results (tuned for an alpha value of 0.05) yielded poor R^2 (< 0.28) and RMSE (> 0.41) for all attempts, so we decided against the algorithm.

4. Discussion & Conclusions

While our Random Forest model is not the best predictive model (R^2 = 0.479), it clearly identifies key features that influence the IMDb score. From the SHAP plots, it appears that character presence and lines seem to dominate the effect, but also some characters and writers.

For the characters, Jim, Michael, Pam, Dwight, Kelly, and Stanley seem to be the most important characters in the show. Their presence has the most positive influence the IMDb score, so any reunion episode should these characters. Also, Michael should have a higher number of lines while the others just need to be present.

Additionally, some character interactions appear to be important. The data suggests that a combination of high Kelly and high Jan lines in an episode (not necessarily dialogue with each other) are associated with better IMDb ratings. Jan is not a common character, so this suggests that including Jan (with a high number of lines)

might make a better reunion episode. Interestingly, episodes with high Dwight and Jim lines seem to have lower IMDb ratings. Given their individual importance this suggests that a reunion episode should include them but not give them both a high number of lines.

Lastly, some writers and directors appear to influence the IMDb ratings. Directors Greg Daniels and David Rogers are associated with better ratings while Harold Ramis is associated with lower ratings. The writer Carrie Kemper also appears to be associated with better ratings. These results suggest that NBC producers may wish to hire Greg Daniels or David Rogers as directors and Carrie Kemper as a writer. Interestingly, the exploratory data analysis showed that Harold Ramis had the highest average IMDb rating for his episodes while the model suggests he has a negative impact on IMDb rating. This could either indicate that average score is not a good indicator for IMDb rating or that the model is predicting incorrectly (we do know the model is not perfect).

Given the predictive quality of our model is low, these interpretations may be innaccurate (see Harold Ramis). This model is a great tool to identify potential ingredients for a high quality reunion episode, but it does not capture all of the variability in the data.

An episode's performance is largely driven by the story, character interactions, non-verbal cues, etc., which cannot be captured in the data we have. We may be able to improve the model with a more complex text analysis of the transcript, like sentiment and tone analysis, but it requires complex NLP analysis not included here. Additionally, viewer's opinions are inherently varied, so an accurate prediction of rating is not particularly expected.

One last critical note is that the office aired from 2005 to 2013. Societal culture has changed dramatically since then, so any predictions from these models may not accurately reflect the reception in current times. Additionally, many of the current fans have watched most of the show on streaming services instead of TV, so the reunion episode would likely be streamed instead of aired. Thus, a better prediction metric for a reunion episode's success would be views per episode on the streaming platform rather than IMDb rating.

In conclusion, the best chance at a successful reunion episode is to bring the writers, directors, and characters identified above together to build a story that hopefully resonates with viewers, but without data that more accurately reflects the storyboard and current society's viewing norms, we will not have the best prediction.