Group 2 Final Project: Predicting Movies' IMDb Score

Adam Kritz, Sahara Ensley, Josh Ting
December 6th, 2021
George Washington University
Introduction to Data Mining - DATS 6103
https://github.com/Saharae/Final-Project-Group2

Introduction

The Internet Movie Database (IMDb) is a popular website for cataloging movies

Over 600,000 movies recorded on IMDb as of September 2021

Users and critics can vote on movies and leave reviews

Research Question

Movie industry wants to invest in movies that they know will be successful

Is there a way to predict if a movie will be enjoyed before it is even released?

Can we build a model to predict the IMDb score of a movie based on its characteristics?

Our research will shed light on whether a model can predict if a movie will be successful solely based on information about the movie

Data Description

IMDb Movies Extensive Dataset on Kaggle

Four .csv files initially, narrowed down to Movies and Ratings

Combined the features from Movies and the weighted average vote from Ratings

Target Variable

Several target variables we were interested in initially: Female average vote, Male average vote, average vote, and weighted average vote

Decided on weighted average vote

Weighted average vote appears on IMDb, aka IMDb Rating

Weighted average vote can prevent drastic changes

Weighted average vote ranges from 1-10, to one decimal place

Variables That Did Not Make The Cut

- Average vote and metascore
  - Both based on the same factors as weighted average vote, could not not be used as predictors
- # of reviews from critics, # of reviews from users, total votes
  - Were not actually features of movies
  - Confusion on definitions
- Language
  - High correlation with country

Variables We Chose

Four Numerical Variables

- Duration
- Budget
- USA Gross Income
- Worldwide Gross Income

Eight Categorical Variables

- Date Published
- Actors
- Writers
- Directors
- Production Company
- Title
- IMDb Description
- Country
- Genre

Our Modeling Plans

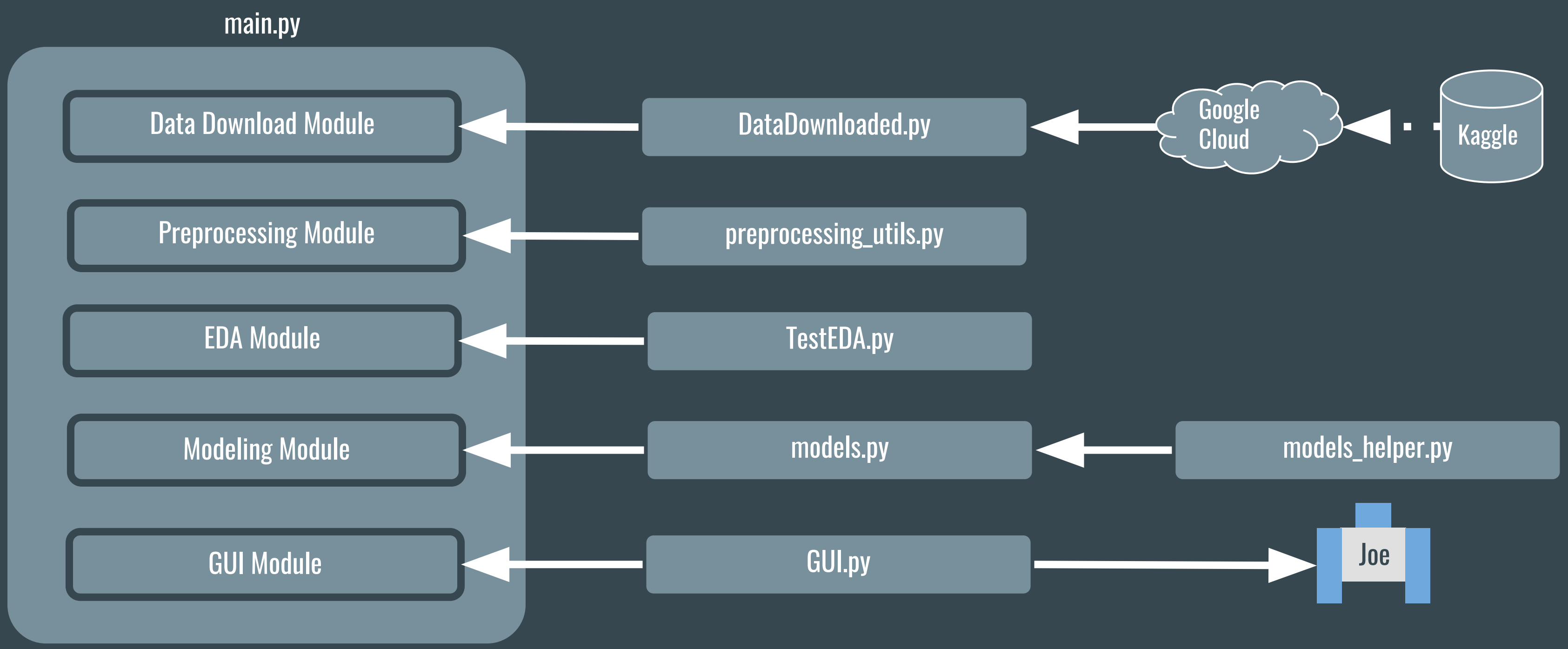
Create a model that could most accurately predict weighted average vote

Took a set of models from SKlearn to use for modelling

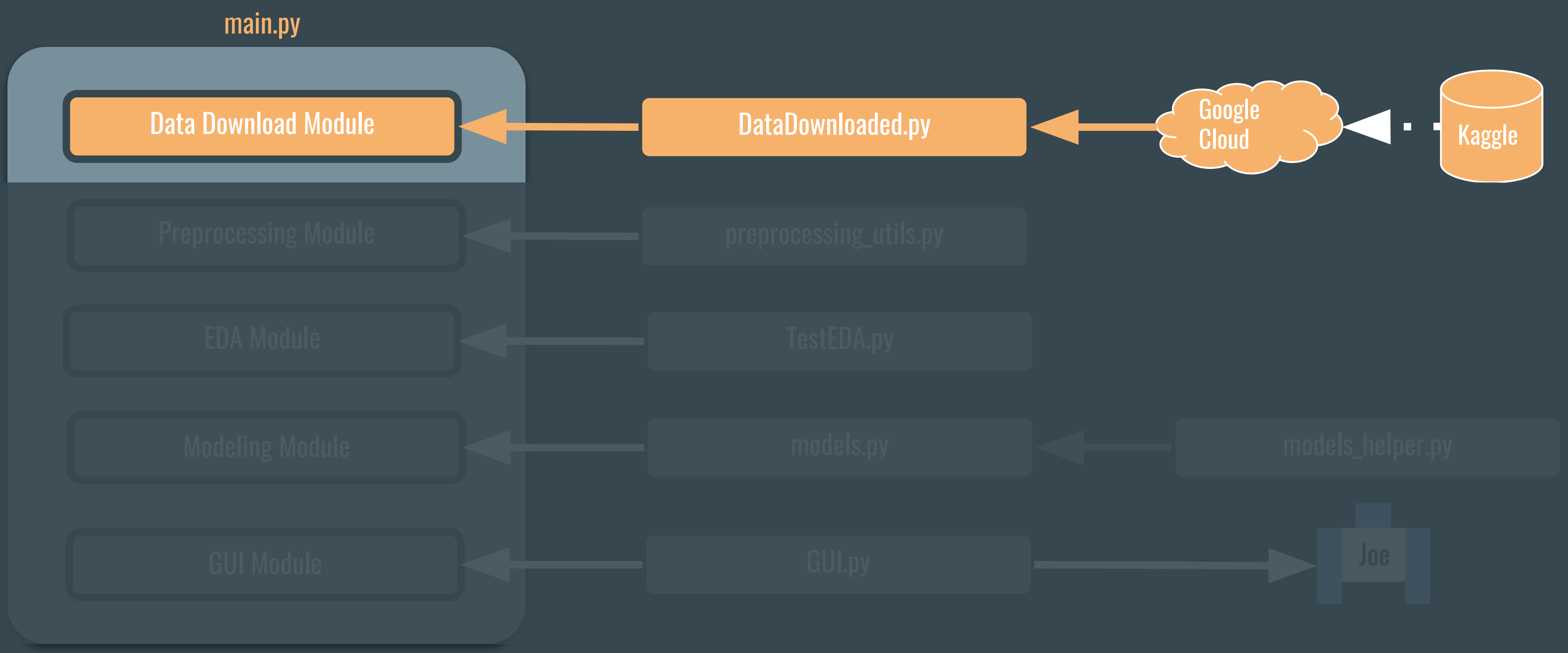
- Linear Regression
- Random Forest
- Gradient Boosting
- Adaptive Boosting
- K-Nearest Neighbors

Planned to improve upon the best performers to create the best model

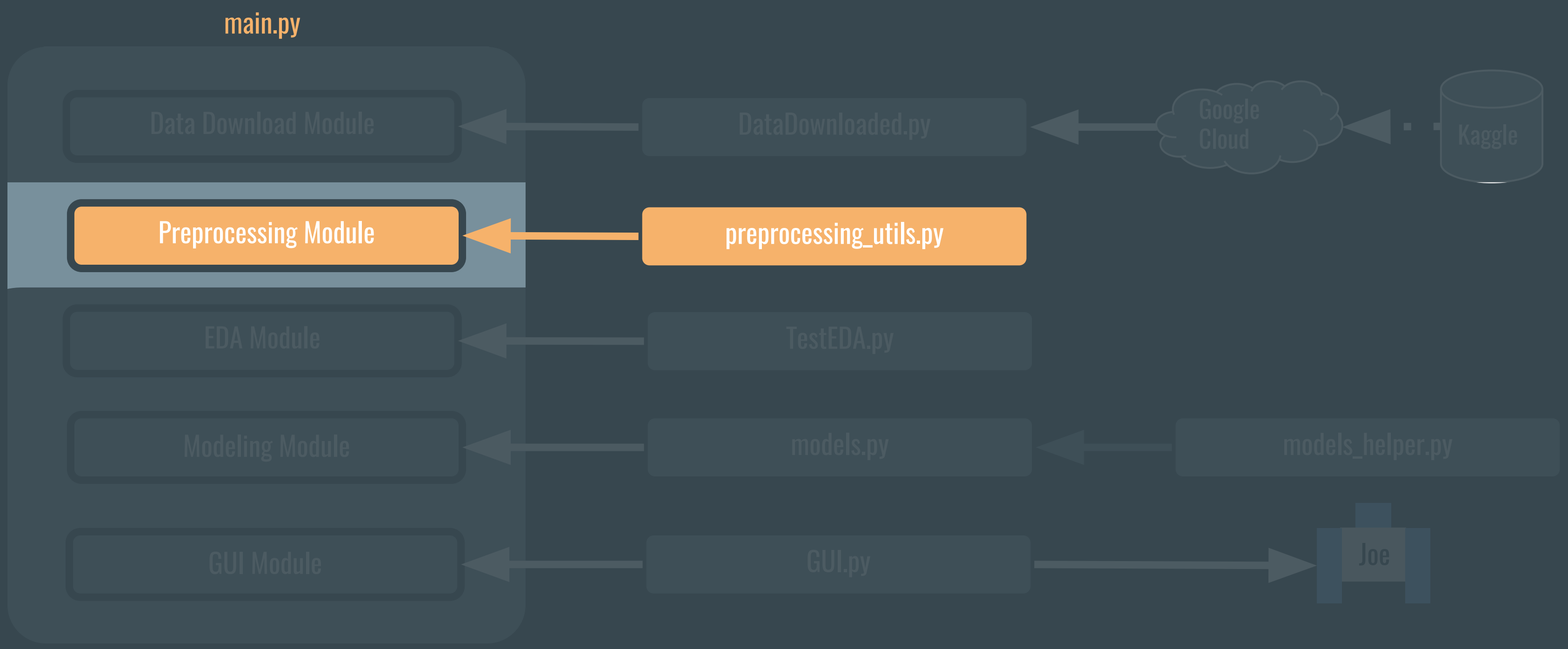
Code Architecture



Data Download



Preprocessing: Code Architecture



Preprocessing: Initial steps

Movies

Date Published
Budget
Worldwide Income
Domestic (USA) Income
Genre
Country
Director
Writer
Actors
Production Company
Title
Description
Duration

Ratings

Weighted Average Vote

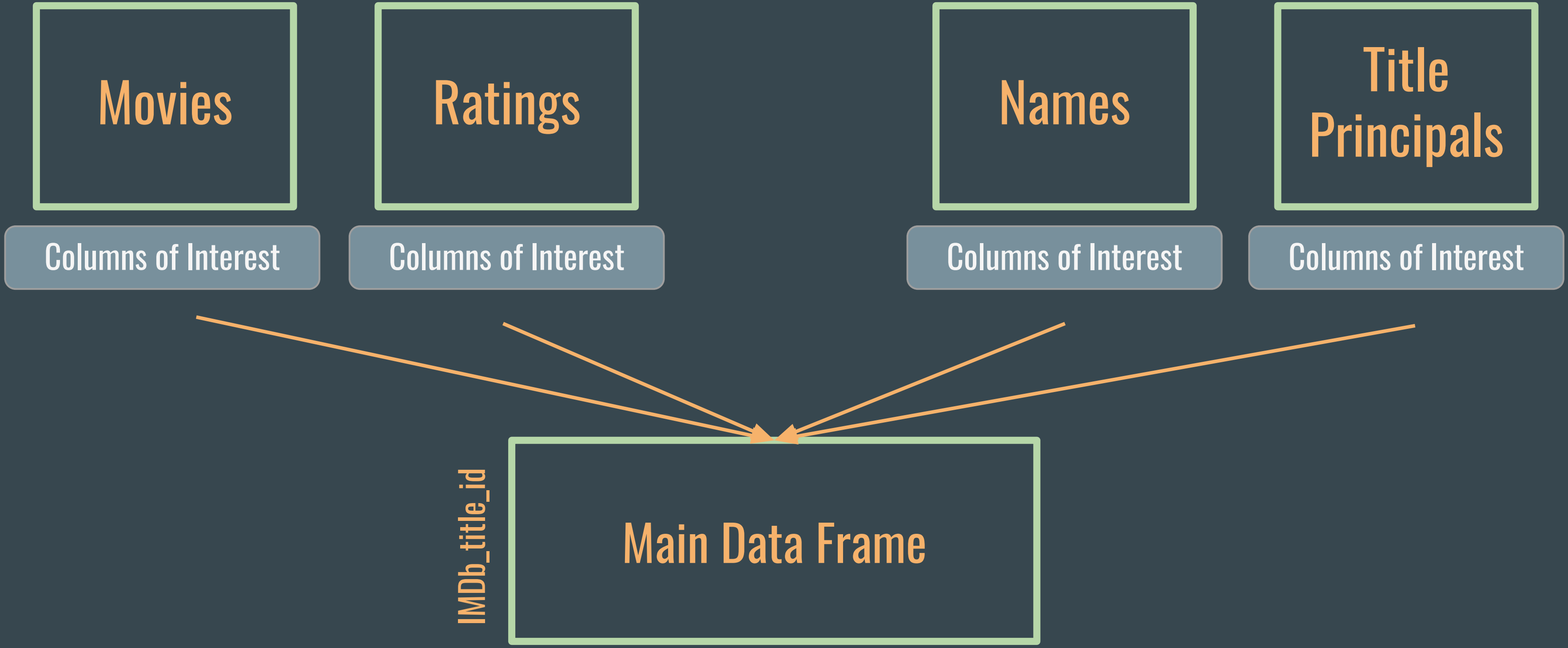
Names

Actor ID

Title Principals

Ordering

Preprocessing: Initial steps



Preprocessing: Feature Transformation

Movies

Date Published
Budget
Worldwide Income
Domestic (USA) Income
Genre
Country
Director
Writer
Actors
Production Company
Title
Description

Date Published

“11/02/1997”

Date Published Month	Date Published Day	Date Published Year
11	02	1997

Preprocessing: Feature Transformation

Movies

Date Published
Budget
Worldwide Income
Domestic (USA) Income
Genre
Country
Director
Writer
Actors
Production Company
Title
Description

1. All non-US dollar values dropped
2. All remaining values adjusted for inflation based on 2021 CPI

Budget		Budget Adjusted
$ 500000	→	11722649.57
(1917)		(2021)

Preprocessing: Feature Transformation

Movies

Date Published
Budget
Worldwide Income
Domestic (USA) Income
Genre
Country
Director
Writer
Actors
Production Company
Title
Description

1. Genre Combinations Encoded
2. Transformed into a binary string
3. Binary string expanded into 10 individual columns

Genre		Genre_1 ... Genre_10
'Action, Comedy, Romance'	→	0 - 0 - 1 - 1 - 0 - 1 - 0 - 0 - 1 - 1

Preprocessing: Feature Transformation

Movies

Date Published
Budget
Worldwide Income
Domestic (USA) Income
Genre
Country
Director
Writer
Actors
Production Company
Title
Description

1. First listed country taken as primary country
2. Primary country mapped to region using UN ISO country/region codes

Country		Region
'USA, UK, Italy'	→	Americas

Preprocessing: Feature Transformation

Names

Title Principals

Movies

1. Popularity calculated by frequency
2. Importance calculated by order of mention
3. Weighted Popularity calculated

Date Published
Budget
Worldwide Income
Domestic (USA) Income
Genre
Country
Director
Writer
Actors
Production Company
Title
Description

Director	Director Weighted Popularity
'Steven Spielberg' →	0.00015

Preprocessing: Feature Transformation

Movies

Date Published
Budget
Worldwide Income
Domestic (USA) Income
Genre
Country
Director
Writer
Actors
Production Company
Title
Description

1. Popularity calculated by frequency

Production Company		Production Company Popularity
'Warner Bros.'	→	0.115

Preprocessing: Feature Transformation

Movies

Date Published
Budget
Worldwide Income
Domestic (USA) Income
Genre
Country
Director
Writer
Actors
Production Company
Title
Description

Title

"Harry Potter and the Sorcerer's Stone"

# Words	Ratio long words	Ratio capital letters	Ratio vowels
6	0.66	0.12	0.32

Preprocessing: Feature Transformation

Movies

Duration

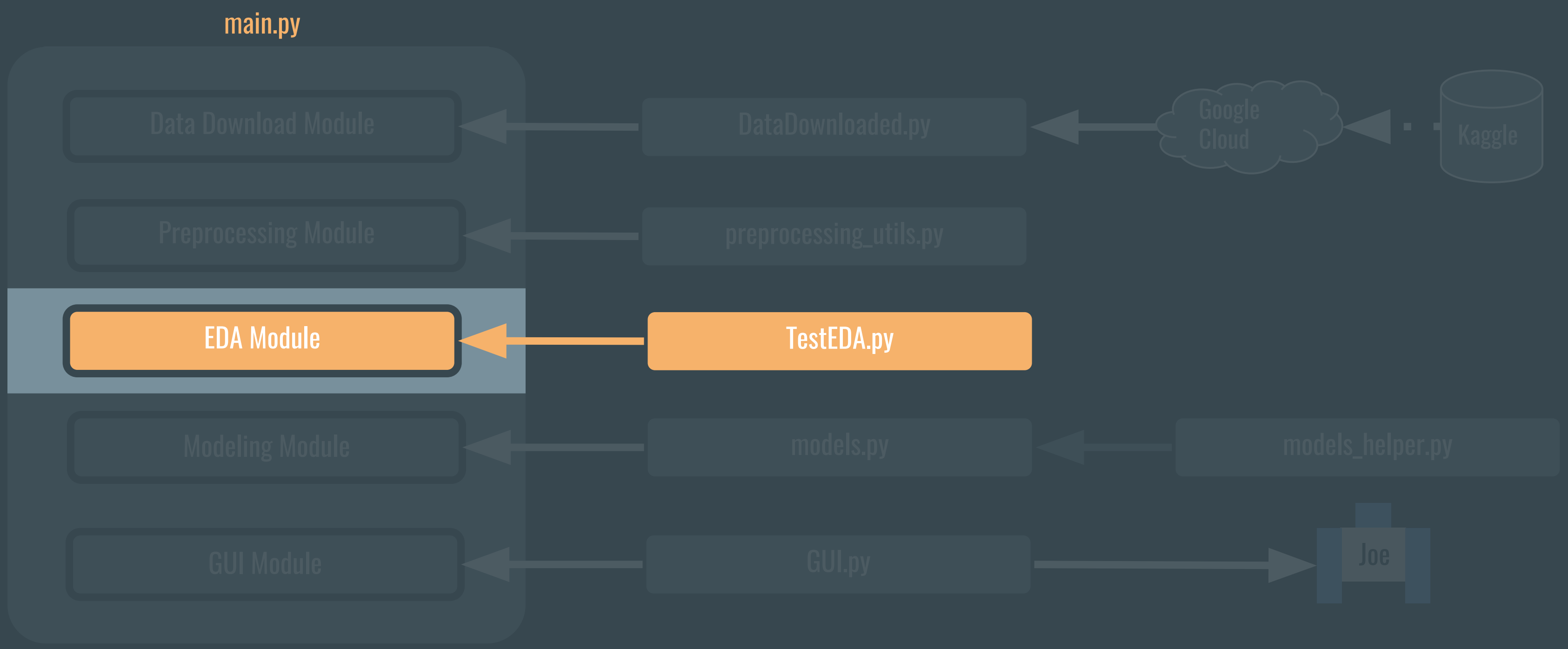
Ratings

Weighted Average Vote

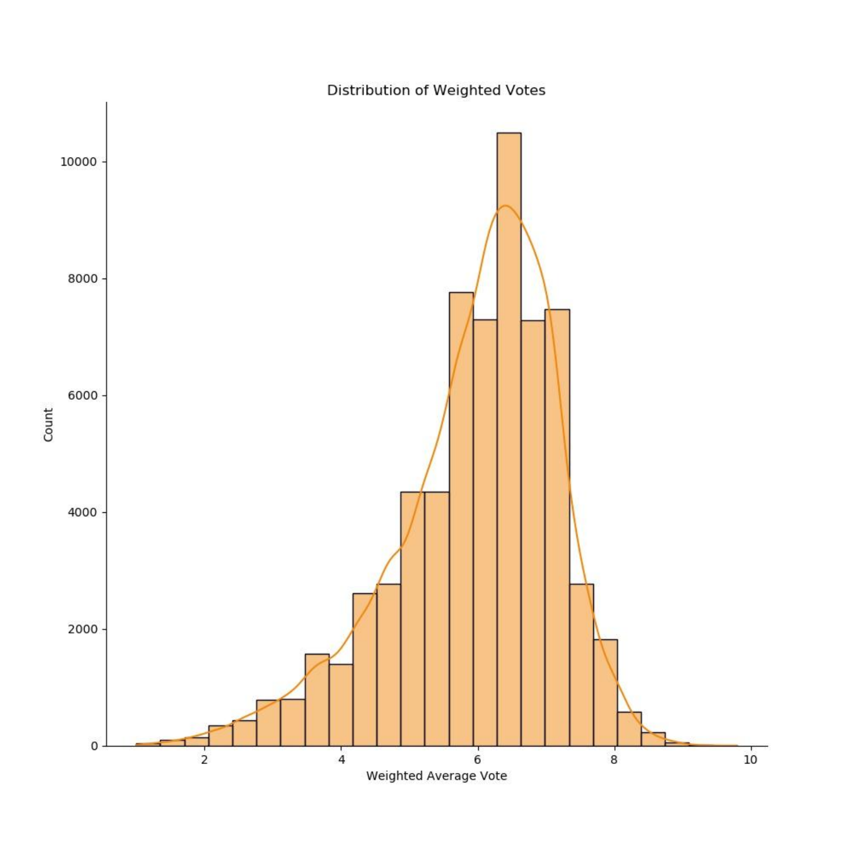
Preprocessing: Final Steps

1. Data set was split into train, test, and validation
2. All missing values were imputed using the mean
3. All values were scaled

Preprocessing: Code Architecture

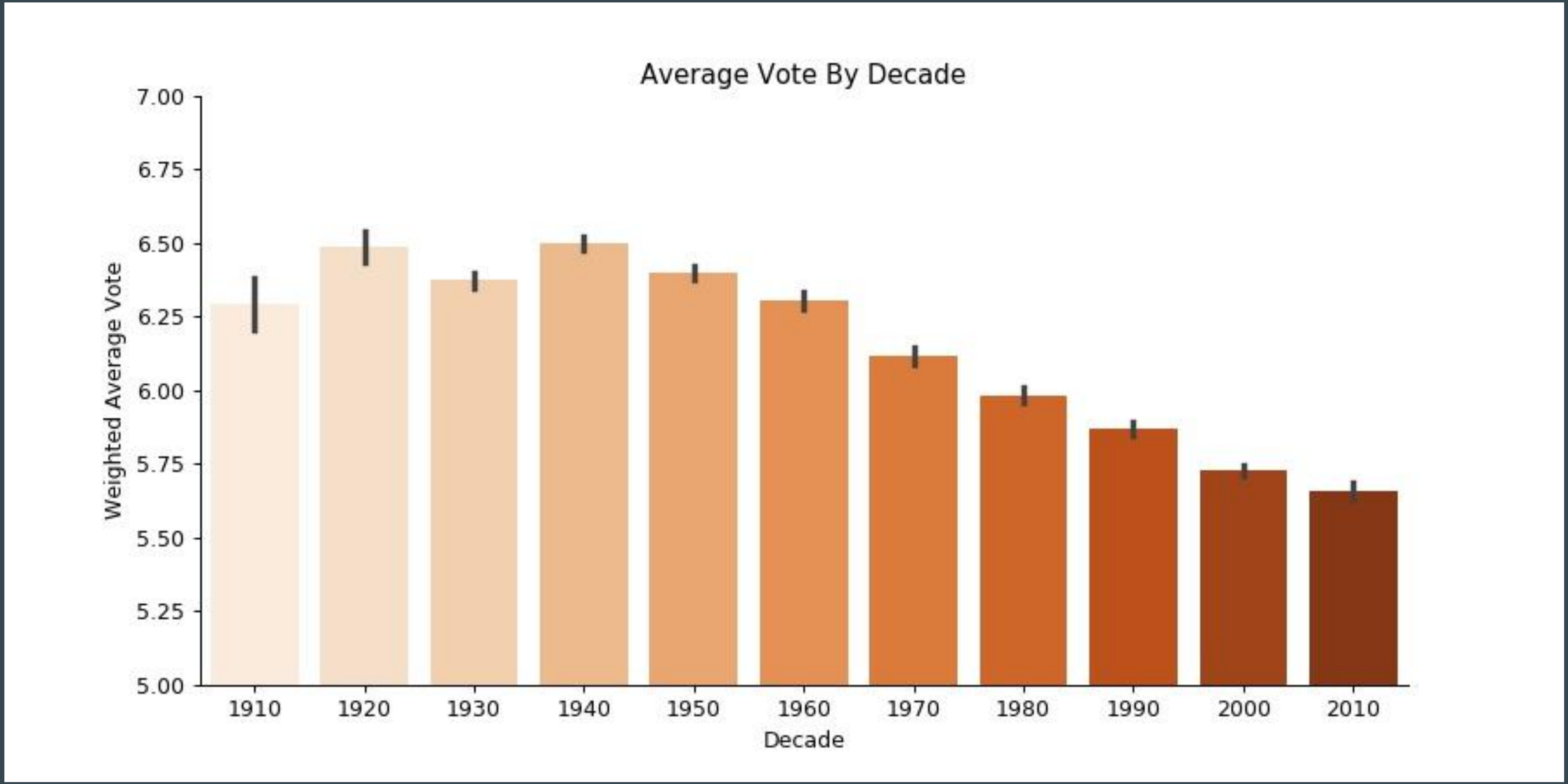


EDA: Target Variable

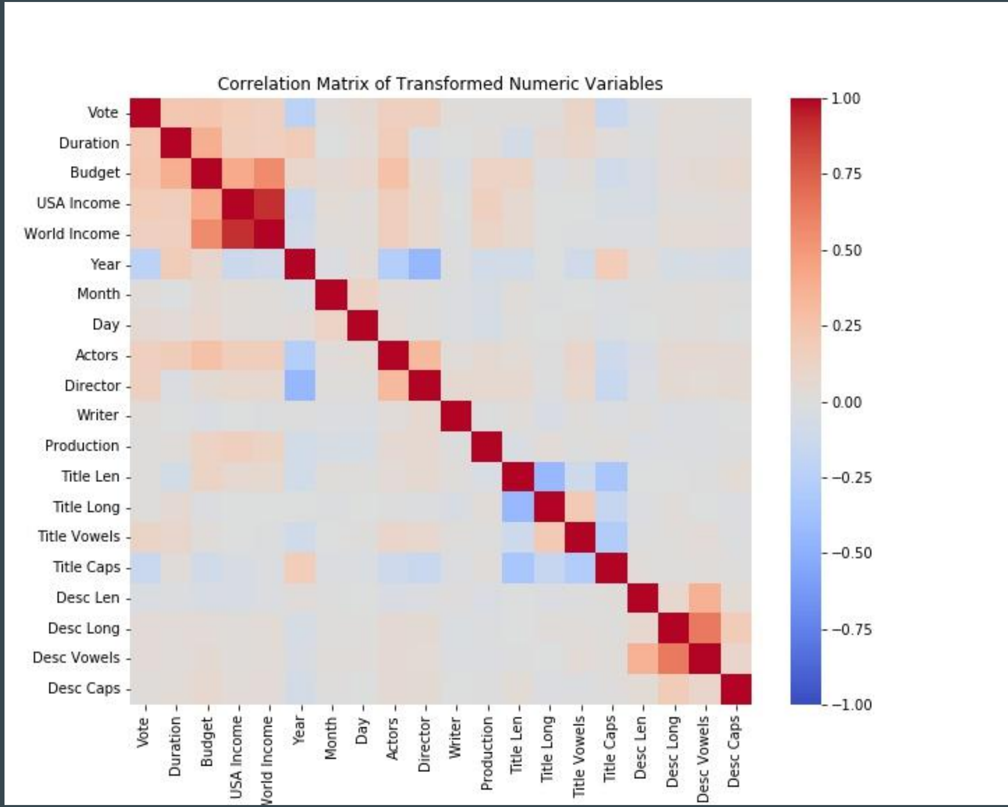


Possible range: 1 - 10
Mean: 5.96
Standard Deviation: 1.19

EDA: Vote by Decade of Release



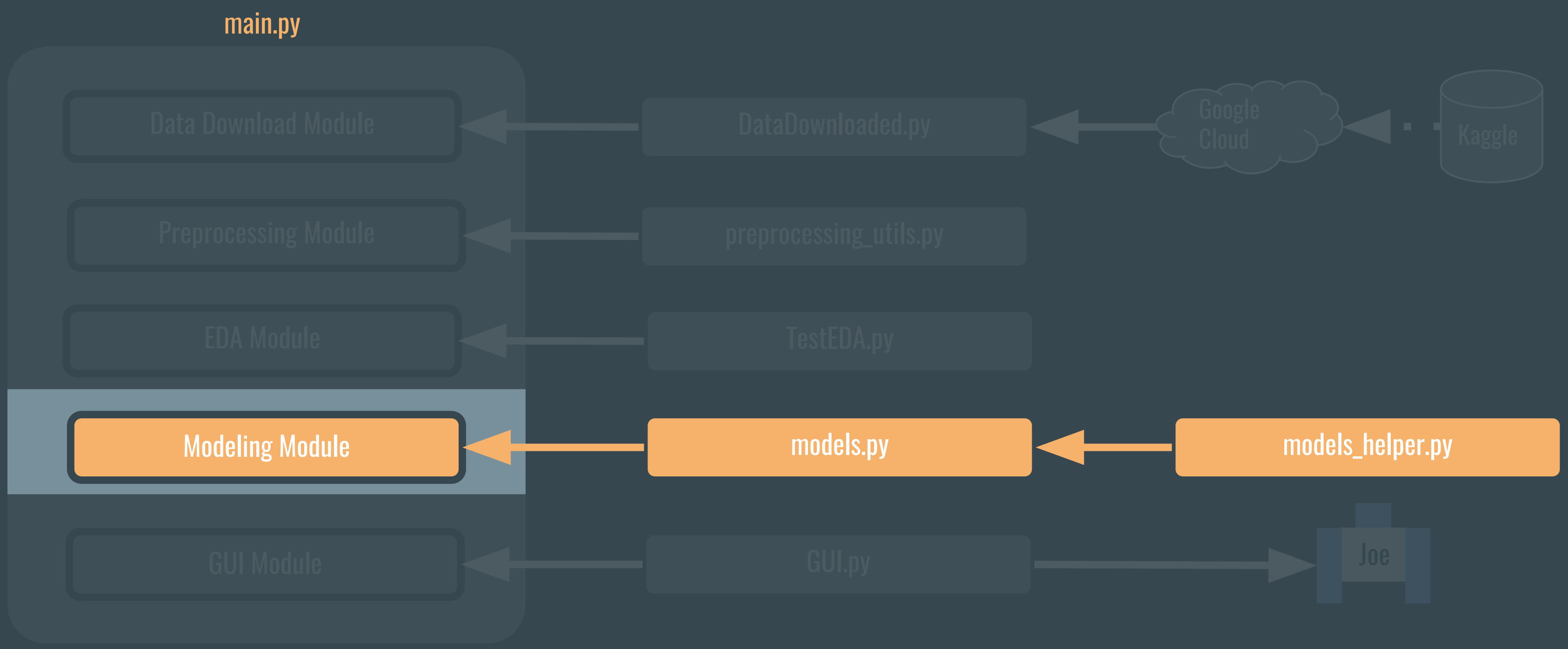
EDA: Correlation



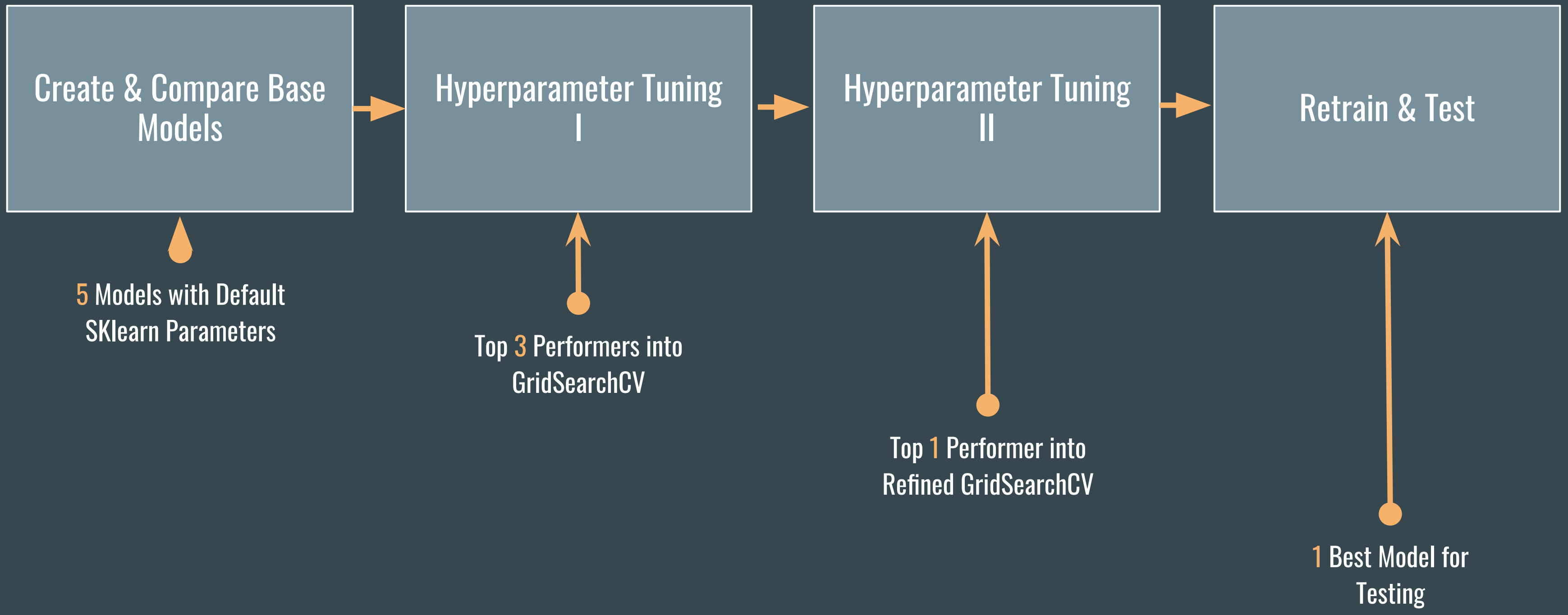
Modeling: Objective

Create a model that fits well to our dataset and would allow us to predict a satisfiable ***weighted average IMDb rating***.

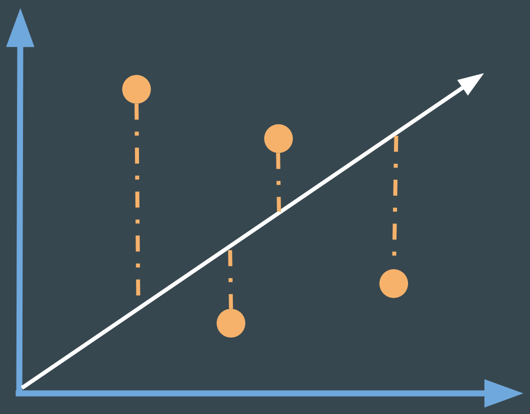
Modeling: Code Architecture



Modeling: Methodology



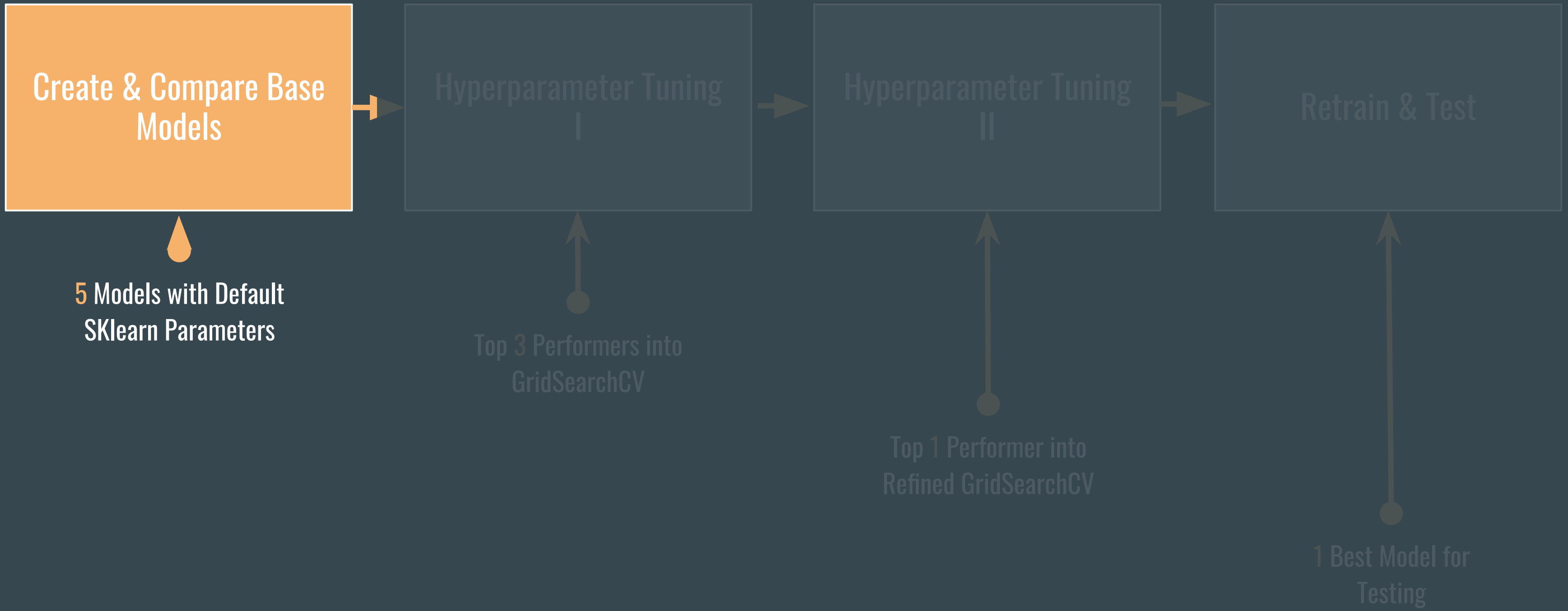
Modeling: Scoring



$$MSE = \frac{1}{n} \sum_{i=1}^{n} (y_{actual,i} - y_{predicted,i})^2$$

, where n is the number of observations in the dataset and i is the ith observation in n.

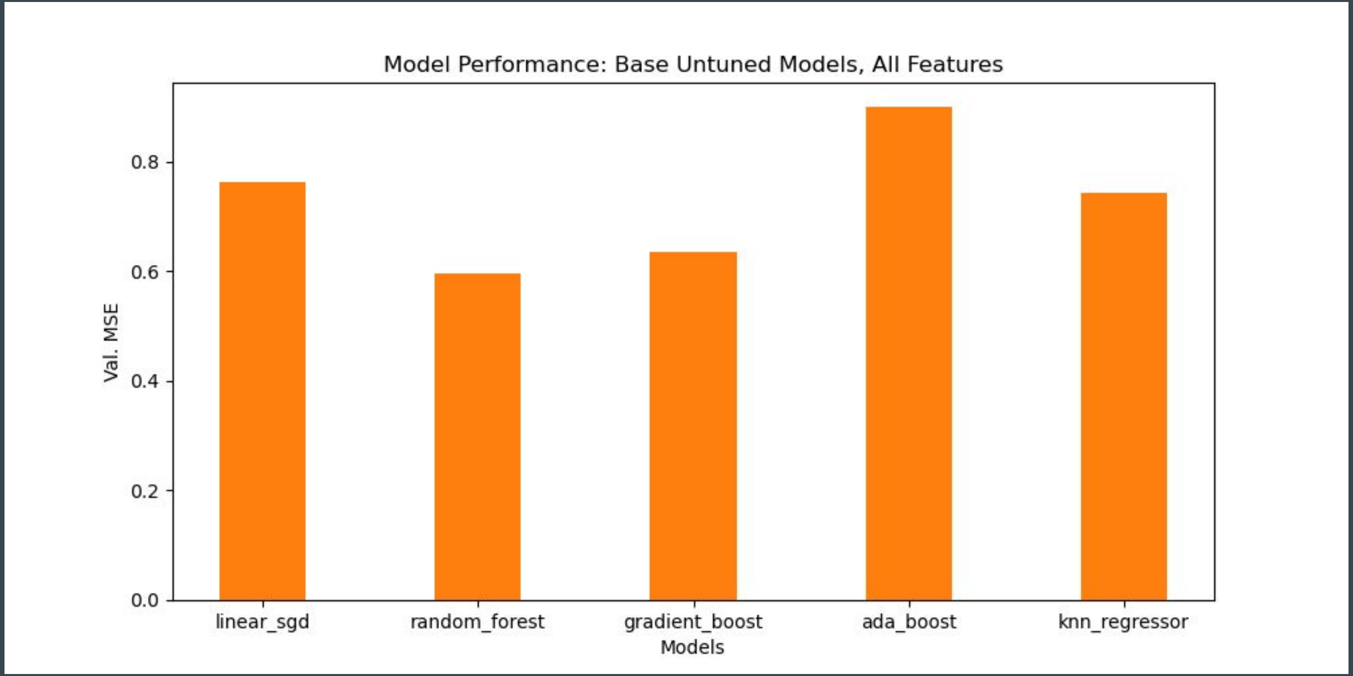
Modeling: Base Models



Modeling: Base Models

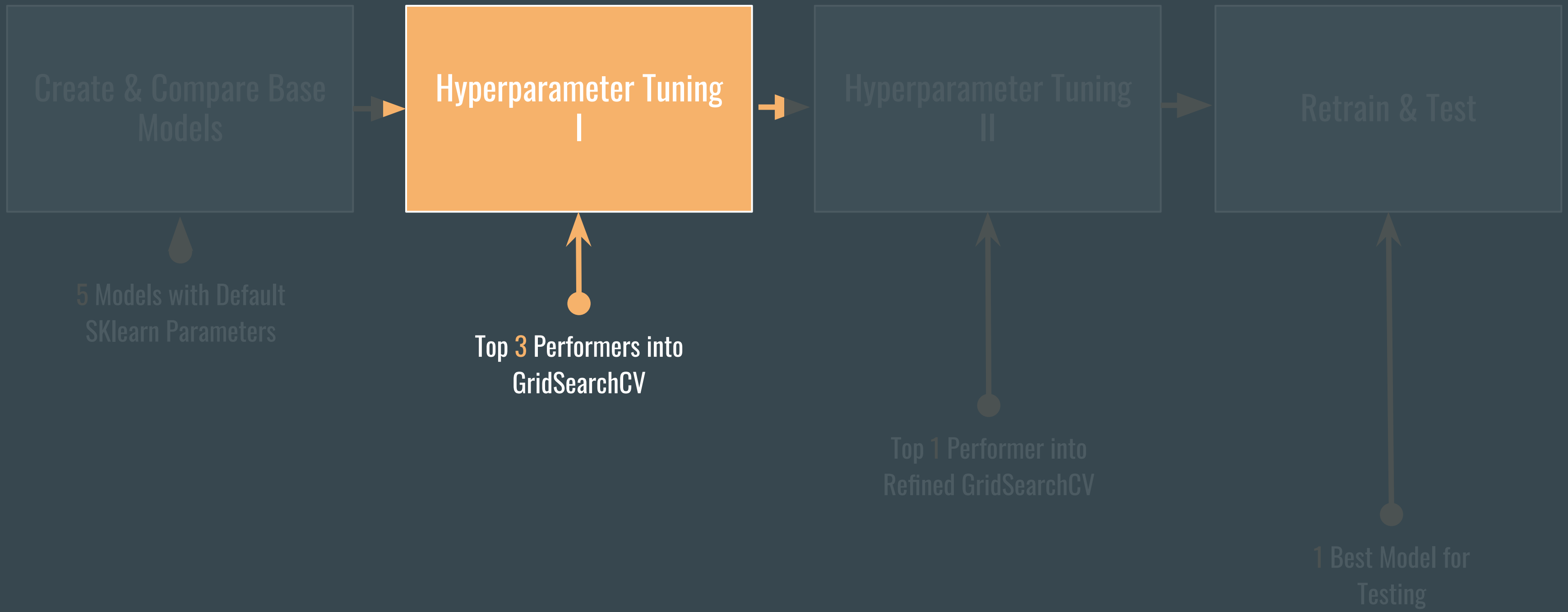
Model Type	SKlearn's Object	Pros	Cons
Linear Regression	SGDRegressor()[6]	Simple to implement and understand.	Doesn't work well for non-linearly separable dataset.
Random Forest	RandomForestRegressor()[7]	Less prone to overfitting.	Large number of trees can be slow to use. Can't extrapolate on data outside range of trained data.
Gradient Boosting	GradientBoostingRegressor()[8]	Fits each subsequent tree on the residuals which can learn very well.	Generally slower to fit than Random Forest and more prone to overfitting.
Adaptive Boosting	AdaBoostRegressor()[9]	Uses many decision stumps and each stump gets a weighted vote.	Not as robust to outliers as it tries to fit to every datapoint.
K-Nearest Neighbors	KNeighborsRegresor()[10]	Simple to understand and low number of hyperparameters.	Not as efficient as dataset grows.

Modeling: Base Models' Results

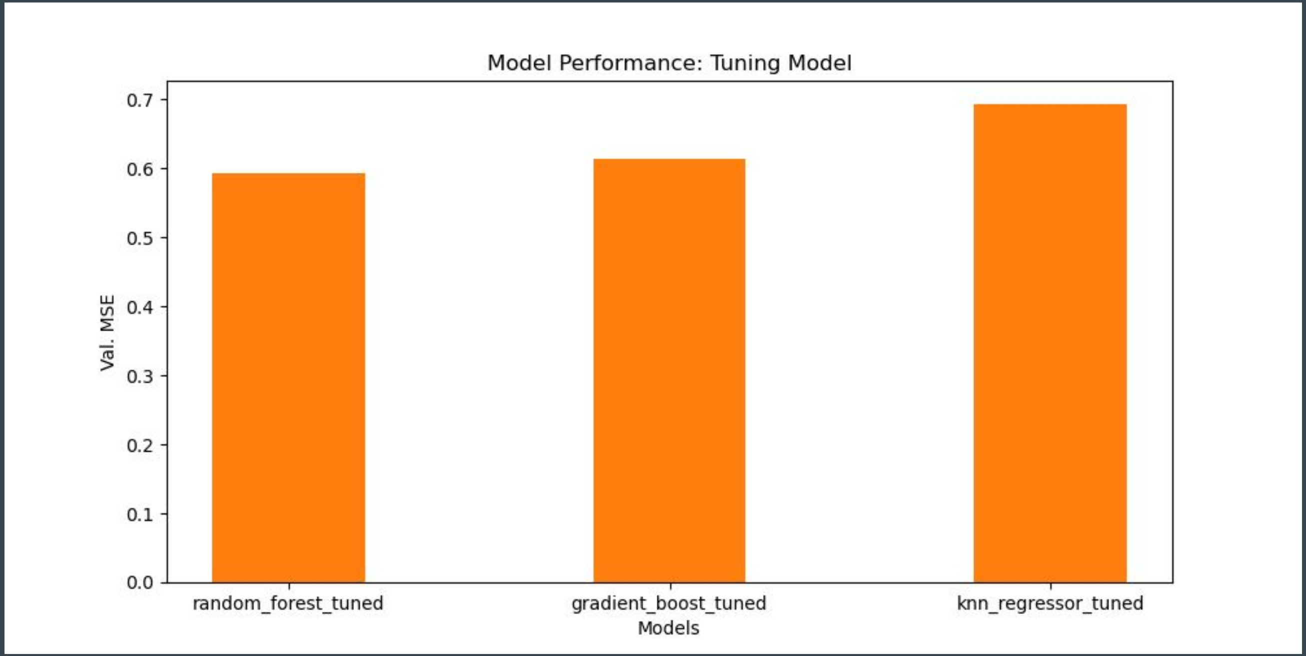


Modeling: Base Models’ Results

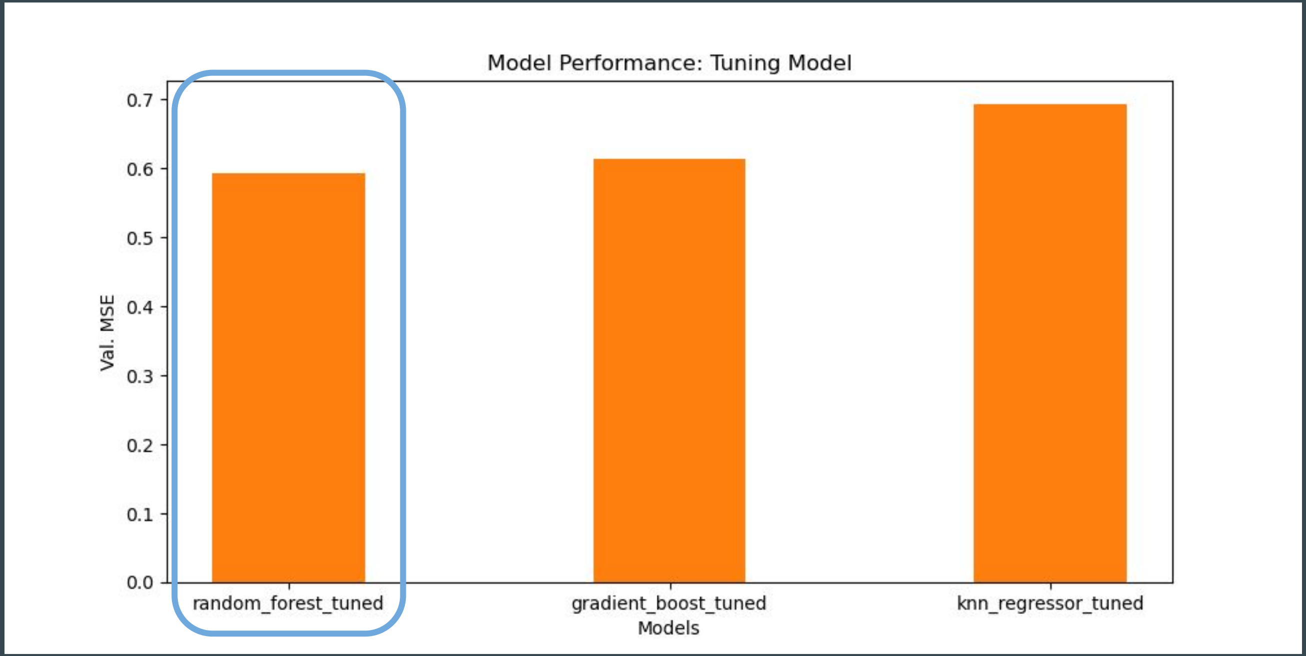
Modeling: Hyperparameter Tuning I



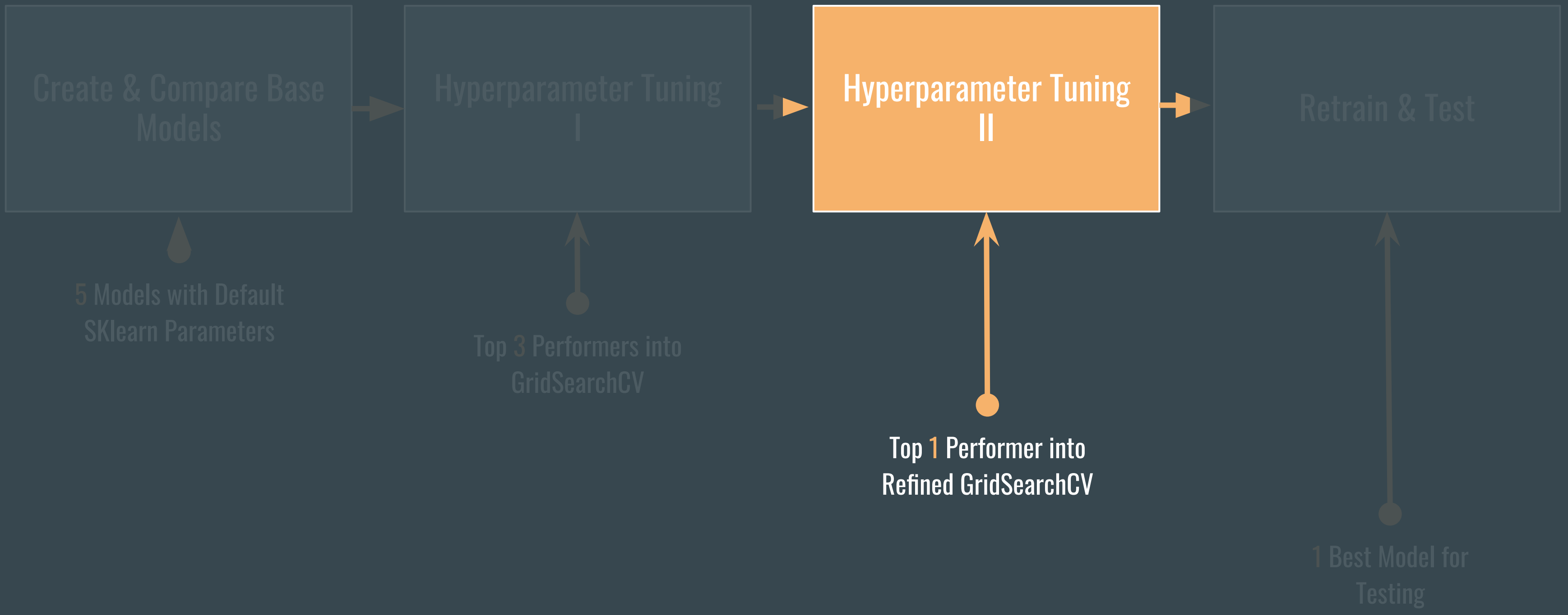
Modeling: Hyperparameter Tuning I



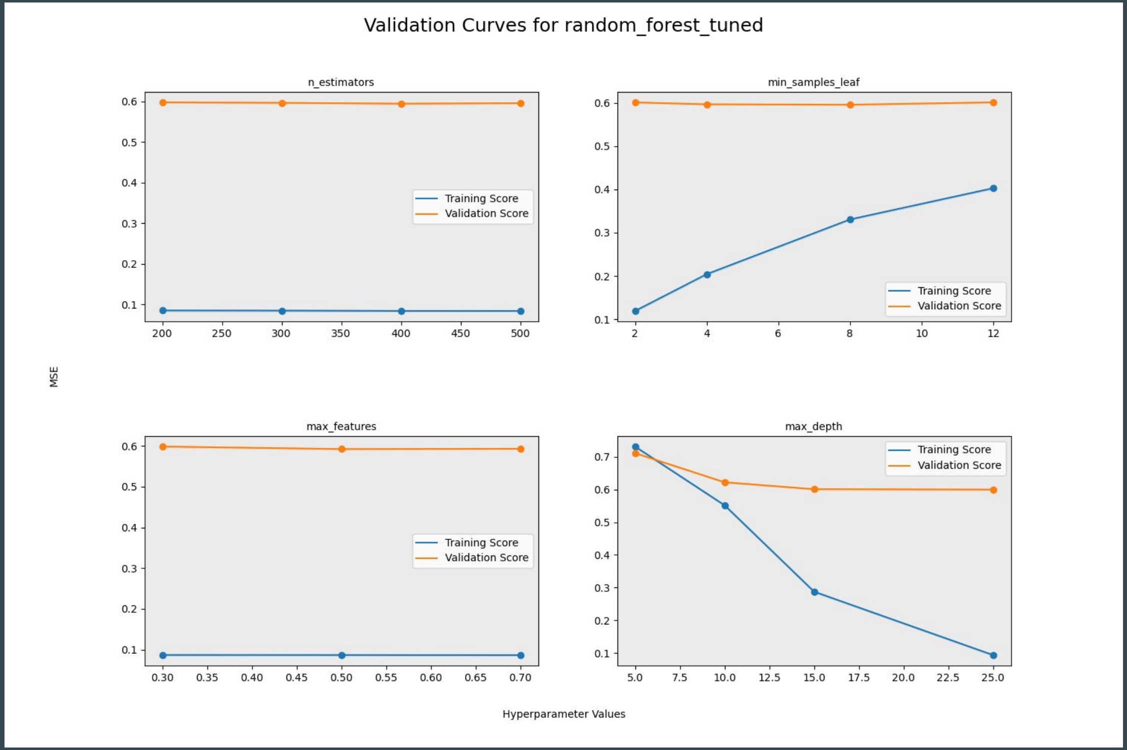
Modeling: Hyperparameter Tuning I



Modeling: Hyperparameter Tuning II

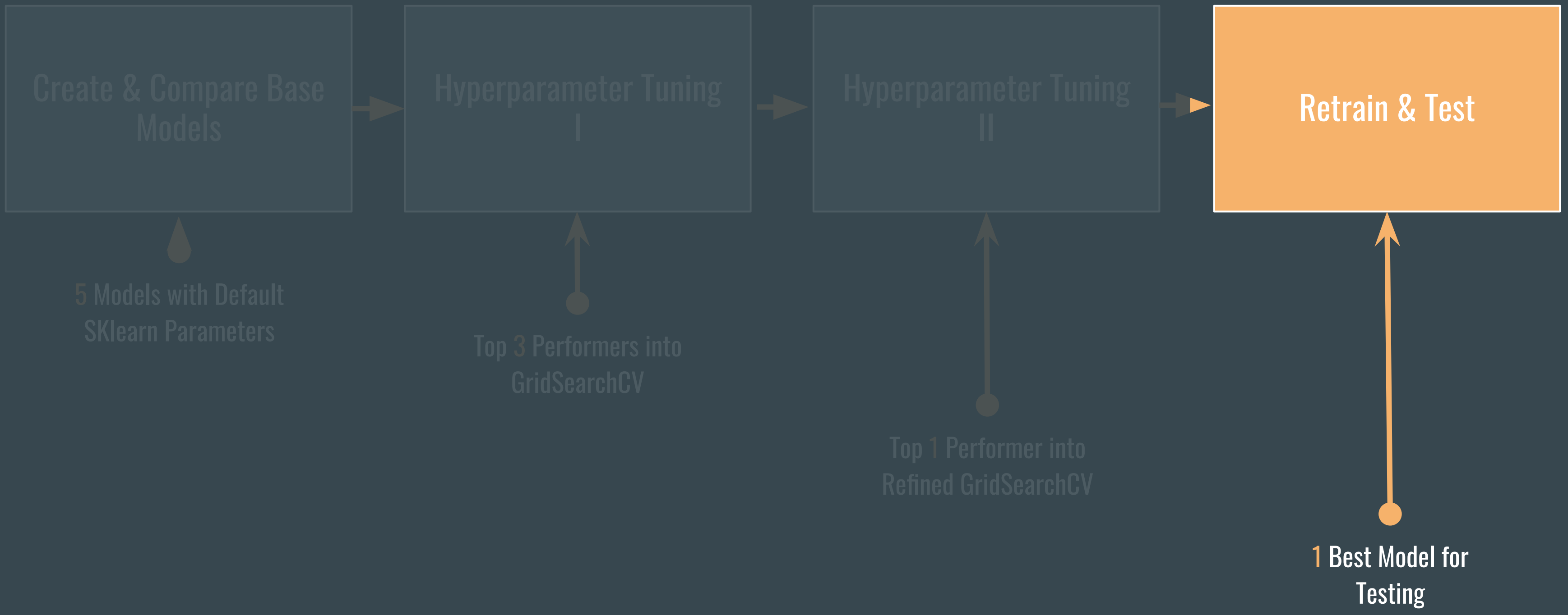


Modeling: Hyperparameter Tuning II

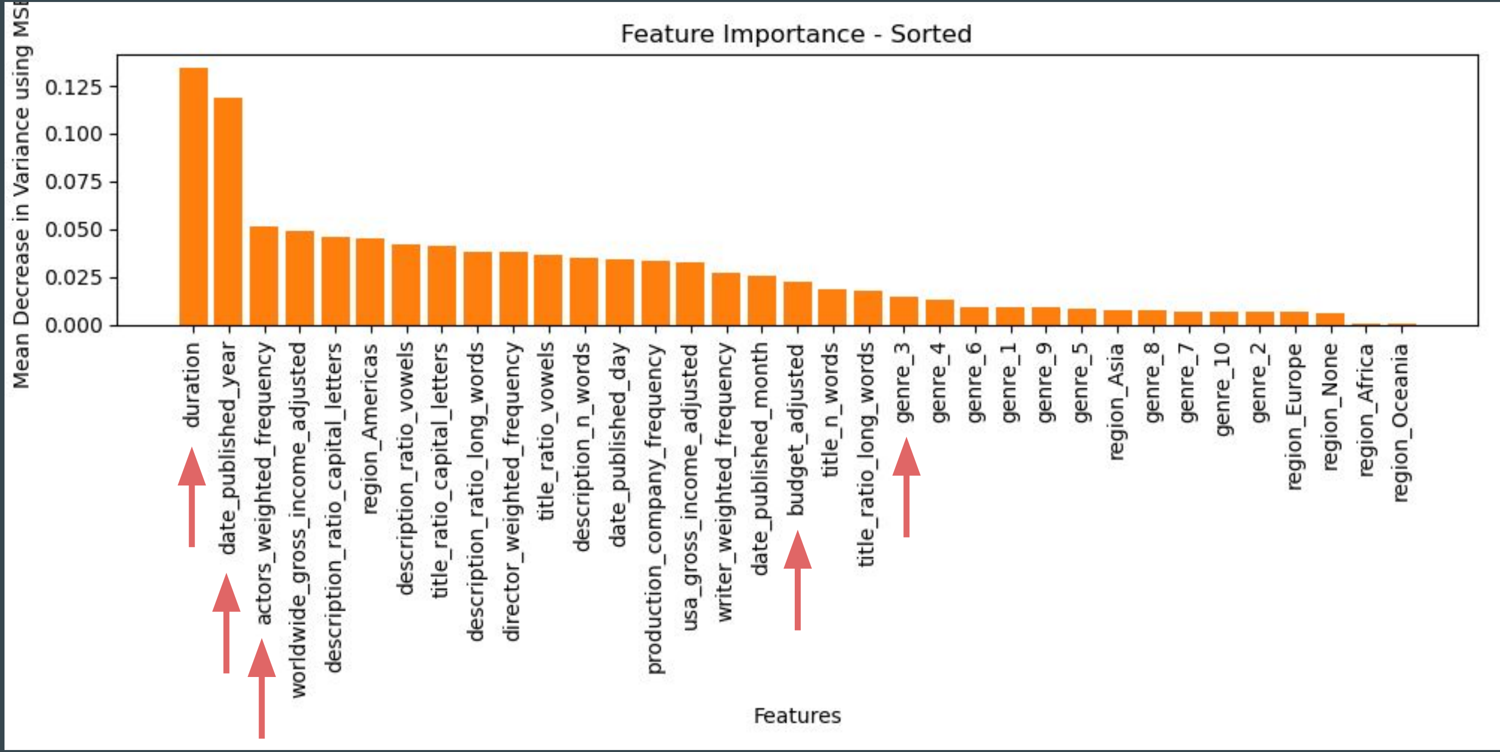


Best Performing Hyperparameters
n_estimators = 500
min_sampes_leaf = 2
max_feature = 0.7
max_depth = 25

Modeling: Retrain & Test

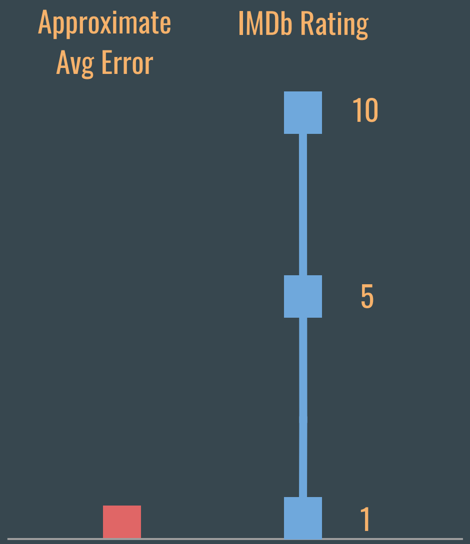


Modeling: Retrain & Test



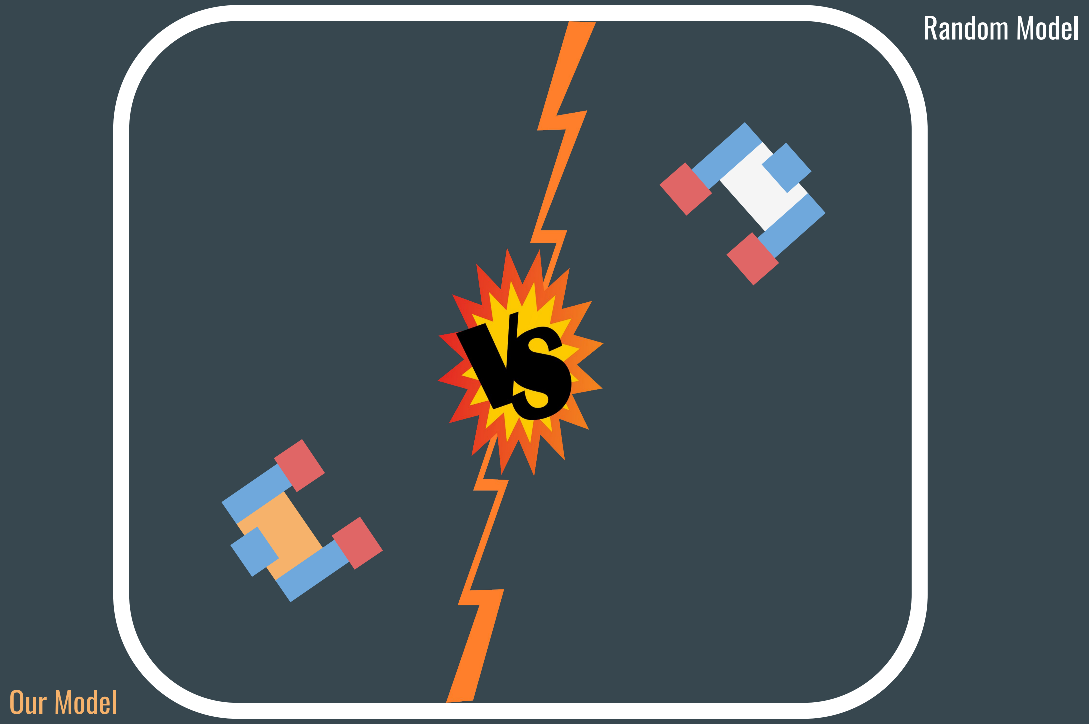
Results: Test Data

Testing Results
MSE = 0.602
MSE (Inverse Scaling) = 0.869
RMSE (Inverse Scaling) = 0.931



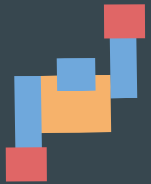
*Not to scale, just to gain some intuition into how big or small our error is compared against the IMDb Rating range

Results: Model Evaluation



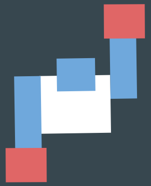
Results: Model Evaluation

Our Model



- (4-0): Beat out 4 other models and many more hyperparameter combinations
- Powered by Random Forest
- Highly trained

Random Model



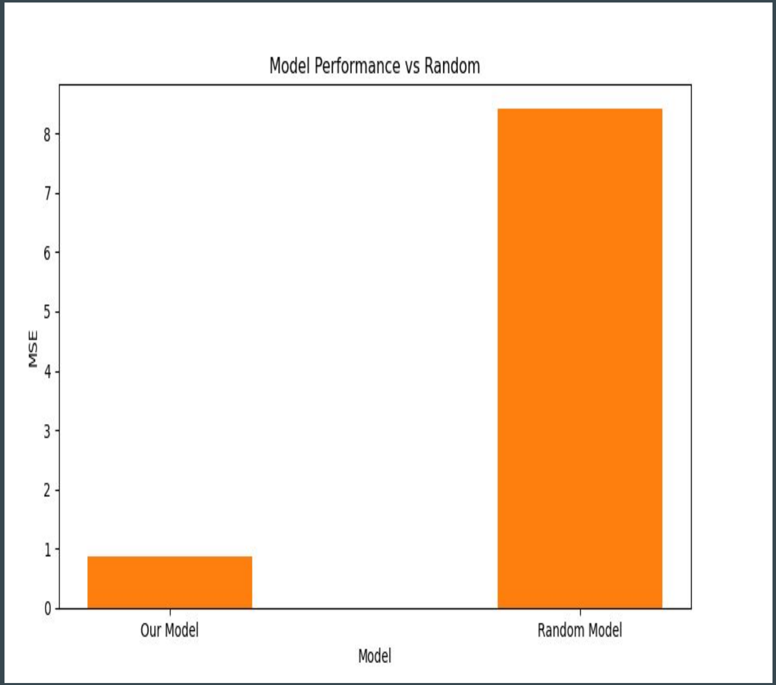
- Is literally just this:

```
test_Y_random = np.random.uniform(1, 10,
size=test_Y.shape)
```

- Thinks he's really good at guessing

Place Your Bets!

Results: Model Evaluation - vs Random Model



Actual Rating	*Our Model* Predicted Rating	*Random Model* Predicted Rating	Smaller Error
4.7	5.3	5.4	RM
7.2	6.2	9.7	OM
6.8	6.7	4.5	OM
6.4	6.2	1.7	OM
6.5	5.6	4.2	OM
6.3	6.0	2.5	OM
5.7	6.8	9.8	OM
7.2	6.0	8.9	OM
4.4	6.3	5.4	RM
6.8	6.4	4.6	OM

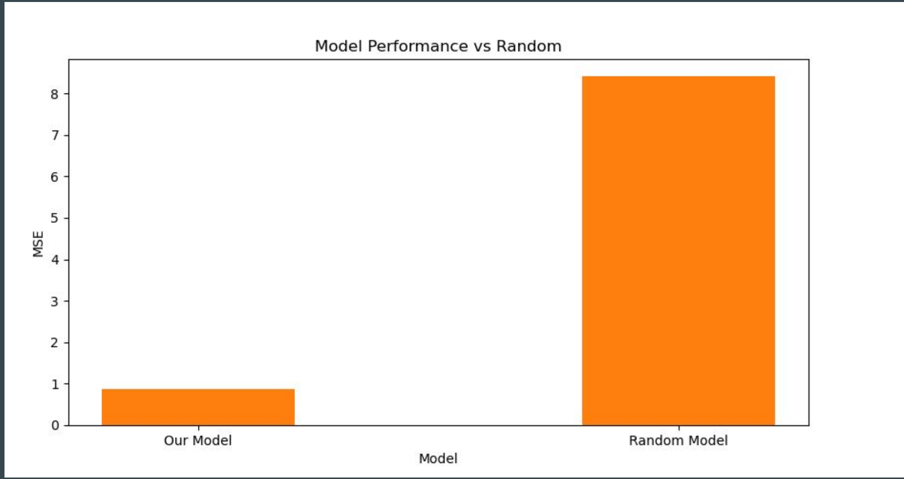
Our model has smaller error 84% of the time in test data

Results: Model Evaluation - Distributions Are Different?

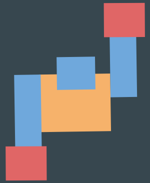
Mann Whitney U Test

Test	H0	p-value	Action
Mann-Whitney U Test	The 2 distributions are equal	3.80e-29	Reject H0

Results: Model Evaluation - vs Random Model



WINNER!!



Our Model

Our Model is statistically different from Random Model and is a better predictor!

Conclusions

There is a relationship between movies’ features and their IMDb score

We were able to successfully create a model that accurately predicts IMDb score based on a movie’s features

Areas of Improvement

Find a dataset on actors w/o missing values

Create a better encoding method for genre

- Group movies based on their genres

Reduce the dimensionality of our data

Increase efficiency and speed of modelling

Future Research

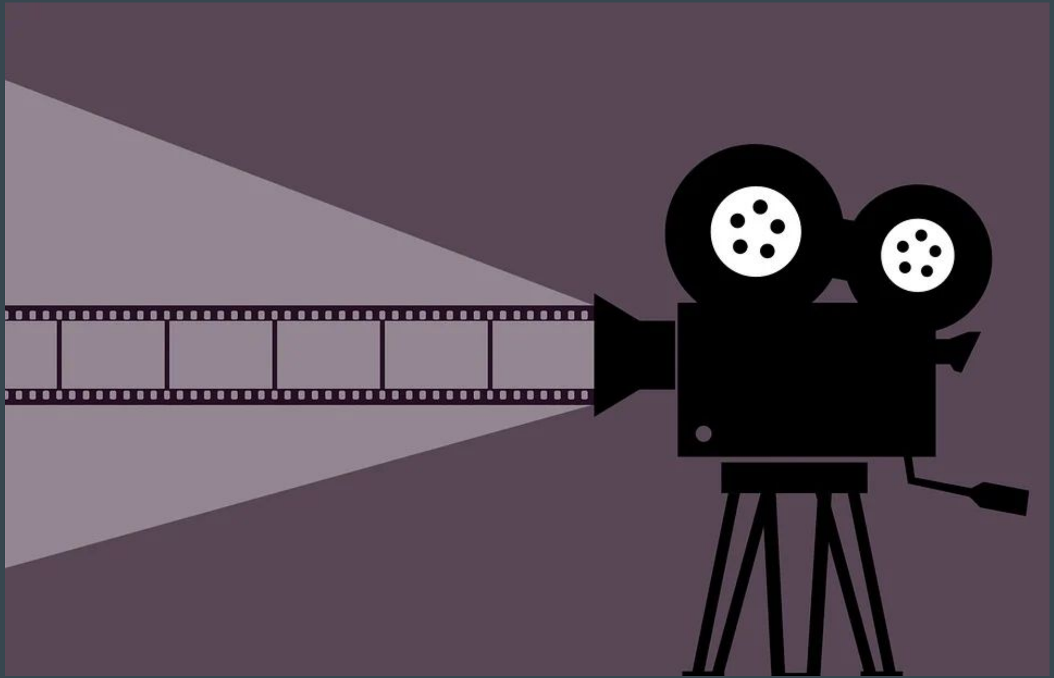
- Different target variables
  - Male versus female votes
  - Gross income
- Titles and description
  - Natural language processing
  - Movie scripts
- Other variables?
  - Screentime
  - Sequel vs non-sequel
  - Release location/method
- Different modeling techniques
  - Voting regressor model
  - Different modeling packages
  - More advanced modeling techniques

The Bigger Picture

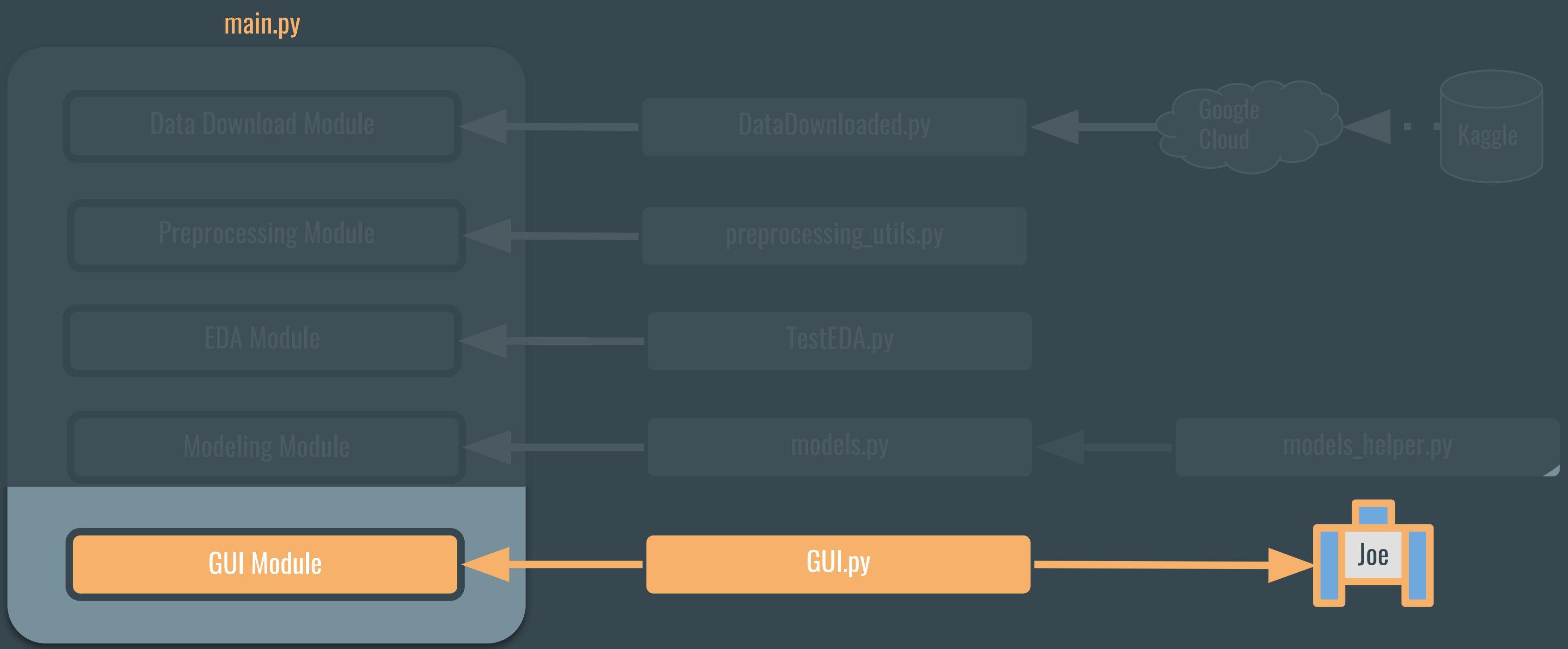
What does this research mean for the movie industry?

How can predictive analytics be used by movie production companies?

Does this pose any problems?



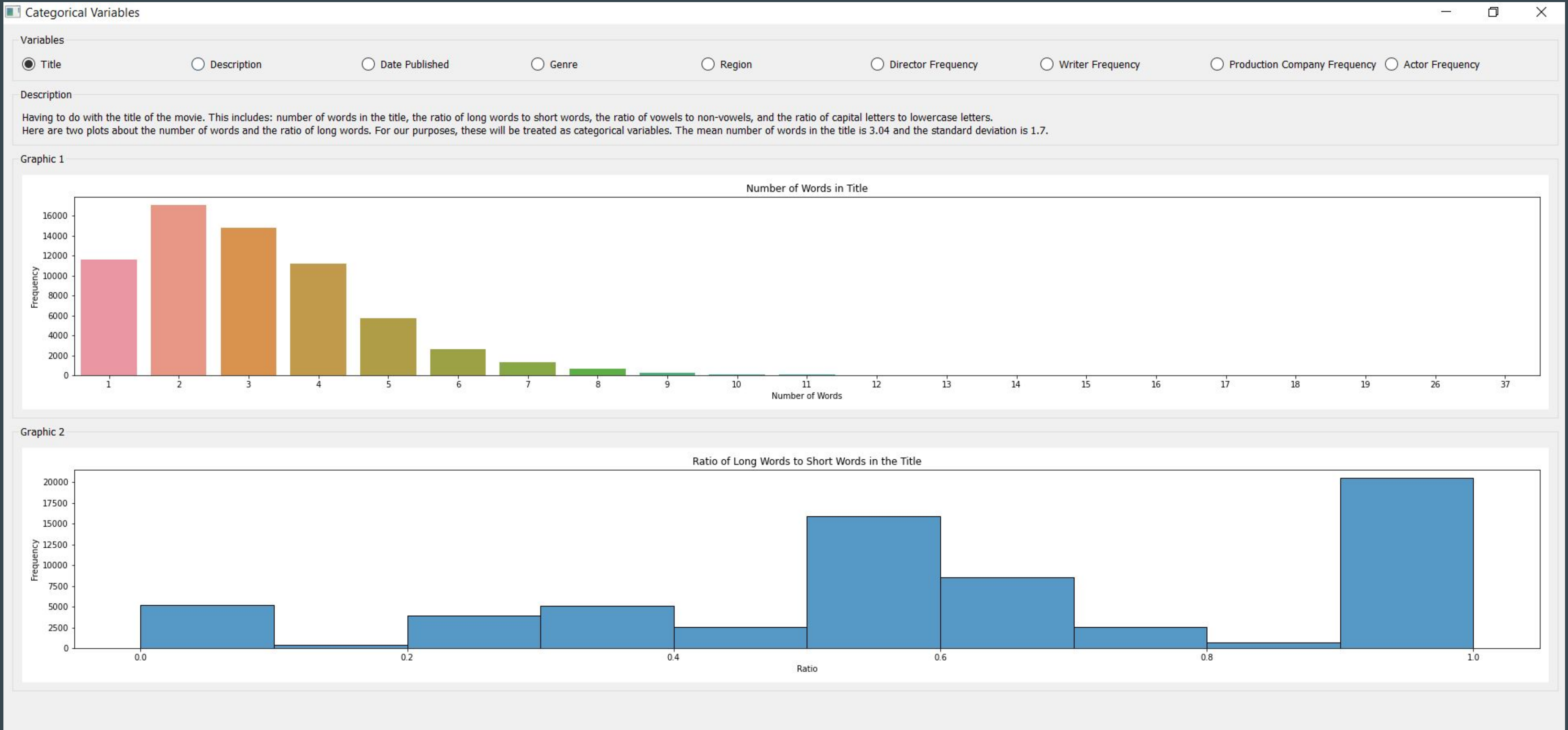
GUI Demo



https://github.com/Saharae/Final-Project-Group2

GUI Images Part 1

GUI Images Part 2



GUI Images Part 3

Hyperparameter Tuning and Validation Phase I

Info

We selected the top 3 base models and tuned them by setting a list of hyperparameters to try in GridSearch validation to see if performance increases. As seen in the Model Comparison graph, our best model still seems to be the Random Forest. Let us focus on just that model in the next phase.

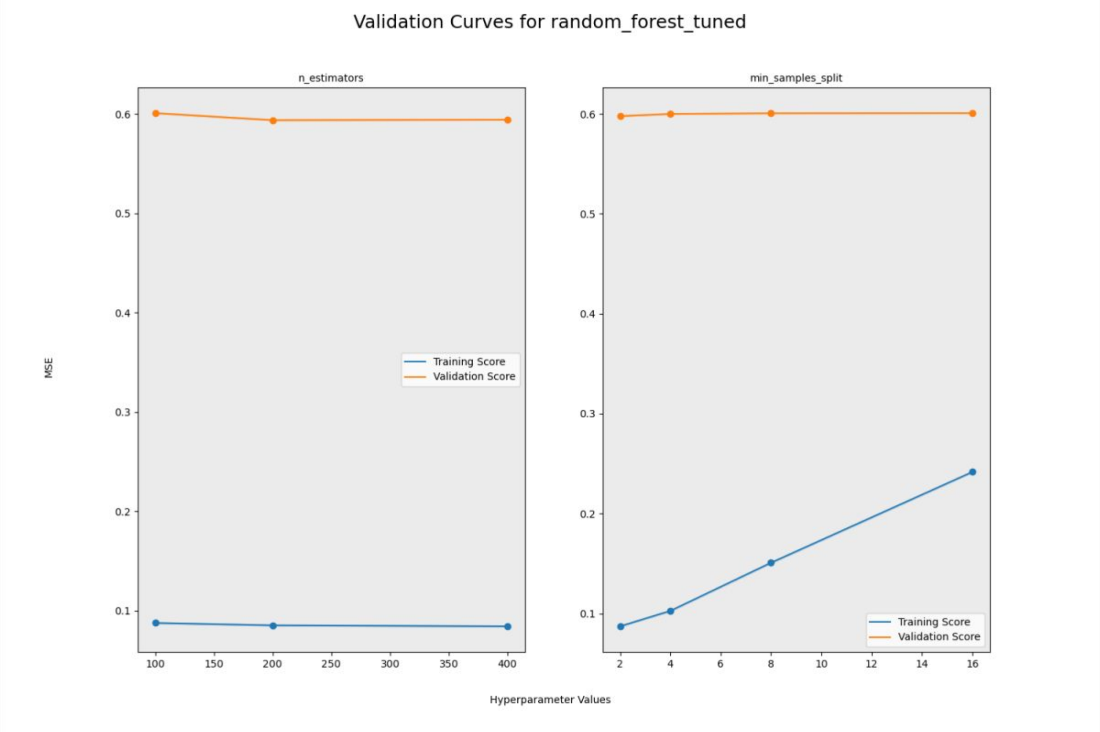
Plot Picker

- Model Comparison
- Random Forest
- KNN
- Gradient Boosting

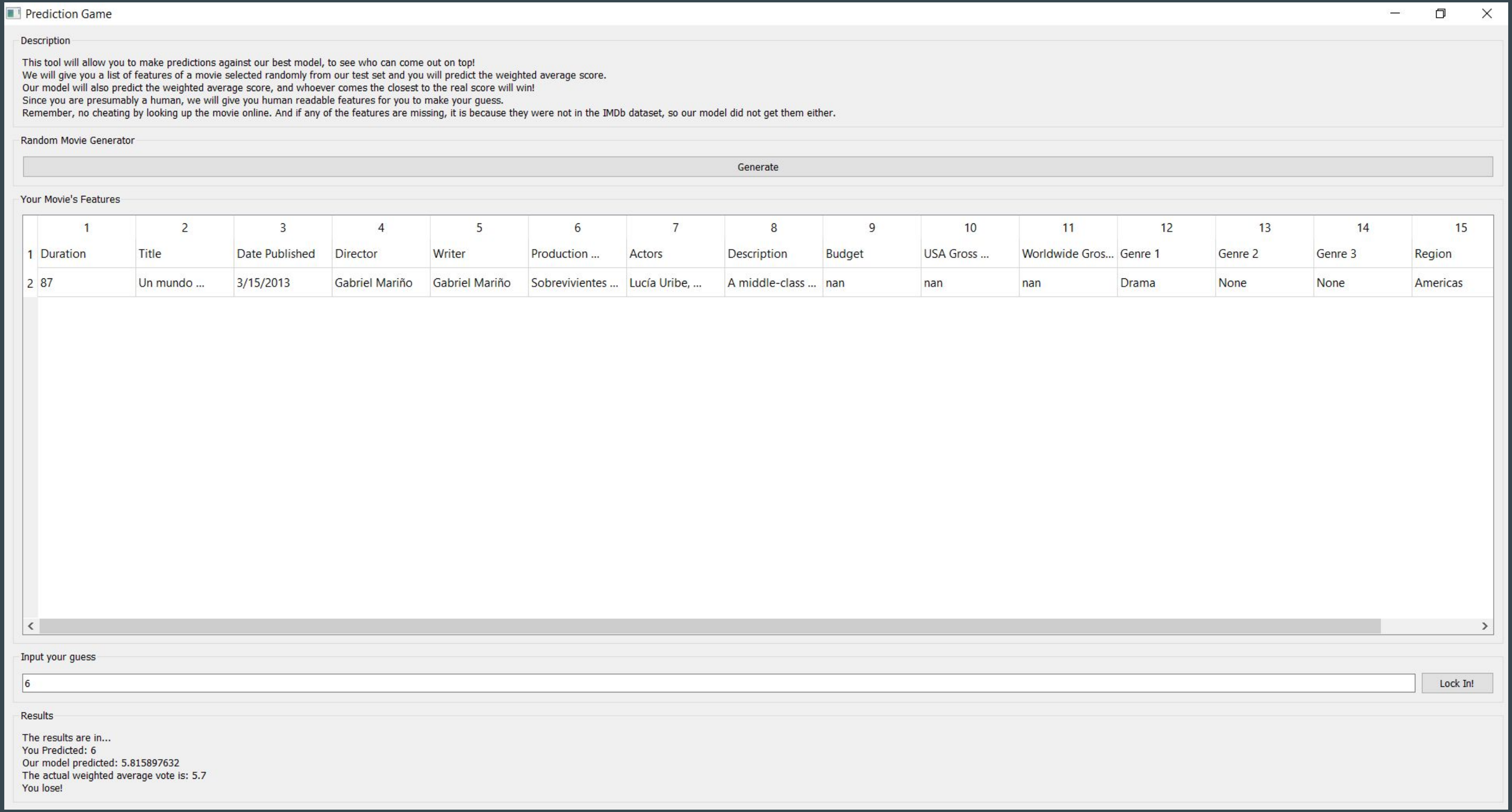
Gridsearch Results

	best_score	best_param	
1	0.5926724032150124	{'min_samples_split': 2, 'n_estimators': 400}	RandomF
2	0.6136042003483392	{'learning_rate': 0.1, 'min_samples_split': 2, 'n_estimators': 400}	GradientB
3	0.6922095267599662	{'n_neighbors': 9, 'p': 1}	KNeighbo

Tuned Plots



GUI Images Part 4



Questions?

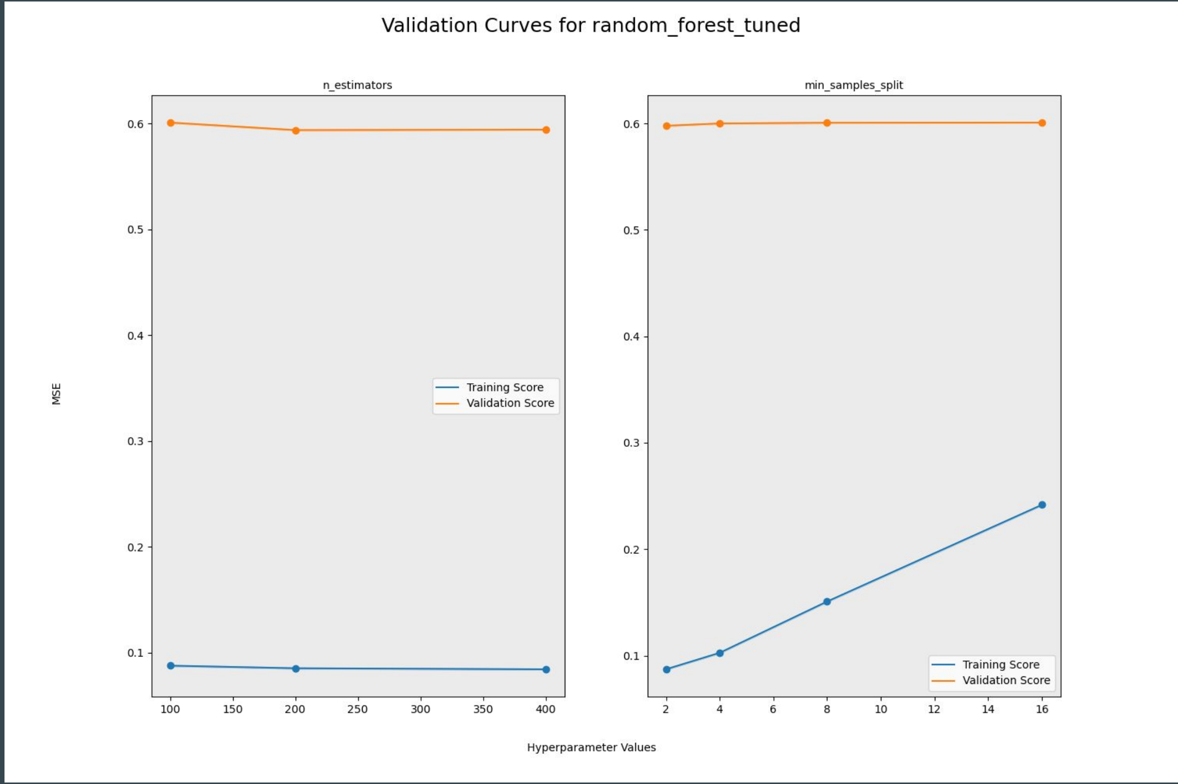
Appendix: Extra Slides

Modeling: Hyperparameters

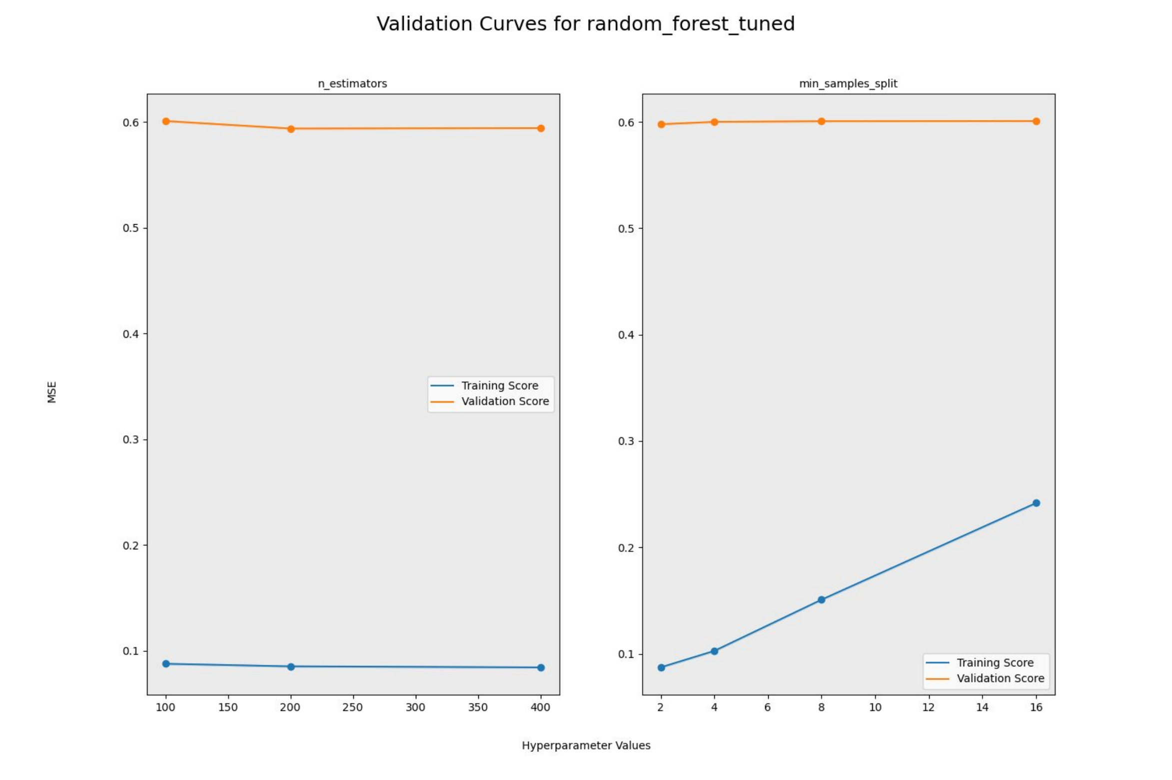
Model Type	Hyperparameter	Hyperparameter Meaning	Dtype and Default
Random Forest*	n_estimators min_samples_split min_samples_leaf max_features max_depth	The number of trees in a forest. Minimum samples required to split an internal node. Minimum samples to be at a leaf node. Number of features to consider for each tree. Max depth allowed for each tree	int, default = 100 int or float, default = 2 int or float, default = 1 {‘auto’, ‘sqrt’, ‘log2’} or int or float, default = ‘auto int, default = None
Gradient Boosting	learning_rate n_estimators min_samples_split	Shrinks the contribution of each tree. The number of boosting stages to use. Minimum samples required to split an internal node.	float, default = 0.1 int, default = 100 int or float, default = 2
K-Nearest Neighbors	n_neighbors p	Number of neighbors to use. p = 1 (Manhattan Distance), p = 2 (Euclidean Distance)	int, default = 5 int, default = 2

*Not all listed hyperparameters tested in this first phase
Source: SKlearn model documentations.

Modeling: Hyperparameter Tuning I

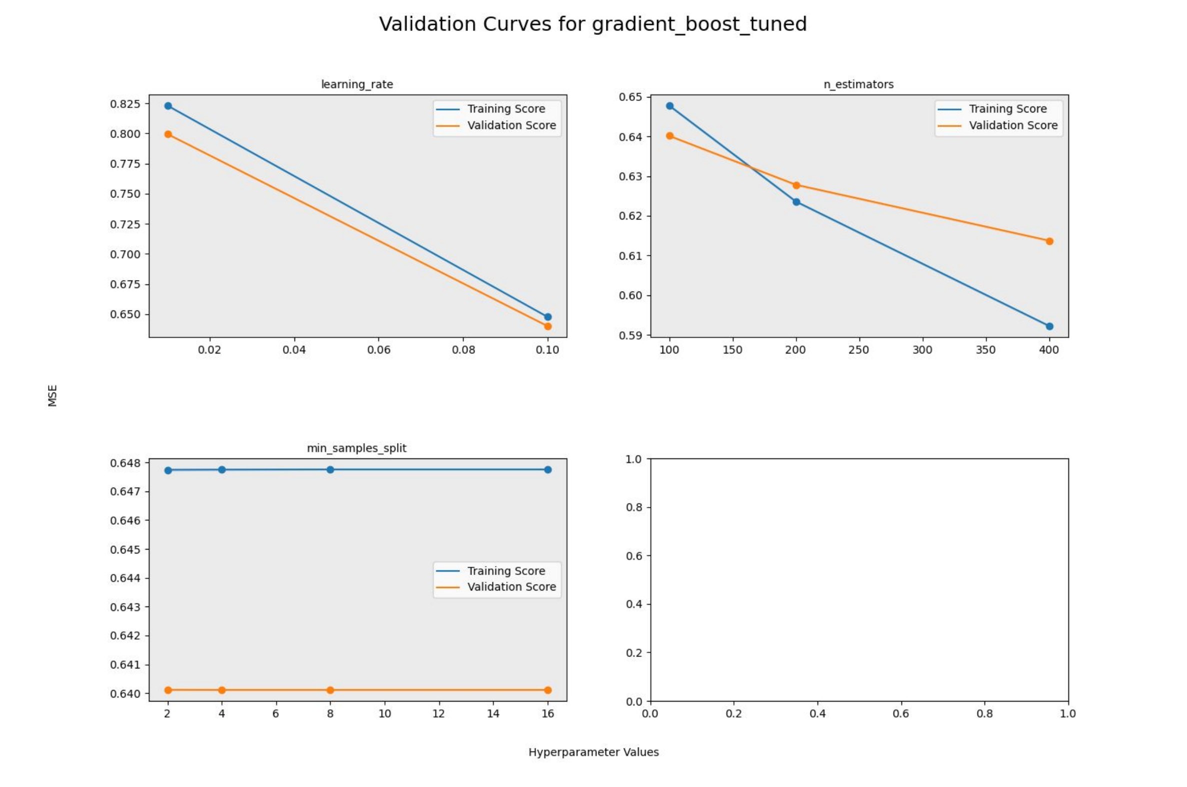


Modeling: Hyperparameter Tuning I

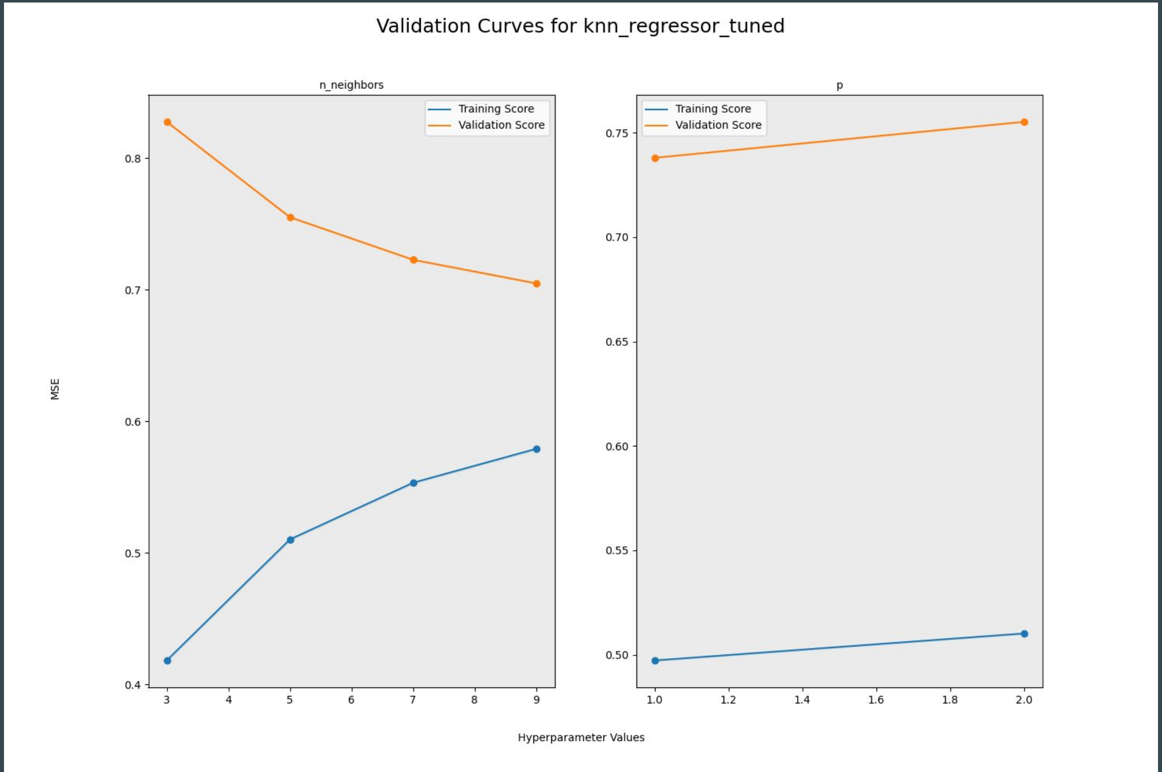


Possible overfitting

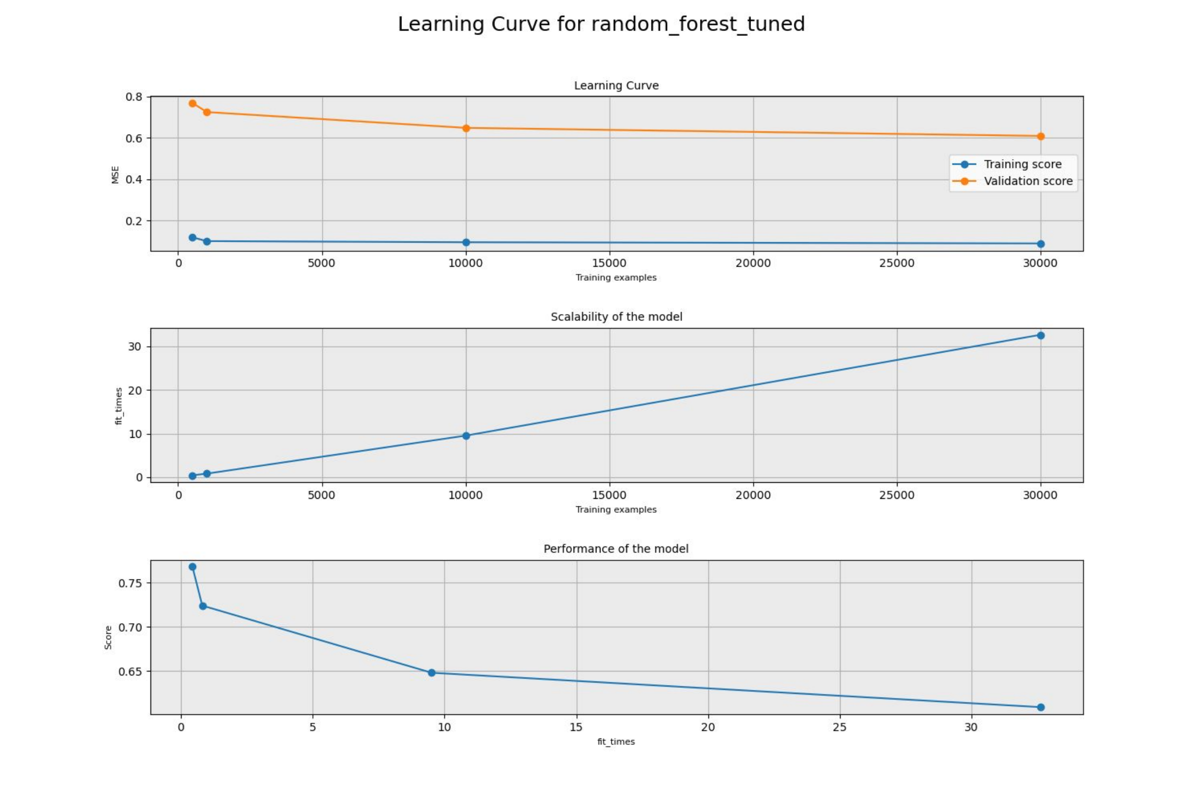
Modeling: Hyperparameter Tuning I



Modeling: Hyperparameter Tuning I



Modeling: Hyperparameter Tuning II



Results: Model Evaluation - Distributions Are Independent?

2 Sample T-Test

Test Type	Test	H0	p-value	Action
Test for normality on Our Model Ratings	Shapiro-Wilk Test	Distribution is normal	4.75e-40	Reject H0
Test for normality on Random Model Ratings	Shapiro-Wilk Test	Distribution is normal	0	Reject H0
Test for equal variance	Bartlett Test	Variances are equal	0	Reject H0

Results: Model Evaluation - Distributions Are Independent?

~~2 Sample T-Test~~

Mann Whitney U Test

Test	H0	p-value	Action
Mann-Whitney U Test	The 2 medians of the distributions are the same	3.80e-29	Reject H0

IMDb License