The Random Forest Model for Multi-Analytic Prediction of Live-Streamer Rank

Introduction

With the rise of artificial intelligence as a research field in the 1990's, there has been exponential growth in the fields of machine learning and deep learning, giving us models that can do anything from beat a Grandmaster (GM) chess player to drive cars. But what exactly is artificial intelligence, machine learning, and deep learning and how are they related?

John McCarthy defines artificial intelligence as "the science and engineering of making intelligent machines, especially intelligent computer programs" (p. 2, 2004). This definition's broadness is indicative of how wide the umbrella of artificial intelligence reaches. Systems of artificial intelligence can be anything from a simple program—if the green button is pressed, dispense soda—to more nuanced applications like predicting the weather from previous years' weather data.

Under artificial intelligence, however, the term machine learning refers to the creation, by humans, of fixed system that can automatically learn from the environment—data—and make better decisions, using statistical techniques (Wu, 2019). The term "learn" as used throughout this paper concerns a model's ability to be able to make predictions and generate new information based on previous data fed into the model's algorithm. Machine learning's difference from artificial intelligence lies in the ability for models to learn from a dataset provided by a human by themselves. Lastly, under machine learning, deep learning encompasses models that learn by themselves using a brain-like neuron-inspired architecture.

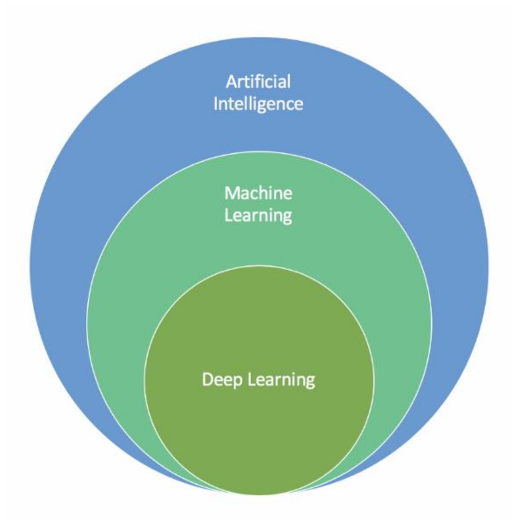


Figure 1. Artificial intelligence to deep learning (Wu, 2019).

The decision tree is one of the most basic forms of machine learning. Given data, the decision tree is able to come to a conclusion or prediction about what is being observed.

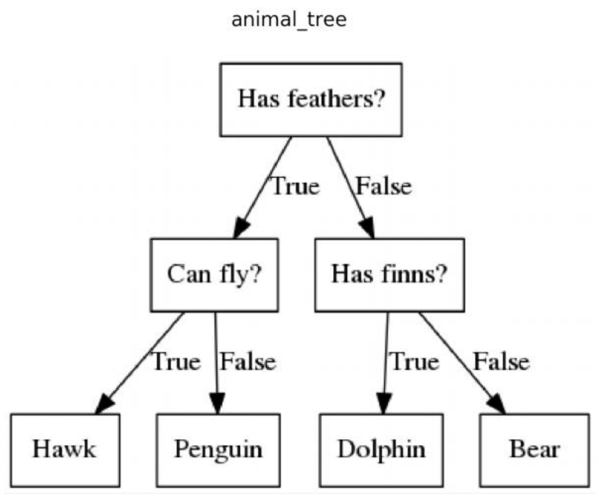


Figure 2. Animal classification decision tree (Müller, 2020)

In the context of classifying animals shown by Figure 2, this decision tree creates splits first on the variable ‘Has feathers?’ then on the variables ‘Can fly?’ and ‘Has fins?’ respectively. The ability for a variable like ‘Can fly?’ to accurately predict what an animal is referred to is the variable’s information gain. Often, variables may have no bearing on the segregation of data; for instance, a variable ‘has a heart’ would apply to all the concerning animals and thus would not help us reach a conclusion at all. Moreover, it is highly unlikely one variable will be able to completely classify some input data, often leading to some error in the decision tree’s prediction. Decision trees can be created automatically given certain variables by optimizing for splits with highest information gain first. The number of splits a decision tree makes before coming to a decision is referred to as that tree’s depth.

But such a straightforward technique can be made resilient to pattern complexity using the random forest model. The random forest model works by using multiple decisions trees with short depths, whose concerning variables are randomly assigned, and taking a democratic vote of their outputs to come to a single answer. The number of trees included and depth of those trees are called hyperparameters of the random forest model and they can be adjusted from model to model to achieve the best results.

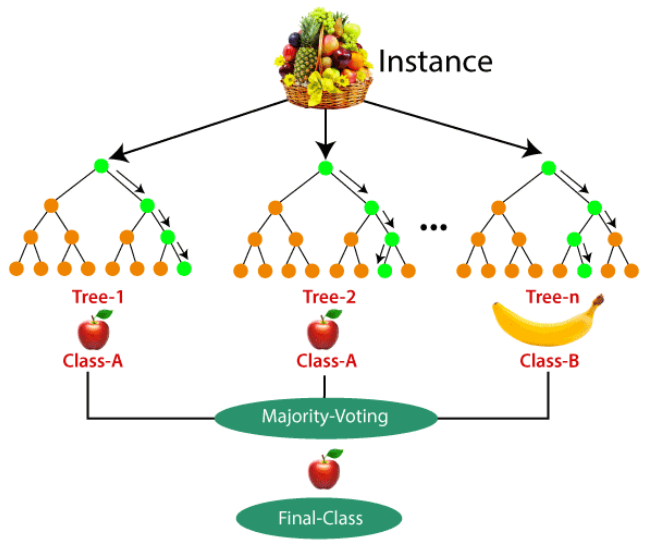


Figure 3. Random forest model with a depth of 3 and tree size of 3 (Mbaabu, 2020)

Live streaming is a relatively new form of media that allows content creation and consumption to happen in real-time, opening the door for synchronous interaction between an entertainer and their viewers (Giertz et al, 2020). One of the most popular live streaming platforms, Twitch.tv, in 2018, had daily traffic of 15 million unique viewers on average in 2018 (Twitch Tracker, 2022). Such a platform is a pool of data ready to be explored using machine learning, but to this point has not had an extensive amount of research done of the topic. One such application of machine learning could be used for modeling the competitiveness of such an environment as a streamer, leading to my research question: can the random forest model, a machine learning model, learn how to rank twitch streamers effectively?

Literature Review

Narrowing down to my specific application of machine learning is best accomplished through first understanding the aims and reaches of machine learning in research currently.

Machine learning application

The applications of machine learning are vast. Iqbal and other researchers develop that such computational ability has been applied to overcome complex healthcare challenges and that such superintelligence is a good fit for understanding and predicting cancer for example, a multifaceted mutation, which is difficult to classify (Iqbal et al, 2022). In the paper, the authors covered various aspects and current research involving artificial intelligence and its integration within medicine to present its efficacy. The authors related how artificial intelligence is being used in precision oncology, drug discovery, and surgery, doing everything from making decisions about whether to pursue surgery to providing surgeons with real-time clinical decisions based on patient biological parameters and camera feed (Iqbal et al, 2022). Iqbal discussed current research which utilizes the random forest model to predict cancer mortality and the lasting cognitive effects of it on a patient (2022). They concluded that although the standard for human intervention in medicine will continue for a while, there is a current demand for better clinical tools for dealing with cancer; the integration of artificial intelligence will be the catalyst for technological revolution for predictive medicine (Iqbal et al, 2022). Iqbal's research presents a highly optimistic view of machine learning's application to healthcare and promotes it as a key player in healthcare of the future.

Supporting the optimism of machine learning in healthcare presented by Iqbal, Zou creates an effective model for predicting diabetes using machine learning. The authors randomly selected 68994 patients, diabetic and healthy, from a hospital in Luzhou, China. First, Zou and their team identified the most important factors from the dataset that are involved in determining diabetes mellitus using PCA and minimum redundancy maximum relevance feature selection methods, finding 14 different factors most important (2018). From this they implemented the random forest and neural network models on their dataset in order to create a model for predicting diabetes mellitus in Chinese patients, verifying their model using the k-fold validation method (Zou et al, 2018). Overall, the authors find that a best result of 0.8084 or 80.8% prediction accuracy using the random forest model (Zou et al, 2018). Zou and Iqbal's research both convey the effectiveness of machine learning in the clinical setting and present a push towards its integration with artificial intelligence. But the optimism behind machine learning is not shared by all.

High appraisal of machine learning in medical literature has been criticized by other computer scientists in the field. DeMasi and other researchers oppose the optimistic perspectives presented by Iqbal and Zou. DeMasi and other researchers conducted a meta-analysis of current machine learning literature regarding the monitoring of patient well-being. The researchers found that approximately 77% of current literature establishes meaningless comparisons by ignoring patient baseline, which lead to ultimately useless models as roughly 80% of the variability in patient mood can be explained by simply guessing a patient is at their normal baseline (DeMasi et al, 2017). Ultimately, DeMasi and other researchers define a more cautious perspective regarding machine learning's use in medicine, urging other researchers to be more speculative and impartial amidst a machine-learning-medicine craze.

However, outside of the field of medicine, Demirel and other researchers apply predictive analysis machine learning to understanding the stock market. The researchers aimed to predict the opening and closing prices of 42 firms' stocks from the Istanbul Stock Exchange National 100 Index. In their analysis, the authors use nine years of data ranging from January first, 2010 to January first, 2019. Covering a large amount of time improves the authors' neural network models by allowing for more data for machine learning and extends the models application to use for long-term investments. But at the same time, it constricts the practical application of the model for day trading. Overall, the authors found that the Multilayer Perceptron model (MLP) outperformed other models in opening and closing stock price—second was the Long Short Term Memory (LSTM), followed by the Support Vector Machines—predicting the stock prices with no statistical difference from the actual stock prices (Demirel et al, 2022). The authors show how machine learning can be a beneficial tool to supplement stock trader's and economists' predictions and analysis of the stock market. But more broadly, Demirel's research supports the vast implementation of machine learning techniques for predictive analysis, extending the focus on healthcare presented by Iqbal and Zou. But at the same time, Demirel's supportive position on machine learning contests that of DeMasi.

But abstracting the use of machine learning, Mazzone and other researchers show how data even permeates the arts. The authors outlined how art can be created by artificial intelligence using generative adversarial networks (GANs), which consist of two neural networks: one that generates new art and one that makes sure the art is not just noise (Mazzone & Elgammal, 2019). The authors argued that the artist in this case is responsible for the input data, tweaking the model, and culling the output. the creativity and breakthrough process of artists defined by Colin Martindale's psychological theory can be emulated by combining separate art styles and periods in the training data for the GAN model. The researchers cemented the efficacy of the machine learning model in study, researching whether people are able to tell the difference between AI art and manmade art; they found that 75% of time, participants thought the AI art was manmade, even commenting on the intentionality, meaning, and other connotative qualities of the art (Mazzone & Elgammal, 2019). This is compared to the 85% of the time people thought that real expressionist art was manmade (Mazzone & Elgammal, 2019).

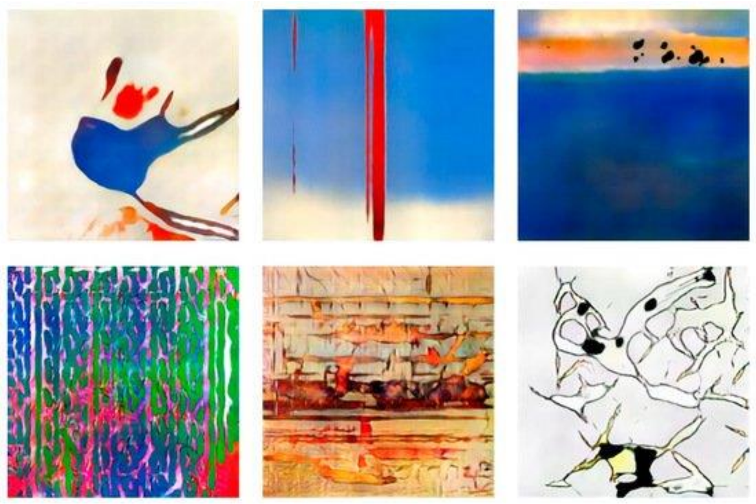


Figure 4. Art generated by neural networks (Mazzone & Elgammal, 2019)

This baseline is important to contextualize how close AI expressionist art resembles manmade art to outside perspectives; though the closeness of the percentages reveal AI art's effectiveness, the apparent difference between AI art and manmade art is still identifiable (). Researchers developed that humans should work with artificial intelligence to create art in order for art to be created from the strengths of both parties (Mazzone & Elgammal, 2019). Mazzone's research strengthens the universality of data in prescribing machine learning to an unconventional scenario and problem. Mazzone shares the optimism highlighted by Zou for the implementation of machine learning but this time relating to more of an expressionistic and fluid application.

In the field of machine learning, widespread optimism is met with smaller but important criticisms. While not all machine learning models perform at a perfect level, their improvement upon previous analytical/ predictive methods continues to inspire further research in an increasing number of fields.

Machine learning on Entertainment and Twitch.tv

Concerning entertainment, machine learning's history has been more brief. Liu and other researchers applied machine learning to understand and predict movie viewer behavior, creating a model that can provide personalized movie recommendations based upon viewing history. The researchers base their data upon the Movie Lens dataset by GroupLens research team from the University of Minnesota in the United States (Liu, 2021). The authors implemented a LSTM deep learning model with specific parameters: 131 neurons, dropout value of 0.2, learning rate of 0.002, sigmoid function as activation function, Adam optimizer, and batch size of 35 (Liu, 2021).

Reaching an accuracy rate of around 95% on a testing data set, Liu's research supports the use of machine learning in entertainment.

Extending the focus on entertainment but narrowing down to live streaming, Kobs and other researchers have used machine learning to understand viewer-streamer interactions on TwitchTv. The researchers seek to explore whether it is possible to predict a user's subscription status to a streamer based on their activity and interaction in the chat of said streamer. The authors held a coding competition/challenge, proving a dataset, for teams to create a model for achieving such a purpose (Kobs et al, 2020). The authors then briefly broke down how each of the four teams made their model; the winner from the challenge implemented a gradient boosted tree machine learning model that did not consider the content of the chat user's chat—the other high performing models had used an LSTM approach (Kobs et al, 2020). Overall, the authors research, showed that there is still room for improvement. In doing a review of a competition, the authors are able to cross compare the methods of each competitor, seeing what aspects of models worked and what didn't. However, in only having a competition with 4 competitors, meaning four models, the authors fail to explore other machine learning models. Unsurprisingly, the platform of Twitch.tv has recently been the focus of data science research, given livestreaming and real-time interaction present a constant waterfall of data.

This focus is extended by Melhart and their team of researchers who similarly aim to capitalize on Twitch.tv's wealth of data. But rather than focusing on viewer subscriptions, Melhart and other researchers apply machine learning to understand moment to moment engagement of the viewership with a videogame. The authors explored whether it is possible to predict exact gameplay viewer engagement (each moment) based on game telemetry. The authors gather data from five streamers, not named, who streamed themselves playing PUBG, an online multiplayer first-person-shooter game, employing artificial neural networks as the predictive model. The authors generalize their model by creating one for four streamers, testing it against the remaining streamers, then repeating this for every combination of the five streamers, preventing overfitting (Melhart et al, 2020). While this may allow the mode to be effectively applied to most PUBG streamers, because each game and streamer on twitch has their own personality and fan-base culture, any further extrapolation of the use of the author's model for other streamers for other games is not justified. Overall, however, the authors find that their model was on average accurate 80% of the time (Melhart et al, 2020).

Gap

Optimism in the field of machine learning has led to recent growth in the research subject, however, there is still much to be explored. Research done by Zou, Demirel, and Mazzone for example contextualize the current pursuits of machine learning in a diversity of topics, from healthcare to art, showcasing both its successes and reservations. More specifically however, research done by Liu highlights machine learning's application to an entertainment platform's analytics to enhance user experience. And this focus on entertainment is further demonstrated by Melhart and Kobs who apply machine learning specifically to data derived from the Twitch.tv livestreaming platform also in pursuit of understanding or improving user experience. However, within the mentioned research, there lies a gap in analyzing content creators: using machine learning to evaluate live- streamers' success.

Research Question

Will the random forest model be able to effectively rank Twitch.tv streamers based on their performance analytics?

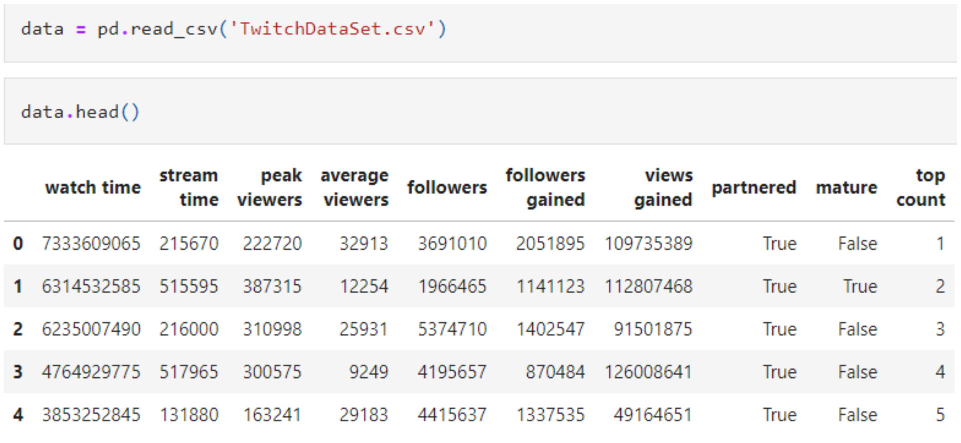
Hypothesis

The random forest model should perform with modest to high precision and accuracy, being able to correctly rank twitch streamers, ±5 spots, with at least 95% accuracy.

Method

Installation

First, a dataset containing a ranking of the top 8800 streamers was sourced via the search data set function on the website www.kaggle.com and downloaded, using the website's download function. Next, the dataset was imported by placing the dataset file in the same directory as the Jupyter notebook and setting a variable 'data' to the file path. The modules *pandas*, *NumPy*, and *matplot.lib*, were installed. While the *pandas* package was used for creating and modifying data frames, *numpy* was responsible for doing linear algebra and *matplot.lib* was used for creating graphs.



```
data = pd.read_csv('TwitchDataSet.csv')
```

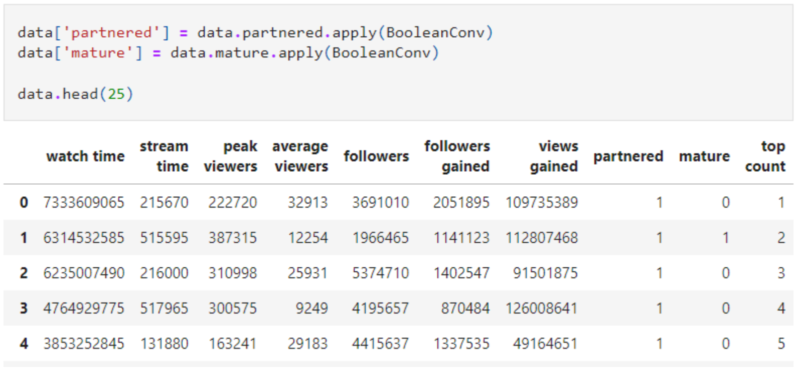
```
data.head()
```

	watch time	stream time	peak viewers	average viewers	followers	followers gained	views gained	partnered	mature	top count
0	7333609065	215670	222720	32913	3691010	2051895	109735389	True	False	1
1	6314532585	515595	387315	12254	1966465	1141123	112807468	True	True	2
2	6235007490	216000	310998	25931	5374710	1402547	91501875	True	False	3
3	4764929775	517965	300575	9249	4195657	870484	126008641	True	False	4
4	3853252845	131880	163241	29183	4415637	1337535	49164651	True	False	5

Figure 5. 4/8800 rows of the dataset

Preparation

In cleaning the dataset, the non-numerical values ‘True’ and ‘False’ on the features ‘partnered’ and ‘mature’ were converted to binary, calling a defined function on the columns ‘partnered’ and ‘mature.’ This was done so that all the features could be processed by the random forest model, as the model only takes numerical data.

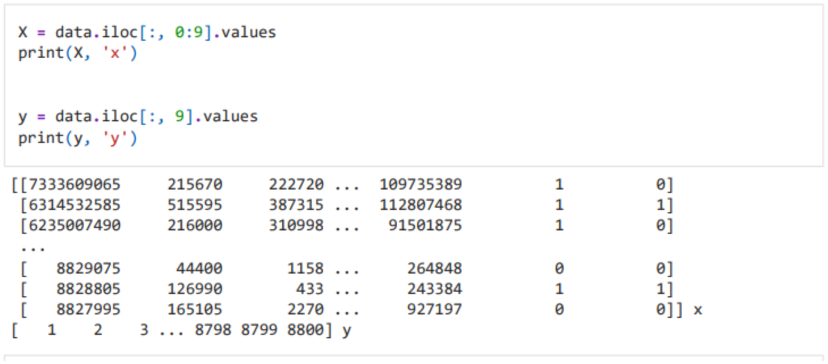


```
data['partnered'] = data.partnered.apply(BooleanConv)
data['mature'] = data.mature.apply(BooleanConv)

data.head(25)
```

	watch time	stream time	peak viewers	average viewers	followers	followers gained	views gained	partnered	mature	top count
0	7333609065	215670	222720	32913	3691010	2051895	109735389	1	0	1
1	6314532585	515595	387315	12254	1966465	1141123	112807468	1	1	2
2	6235007490	216000	310998	25931	5374710	1402547	91501875	1	0	3
3	4764929775	517965	300575	9249	4195657	870484	126008641	1	0	4
4	3853252845	131880	163241	29183	4415637	1337535	49164651	1	0	5

Figure 6. Numericized data

The dataset was then split into the lists X and Y; where Y, streamer rank, the variable we are trying to predict, is an 8800 by 1 array of all the streamers’ rankings and X, the nine variables associated, is a nine by 8800 array of all the streamer analytics.

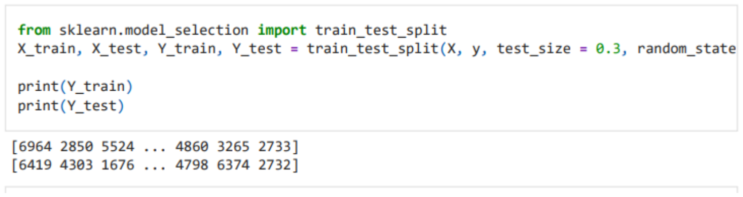


```
X = data.iloc[:, 0:9].values
print(X, 'x')


y = data.iloc[:, 9].values
print(y, 'y')
```

```
[[7333609065     215670     222720 ...  109735389          1          0]
 [6314532585     515595     387315 ...  112807468          1          1]
 [6235007490     216000     310998 ...   91501875          1          0]
 ...
 [   8829075      44400       1158 ...     264848          0          0]
 [   8828805     126990        433 ...     243384          1          1]
 [   8827995     165105       2270 ...     927197          0          0]] x
[   1    2    3 ... 8798 8799 8800] y
```

Figure 7. Separated variables

Next, the *train_test_split* module from *sklearn.model_selection* was imported and applied to both lists X and Y to split the data into the lists X_train, X_test, Y_train, Y_test by randomly assigning rows to each list. The ‘train’ lists are the datasets that the random forest model will train on with both X and Y variables, and the ‘test’ lists are the datasets the random forest model will test on with both X and Y variables. This split of data was done so that our random forest model will be able to display its accuracy after we are done training. Afterall, if the model is trained on the same data that it is tested on, we can expect it to perform exceptionally well. However, in practical applications, we need our model to be able to perform well on new data that our model has not seen before—hence, the ‘test’ lists.



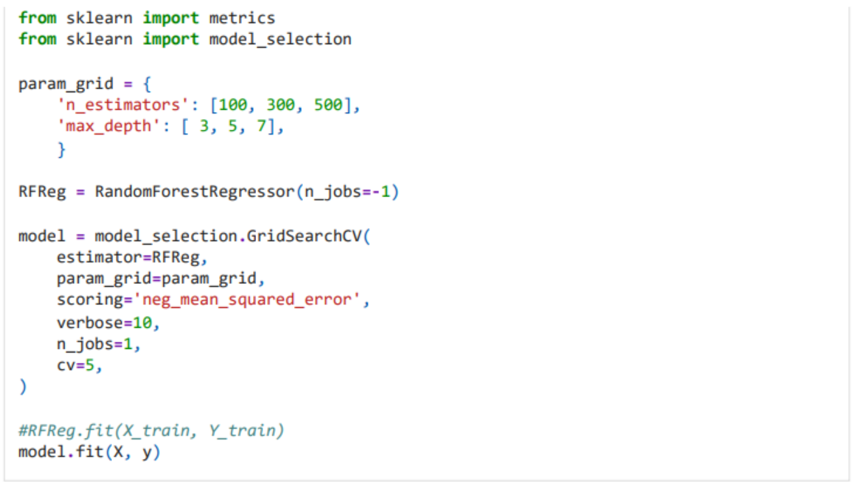
```
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X, y, test_size = 0.3, random_state

print(Y_train)
print(Y_test)
```

```
[6964 2850 5524 ... 4860 3265 2733]
[6419 4303 1676 ... 4798 6374 2732]
```

Figure 8. Randomized Lists

Tuning

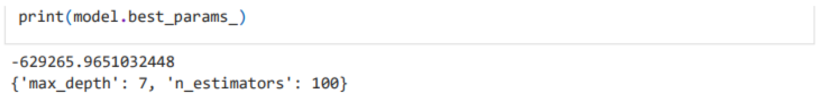
However, before the final model was created, the best hyperparameters needed to be found. This was accomplished through a grid search in which we evaluate the accuracy of every every combination of parameters provided; the hyperparameters that resulted in the best accuracy were recorded. This is done so that the most optimal model, good or bad, can be created in pursuit of answering the research question. The module *RandomForestRegressor* was responsible for creating the random forest model; the *metrics* module was responsible for reporting the best hyperparameters; *model_selection* was responsible for completing the grid search. Creating the random forest with the parameter *n_jobs=-1* makes sure the computation is done by all available processors on the local computer. Running the process on all processors available was done so that the whole search, which involved 45 fits, could be accomplished in a timely manner.



```
from sklearn import metrics
from sklearn import model_selection

param_grid = {
    'n_estimators': [100, 300, 500],
    'max_depth': [ 3, 5, 7],
    }

RFReg = RandomForestRegressor(n_jobs=-1)

model = model_selection.GridSearchCV(
    estimator=RFReg,
    param_grid=param_grid,
    scoring='neg_mean_squared_error',
    verbose=10,
    n_jobs=1,
    cv=5,
)

#RFReg.fit(X_train, Y_train)
model.fit(X, y)
```

Figure 9. Grid Search

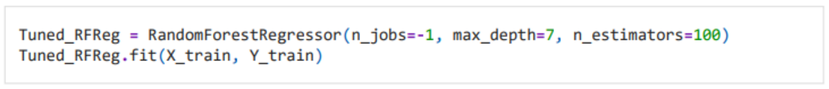
The grid search showed that the best max depth for each tree was 7 and that the most optimal number of trees was 100. These parameters were then passed into a final random forest model that was finally trained on the X_train and Y_train lists.



```
print(model.best_params_)
```

```
-629265.9651032448
{'max_depth': 7, 'n_estimators': 100}
```

Figure 10. Best hyperparameters

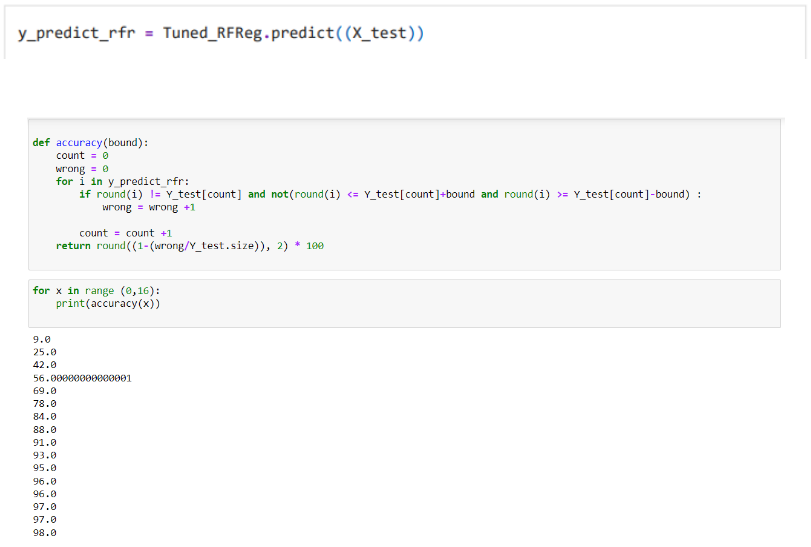


```
Tuned_RFReg = RandomForestRegressor(n_jobs=-1, max_depth=7, n_estimators=100)
Tuned_RFReg.fit(X_train, Y_train)
```

Figure 11. Training the final model

Application

The final, trained model is then applied to the X_test data, producing predictions of streamer rank for each example in the datasets, which are stored in *y_predict_rfr*, and comparing these predictions against their actual values, the accuracy of the model is evaluated at several ranges.



```
y_predict_rfr = Tuned_RFReg.predict((X_test))
```

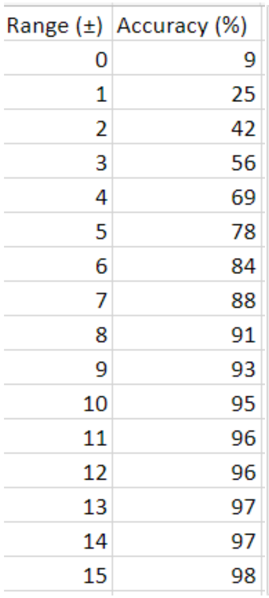
```
def accuracy(bound):
    count = 0
    wrong = 0
    for i in y_predict_rfr:
        if round(i) != Y_test[count] and not(round(i) <= Y_test[count]+bound and round(i) >= Y_test[count]-bound) :
            wrong = wrong +1

        count = count +1
    return round((1-(wrong/Y_test.size)), 2) * 100
```

```
for x in range (0,16):
    print(accuracy(x))
```

```
9.0
25.0
42.0
56.00000000000001
69.0
78.0
84.0
88.0
91.0
93.0
95.0
96.0
96.0
97.0
97.0
98.0
```

Figure 12. Determining model accuracy

Results



Range (±)	Accuracy (%)
0	9
1	25
2	42
3	56
4	69
5	78
6	84
7	88
8	91
9	93
10	95
11	96
12	96
13	97
14	97
15	98

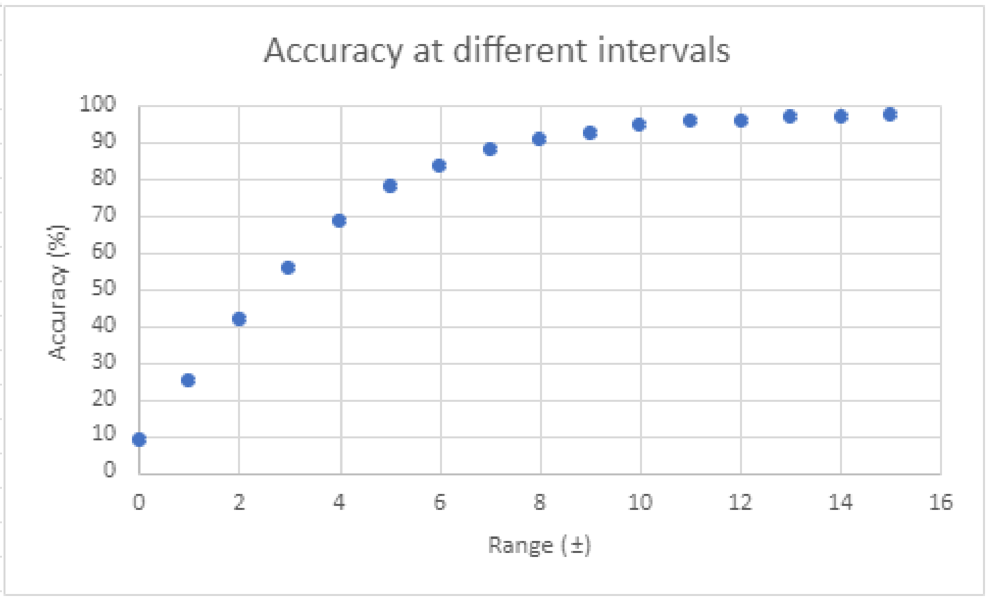
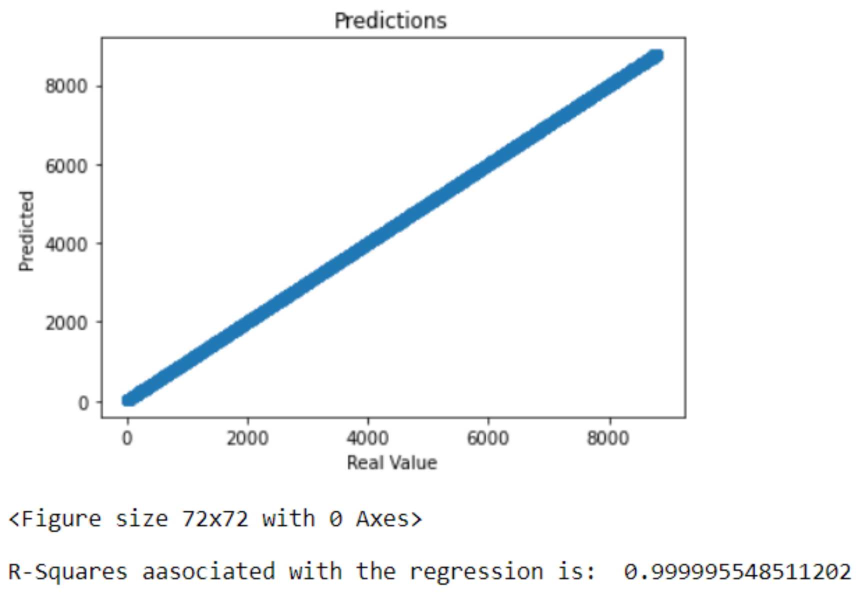


Figure 13. Accuracy of the random forest model, chart and graph, at various ranges 0-15

The random forest model effectively predicts 10% of streamers’ ranks exactly, 25% of streamers’ ranks within ±1 spots, and 91% of streamers’ ranks within ±8 spots. It takes a range of ±11 spots to reach 96% accuracy on the testing data.



```
<Figure size 72x72 with 0 Axes>

R-Squares aasociated with the regression is:  0.999995548511202
```

Figure 14. Predicted values versus real values

Graphing the predicted values against their expected real values we get an exceptionally linear graph where approximately 100% of the variability in the predicted values can be explained by their linear relationship with the real values.

Discussion

Branching off of research such as that done by Konstantin Kobs who analyzes twitch viewer interactions using the gradient boosted tree machine learning model, this paper is concerned with not the viewer but the streamer as well as a completely different machine learning model, covering a gap at the cross between machine learning and live streaming that is the use of the random forest model in the comparison of streamer success.

In seeking to evaluate the efficacy of the random forest model in predicting Twitch streamer rank, it was hypothesized that the random forest model would be able to correctly rank twitch streamers, ±5 ranks, with at least 95% accuracy. However, from the data it is shown that at a range of ±5 ranks, the model was only about 78% accurate. Moreover, it is found that the random forest model accuracy predicts only 9% of streamers’ ranks exactly. This accuracy, however, increases with range.

From this analysis it is evident that the model performs with much less precision than initially hypothesized, but the model’s accuracy is still evident as it reaches the 95% accuracy threshold at a reasonable range of ±10 ranks. While the initial hypothesis is not well-supported, the new understanding derived from the evidence suggests that while the random forest model may not be highly precise in its rank predictions, it is still evidently accurate at obtainable ranges. Looking deeper, in the graph of predicted values versus real values, a uniform correlation between the two variables is shown. In combination with our understanding that the model fails to predict a majority of streamers' ranks exactly, this evidence suggests that at the scale of the whole dataset (8800 streamers) the random forest model is overall reliable; while not useful for differentiating streamers between a few ranks, the model is efficacious at generally placing streamers in the correct quartile—though not the exact position.

Limitations

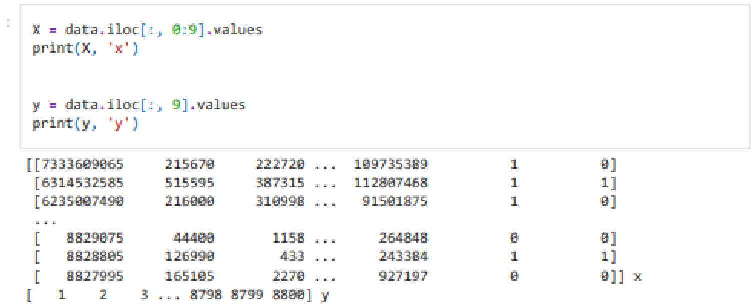
While this research suggests that the random forest model is effective at more generally predicting streamer rank it can only be stated to be so accurate for the range of approximately the top 8800 streamers and lower. This is because the lack of pinpoint accuracy present in the model may be exacerbated at larger intervals, leading to a less accurate model with increased data size. Further research must be applied to better understand this relationship. Moreover, the strength of the random forest model in this situation can directly only be applied when attempting to rank streamers based on the nine present analytics that were used to create the model—watch time, stream time, peak viewers, average viewers, followers, followers gained, views gained, partnered, and mature—or those similar in nature. Since these features are generally considered important performance analytics of a live streamer and are widely used individually to track performance these features can be considered strong, however this limitation is still important to note. Moreover, it is important to understand that the focus of this research was to evaluate the effectiveness of the random forest model in predicting streamer rank not to create the best and most contemporarily applicable model for predicting streamer rank. Meaning, the fact that the dataset was from approximately a year ago—outdated in terms of the actual order of streamer rankings—does not hamper the evidence suggesting that the random forest model is robust in its predictive ability.

Implications

Despite these limitations, the random forest model's strength at predicting in the range of ±10 ranks at the scale of 8800 ranks highlights the random forest model's usefulness for twitch streamers who seek to evaluate their performance relative to their peers in pursuit of improving. With the creation of a live tool utilizing the random forest model that continuously or routinely updated its information about streamers' analytics, streamer would be able to evaluate their performance on multiple analytics relative to their peers through one discreet number. This would be a significant addition to current tools such as www.twitchmetrics.net, which rank streamers but only along one analytic such as viewer hours.

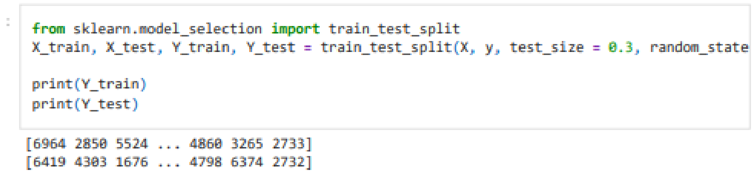
In terms of the impact this research has on the field, the accuracy of the random forest model overall, supports the current optimism in pushing machine learning into various fields, especially entertainment and live streaming. This is a point of view currently supported by research done by Liu, Kobs, and Melhart. Furthermore, the lack of precision present in the current model points current researchers towards looking at other machine learning and possibly deep learning methods in order to improve upon this research.

Appendix



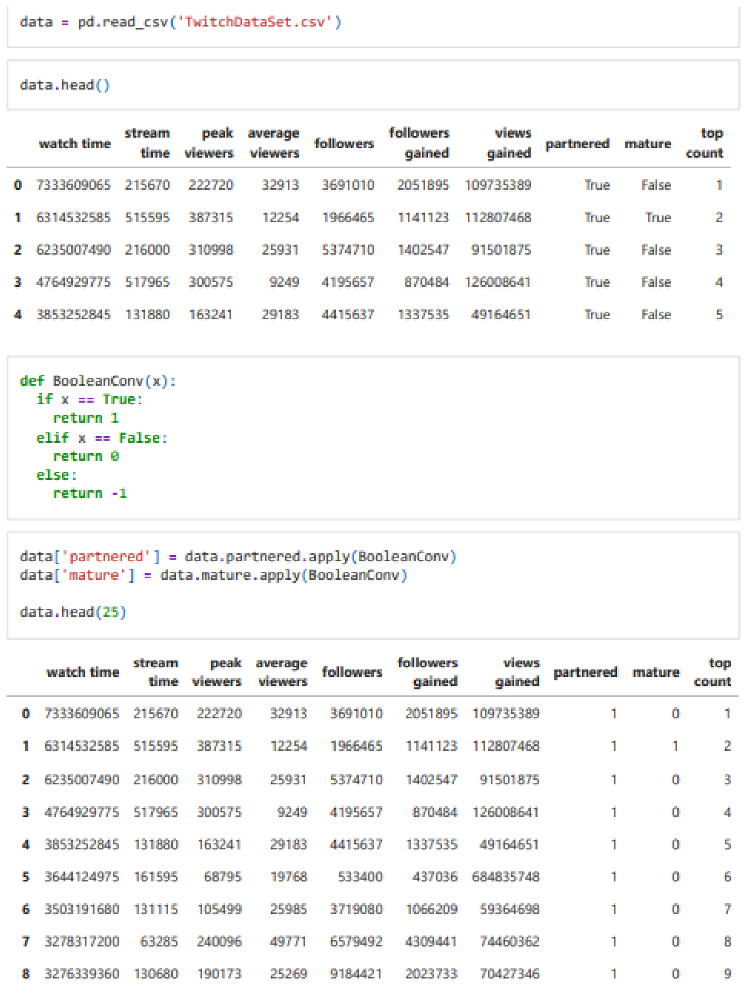
```
X = data.iloc[:, 0:9].values
print(X, 'x')


y = data.iloc[:, 9].values
print(y, 'y')
```

```
[[7333609065     215670     222720 ...  109735389          1          0]
 [6314532585     515595     387315 ...  112807468          1          1]
 [6235007490     216000     310998 ...   91501875          1          0]
 ...
 [   8829075      44400       1158 ...     264848          0          0]
 [   8828805     126990        433 ...     243384          1          1]
 [   8827995     165105       2270 ...     927197          0          0]] x
[   1    2    3 ... 8798 8799 8800] y
```



```
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X, y, test_size = 0.3, random_state

print(Y_train)
print(Y_test)
```

```
[6964 2850 5524 ... 4860 3265 2733]
[6419 4303 1676 ... 4798 6374 2732]
```



```
data = pd.read_csv('TwitchDataSet.csv')
```

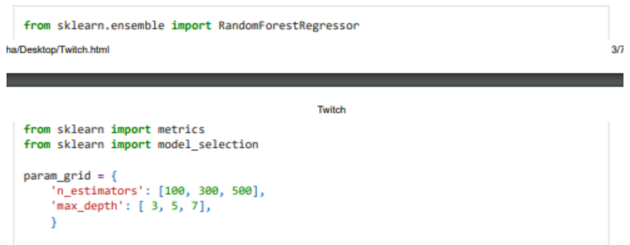
```
data.head()
```

	watch time	stream time	peak viewers	average viewers	followers	followers gained	views gained	partnered	mature	top count
0	7333609065	215670	222720	32913	3691010	2051895	109735389	True	False	1
1	6314532585	515595	387315	12254	1966465	1141123	112807468	True	True	2
2	6235007490	216000	310998	25931	5374710	1402547	91501875	True	False	3
3	4764929775	517965	300575	9249	4195657	870484	126008641	True	False	4
4	3853252845	131880	163241	29183	4415637	1337535	49164651	True	False	5

```
def BooleanConv(x):
  if x == True:
    return 1
  elif x == False:
    return 0
  else:
    return -1
```

```
data['partnered'] = data.partnered.apply(BooleanConv)
data['mature'] = data.mature.apply(BooleanConv)

data.head(25)
```

	watch time	stream time	peak viewers	average viewers	followers	followers gained	views gained	partnered	mature	top count
0	7333609065	215670	222720	32913	3691010	2051895	109735389	1	0	1
1	6314532585	515595	387315	12254	1966465	1141123	112807468	1	1	2
2	6235007490	216000	310998	25931	5374710	1402547	91501875	1	0	3
3	4764929775	517965	300575	9249	4195657	870484	126008641	1	0	4
4	3853252845	131880	163241	29183	4415637	1337535	49164651	1	0	5
5	3644124975	161595	68795	19768	533400	437036	684835748	1	0	6
6	3503191680	131115	105499	25985	3719080	1066209	59364698	1	0	7
7	3278317200	63285	240096	49771	6579492	4309441	74460362	1	0	8
8	3276339360	130680	190173	25269	9184421	2023733	70427346	1	0	9

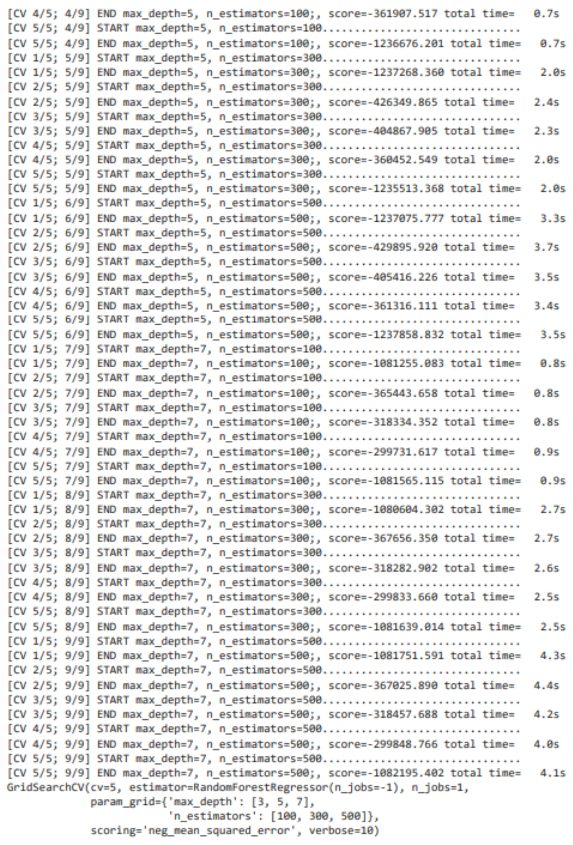


```
from sklearn.ensemble import RandomForestRegressor
```

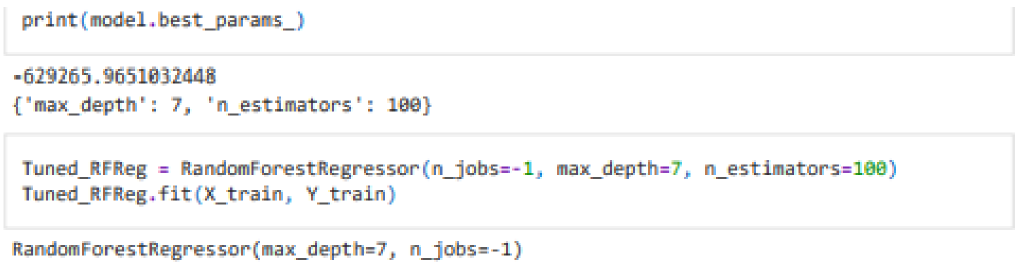
ha/Desktop/Twitch.html 3/7

Twitch

```
from sklearn import metrics
from sklearn import model_selection

param_grid = {
    'n_estimators': [100, 300, 500],
    'max_depth': [ 3, 5, 7],
    }
```



```
[CV 4/5; 4/9] END max_depth=5, n_estimators=100;, score=-361907.517 total time=   0.7s
[CV 5/5; 4/9] START max_depth=5, n_estimators=100...............................
[CV 5/5; 4/9] END max_depth=5, n_estimators=100;, score=-1236676.201 total time=   0.7s
[CV 1/5; 5/9] START max_depth=5, n_estimators=300...............................
[CV 1/5; 5/9] END max_depth=5, n_estimators=300;, score=-1237268.360 total time=   2.0s
[CV 2/5; 5/9] START max_depth=5, n_estimators=300...............................
[CV 2/5; 5/9] END max_depth=5, n_estimators=300;, score=-426349.865 total time=   2.4s
[CV 3/5; 5/9] START max_depth=5, n_estimators=300...............................
[CV 3/5; 5/9] END max_depth=5, n_estimators=300;, score=-404867.905 total time=   2.3s
[CV 4/5; 5/9] START max_depth=5, n_estimators=300...............................
[CV 4/5; 5/9] END max_depth=5, n_estimators=300;, score=-360452.549 total time=   2.0s
[CV 5/5; 5/9] START max_depth=5, n_estimators=300...............................
[CV 5/5; 5/9] END max_depth=5, n_estimators=300;, score=-1235513.368 total time=   2.0s
[CV 1/5; 6/9] START max_depth=5, n_estimators=500...............................
[CV 1/5; 6/9] END max_depth=5, n_estimators=500;, score=-1237075.777 total time=   3.3s
[CV 2/5; 6/9] START max_depth=5, n_estimators=500...............................
[CV 2/5; 6/9] END max_depth=5, n_estimators=500;, score=-429895.920 total time=   3.7s
[CV 3/5; 6/9] START max_depth=5, n_estimators=500...............................
[CV 3/5; 6/9] END max_depth=5, n_estimators=500;, score=-405416.226 total time=   3.5s
[CV 4/5; 6/9] START max_depth=5, n_estimators=500...............................
[CV 4/5; 6/9] END max_depth=5, n_estimators=500;, score=-361316.111 total time=   3.4s
[CV 5/5; 6/9] START max_depth=5, n_estimators=500...............................
[CV 5/5; 6/9] END max_depth=5, n_estimators=500;, score=-1237858.832 total time=   3.5s
[CV 1/5; 7/9] START max_depth=7, n_estimators=100...............................
[CV 1/5; 7/9] END max_depth=7, n_estimators=100;, score=-1081255.083 total time=   0.8s
[CV 2/5; 7/9] START max_depth=7, n_estimators=100...............................
[CV 2/5; 7/9] END max_depth=7, n_estimators=100;, score=-365443.658 total time=   0.8s
[CV 3/5; 7/9] START max_depth=7, n_estimators=100...............................
[CV 3/5; 7/9] END max_depth=7, n_estimators=100;, score=-318334.352 total time=   0.8s
[CV 4/5; 7/9] START max_depth=7, n_estimators=100...............................
[CV 4/5; 7/9] END max_depth=7, n_estimators=100;, score=-299731.617 total time=   0.9s
[CV 5/5; 7/9] START max_depth=7, n_estimators=100...............................
[CV 5/5; 7/9] END max_depth=7, n_estimators=100;, score=-1081565.115 total time=   0.9s
[CV 1/5; 8/9] START max_depth=7, n_estimators=300...............................
[CV 1/5; 8/9] END max_depth=7, n_estimators=300;, score=-1080604.302 total time=   2.7s
[CV 2/5; 8/9] START max_depth=7, n_estimators=300...............................
[CV 2/5; 8/9] END max_depth=7, n_estimators=300;, score=-367656.350 total time=   2.7s
[CV 3/5; 8/9] START max_depth=7, n_estimators=300...............................
[CV 3/5; 8/9] END max_depth=7, n_estimators=300;, score=-318282.902 total time=   2.6s
[CV 4/5; 8/9] START max_depth=7, n_estimators=300...............................
[CV 4/5; 8/9] END max_depth=7, n_estimators=300;, score=-299833.660 total time=   2.5s
[CV 5/5; 8/9] START max_depth=7, n_estimators=300...............................
[CV 5/5; 8/9] END max_depth=7, n_estimators=300;, score=-1081639.014 total time=   2.5s
[CV 1/5; 9/9] START max_depth=7, n_estimators=500...............................
[CV 1/5; 9/9] END max_depth=7, n_estimators=500;, score=-1081751.591 total time=   4.3s
[CV 2/5; 9/9] START max_depth=7, n_estimators=500...............................
[CV 2/5; 9/9] END max_depth=7, n_estimators=500;, score=-367025.890 total time=   4.4s
[CV 3/5; 9/9] START max_depth=7, n_estimators=500...............................
[CV 3/5; 9/9] END max_depth=7, n_estimators=500;, score=-318457.688 total time=   4.2s
[CV 4/5; 9/9] START max_depth=7, n_estimators=500...............................
[CV 4/5; 9/9] END max_depth=7, n_estimators=500;, score=-299848.766 total time=   4.0s
[CV 5/5; 9/9] START max_depth=7, n_estimators=500...............................
[CV 5/5; 9/9] END max_depth=7, n_estimators=500;, score=-1082195.402 total time=   4.1s
GridSearchCV(cv=5, estimator=RandomForestRegressor(n_jobs=-1), n_jobs=1,
             param_grid={'max_depth': [3, 5, 7],
                         'n_estimators': [100, 300, 500]},
             scoring='neg_mean_squared_error', verbose=10)
```

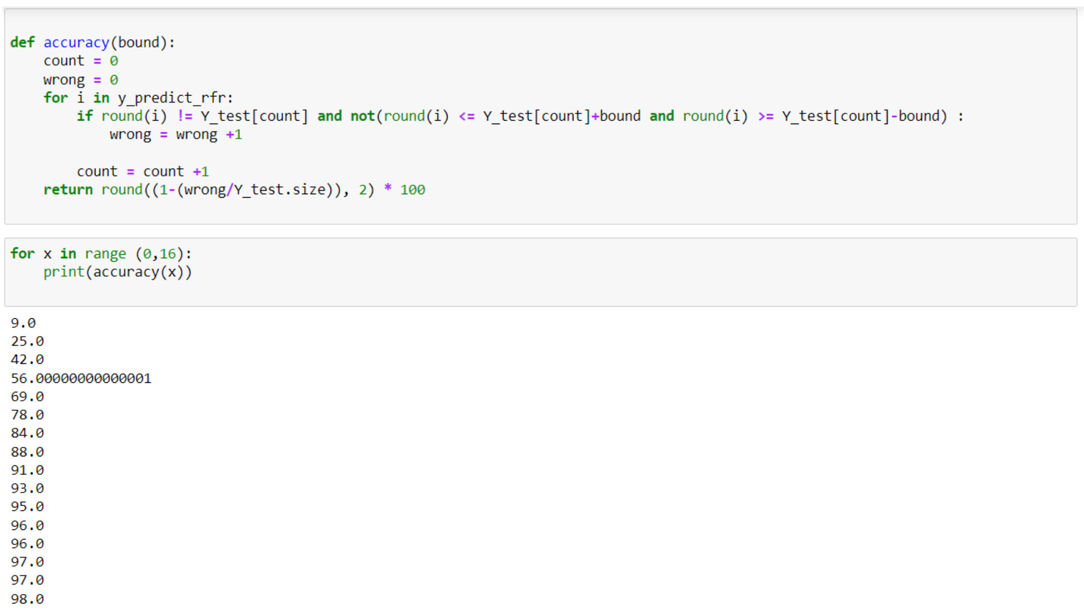


```
print(model.best_params_)
```

```
-629265.9651032448
{'max_depth': 7, 'n_estimators': 100}
```

```
Tuned_RFReg = RandomForestRegressor(n_jobs=-1, max_depth=7, n_estimators=100)
Tuned_RFReg.fit(X_train, Y_train)
```

```
RandomForestRegressor(max_depth=7, n_jobs=-1)
```



```
def accuracy(bound):
    count = 0
    wrong = 0
    for i in y_predict_rfr:
        if round(i) != Y_test[count] and not(round(i) <= Y_test[count]+bound and round(i) >= Y_test[count]-bound) :
            wrong = wrong +1

        count = count +1
    return round((1-(wrong/Y_test.size)), 2) * 100
```

```
for x in range (0,16):
    print(accuracy(x))
```

```
9.0
25.0
42.0
56.00000000000001
69.0
78.0
84.0
88.0
91.0
93.0
95.0
96.0
96.0
97.0
97.0
98.0
```

References

DeMasi O, Kording K, Recht B (2017) Meaningless comparisons lead to false optimism in medical machine learning. PLoS ONE 12(9): e0184604. https://doi.org/10.1371/journal.pone.0184604

Demirel, U. et al (2021). Predicting Stock Prices Using Machine Learning Methods and Deep Learning Algorithms: The Sample of the Istanbul Stock Exchange. Gazi University Journal of Science, 34(1), 63–82. https://doi.org/10.35378/gujs.679103

Giertz, J., Weiger, W. H., Törhönen, M., & Hamari, J. (2020). Understanding the what and how of successful social live streaming. In J. Koivisto, M. Bujić, & J. Hamari (Eds.), GamiFIN Conference 2020: Proceedings of the 4th International GamiFIN Conference (pp. 167-176). (CEUR workshop proceedings; Vol. 2637). CEUR-WS. http://ceur-ws.org/Vol-2637/paper17.pdf

Iqbal, M. J. et al (2021). Clinical applications of artificial intelligence and machine learning in cancer diagnosis: looking into the future. Cancer Cell International, 21(1), 1–11. https://doi.org/10.1186/s12935-021-01981-1

Kobs, K. et al (2020). Towards Predicting the Subscription Status of Twitch.tv Users - ECML-PKDD ChAT Discovery Challenge 2020. ChAT@PKDD/ECML.

Mbaabu, O. (2020). Introduction to random forest in machine learning. Section. Retrieved March 27, 2022, from https://www.section.io/engineering-education/introduction-to-random-forest-in-machine-learning/

Melhart, D. et al (2020). Moment-to-Moment Engagement Prediction through the Eyes of the Observer: PUBG Streaming on Twitch. International Conference on the Foundations of Digital Games. Presented at the Bugibba, Malta. doi:10.1145/3402942.3402958

Müller, Wilmuth & Pallmer, Dirk & Mühlenberg, Dirk & Loumiotis, Ioannis & Remoundou, Konstantina & Kosmides, Pavlos & Demestichas, Konstantinos. (2020). Machine learning for discovery analytics to support criminal investigations. 4. 10.1117/12.2557541.

Liu, J. et al (2021). Personalized Movie Recommendation Method Based on Deep Learning. Mathematical .in Engineering, 1–12. https://doi.org/10.1155/2021/6694237

Mazzone, M., & Elgammal, A. (2019). Art, Creativity, and the Potential of Artificial Intelligence. The Artist and Journal of Home Culture, 8, 26.

Twitch Tracker. (2022, January). Twitch Statistics & Charts. TwitchTracker. Retrieved January 2022, from https://twitchtracker.com/statistics

Wu, J. (2019, July 12). Ai, Machine Learning, deep learning explained simply. Medium. Retrieved January 28, 2022, from https://towardsdatascience.com/ai-machine-learning-deep-learning-explained-simply-7b553da5b960

Zou, Q. et al (2018). Predicting diabetes mellitus with machine learning techniques. Frontiers in Genetics, 9. https://doi.org/10.3389/fgene.2018.00515