RANDOM INTENSIFICATION of HURRICANES

WHEN THE WIND SPEED OF A HURRICANE **INCREASES**

BY OVER 35 MPH WITHIN A 24 HOUR TIME PERIOD

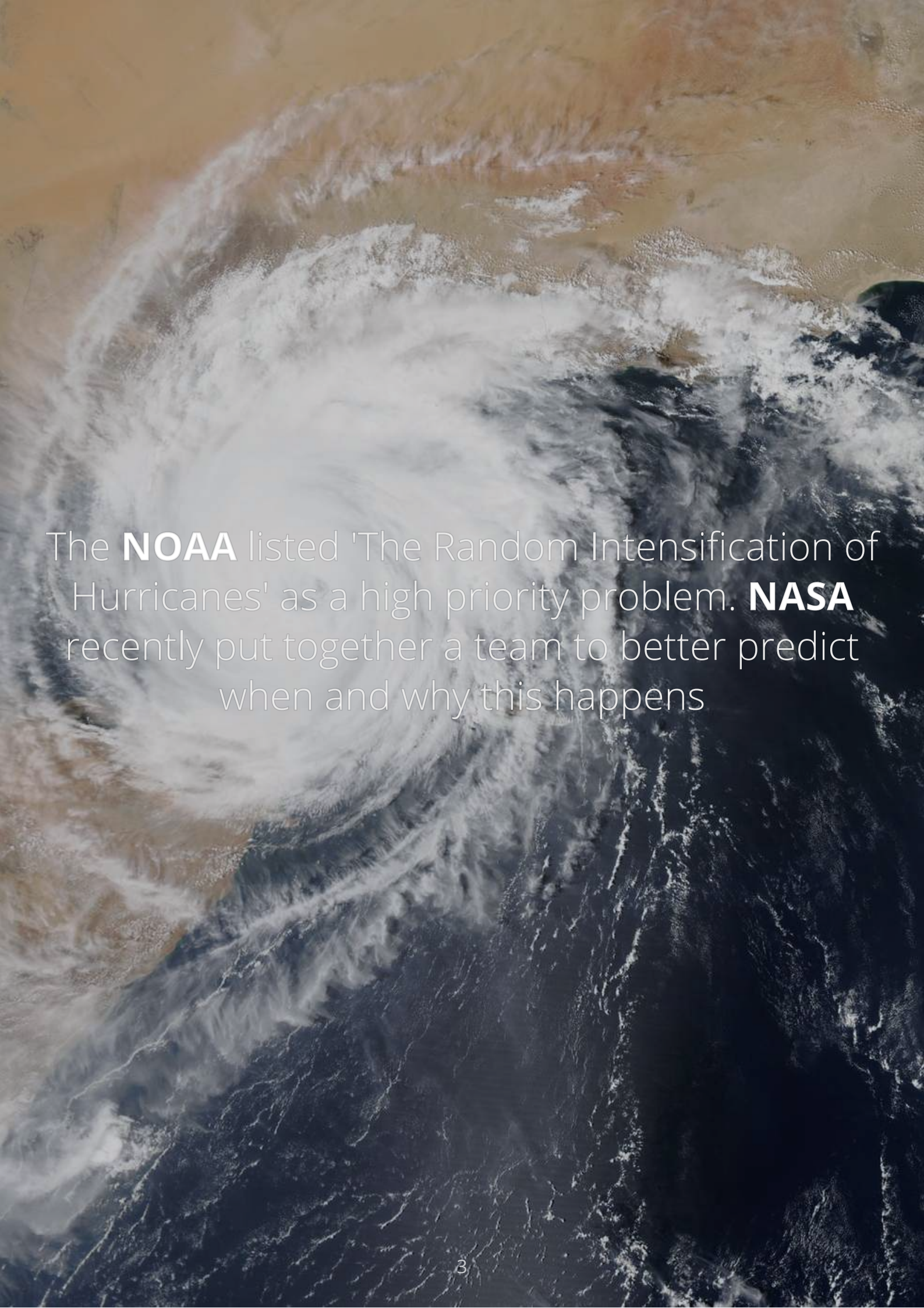
MACHINE **LEARNING**

Hurricanes are randomly intensifying and we are getting closer to understanding why

The weather is going to take time to fully understand. With the help of machine learning we are beginning to speed up the process. There are so many different aspects to a storm. It was interesting to find out that wind speed is the only variable used in the classification for hurricanes. A category 1 wind speed is '74 - 95mph' and a category 5 is 'over 156' mph. The wind speed of 190 mph was previously thought as the maximum theoretical wind speed possible on earth. Then we had typhoon 'Haiyan' that had wind speeds come in at 195 mph. Does that mean its time to add a category 6 classification of hurricane?

It is devastating to communities when wind speed increases by 35 mph within a 24 hour time period. This happened to Hurricane Maria in 2017 when it increased from a category 1 to a category 5 within 24 hours and not a single computer model predicted it. Then we had Hurricane Dorian which was predicted to become a tropical storm right before it intensified into a category 5 hurricane that caused massive damage in the Bahamas.

We are lucky to be living in a time period where accurately predicting future wind speeds is becoming possible.



The **NOAA** listed 'The Random Intensification of Hurricanes' as a high priority problem. **NASA** recently put together a team to better predict when and why this happens

ALGORITHM

Predicting the path of a hurricane has improved significantly over the years. Lets turn our energy towards predicting wind speed and see how quickly we can progress.

GETTING STARTED:

- Download & preprocess the SHIPS(Statistical Hurricane Intensity Prediction Scheme) dataset.
- Train the Random Forest Classifier to determine when a hurricane's wind speed will increase by 35 mph or more.
- Adjust the Hyperparameters of the Classifier in order to maximize results.
- Test the trained program on new data and see how it performs.
- Refine the algorithm

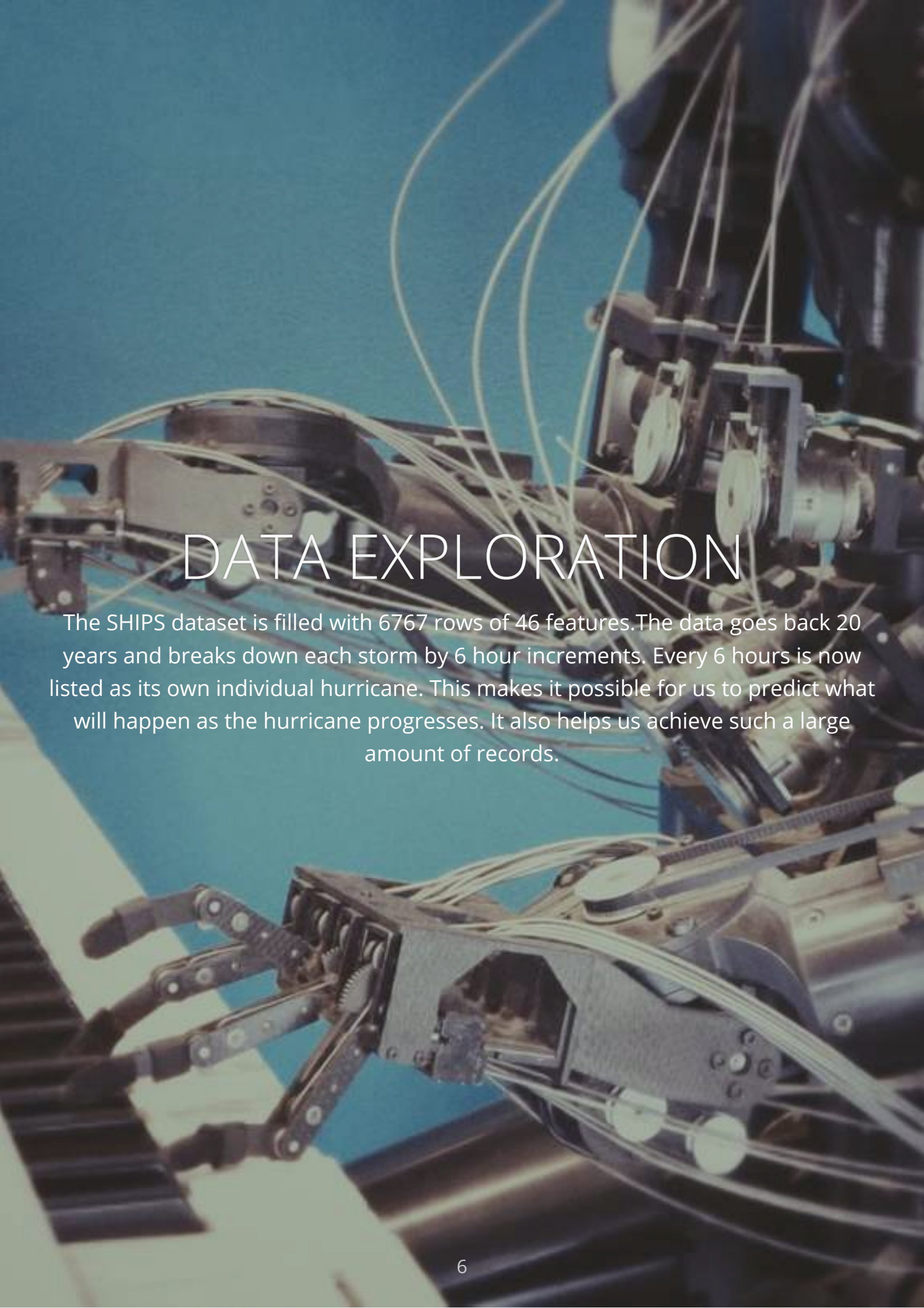
METRICS: NASA and NOAA use PSS score to better calculate the accuracy.

2x2 Contingency Table		Event Observed	
		Yes	No
Event Forecast	Yes	a (hits)	b (false alarms)
	No	c (misses)	d (correct negatives)

Probability Of Detection (POD) POD = a/(a+c). Range: 0 to 1. Perfect score: 1.

False Alarm Ratio (FAR) FAR = b/(a+b). Range: 0 to 1. Perfect score: 0.

Peirce Skill Score (PSS) $PSS = \frac{(ad - bc)}{(a + c)\times(b + d)}$ Range: -1 to 1. Perfect score: 1.



DATA EXPLORATION

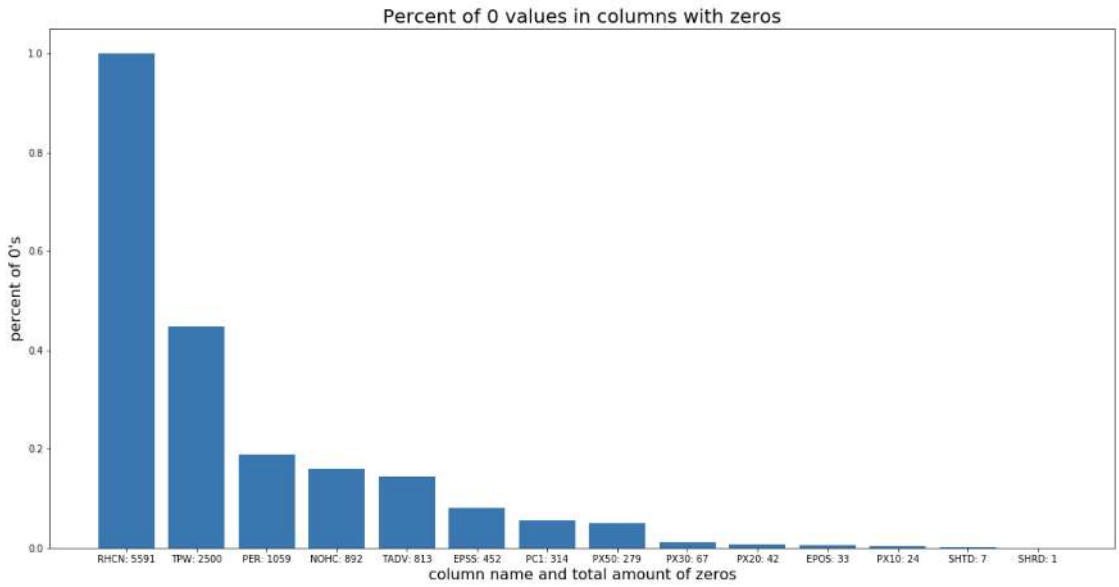
The SHIPS dataset is filled with 6767 rows of 46 features.The data goes back 20 years and breaks down each storm by 6 hour increments. Every 6 hours is now listed as its own individual hurricane. This makes it possible for us to predict what will happen as the hurricane progresses. It also helps us achieve such a large amount of records.

RANDOM **FOREST**

WON'T WORK WITH NULL VALUES OR ANY CATEGORICAL DATA

Lets take a look at all of the 0.0 values within the data. We can keep this data in as 0.0 and it will technically still run... But it will effect the performance of the Algorithm as it is based on decision trees.

Lets look at columns with the most 0's in it below:



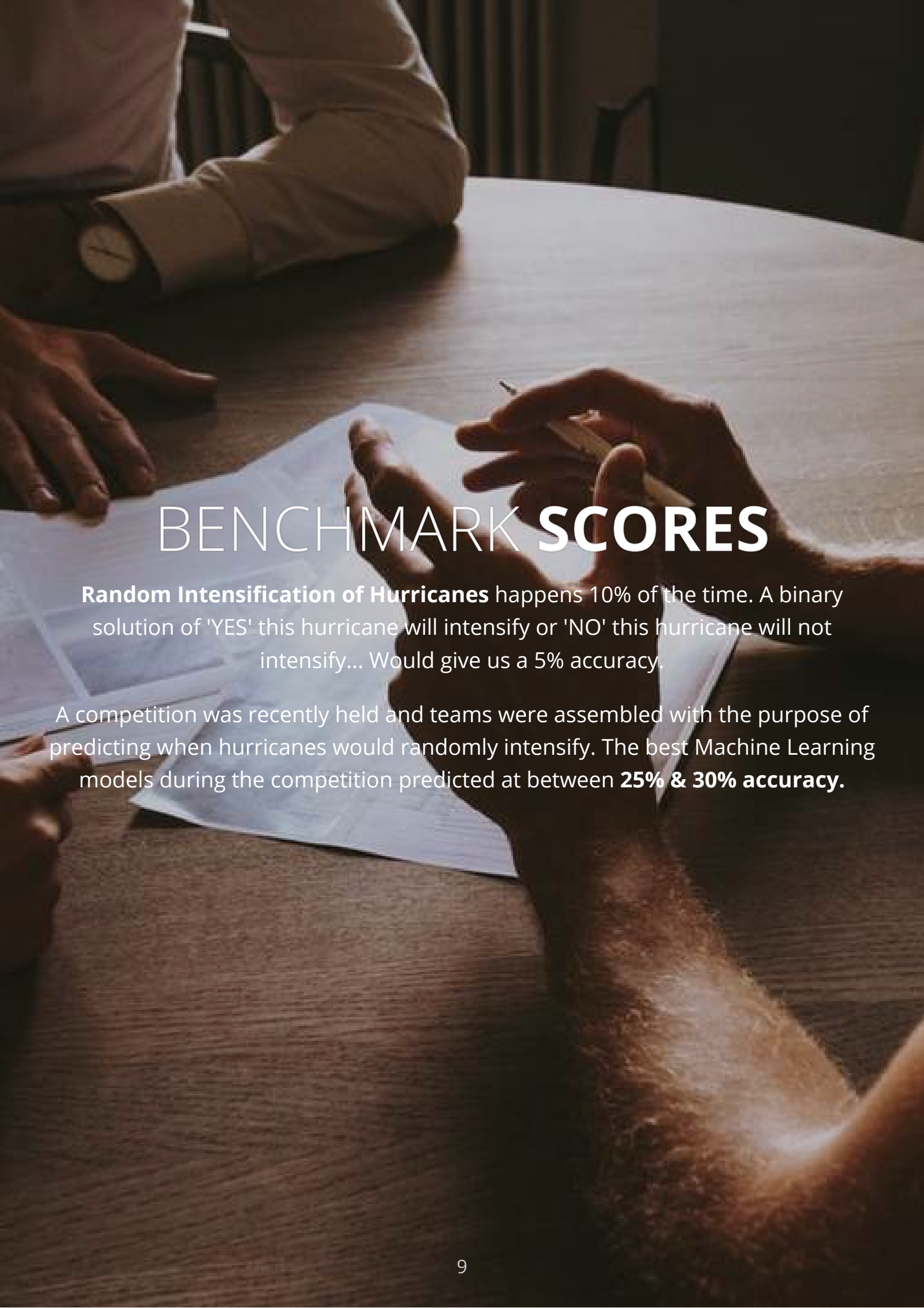
Quick observation will easily encourage us to eliminate RHCN and TPW as they have far too many zeros.

PER is something that has a lot of zeros as well. It is the measurement of 12 hour change in intensity though. Sometimes there will be a 0 in intensity change. For that reason it will stay put.

We also will get rid of both the NOHC & TADV columns as they have too many zeros and we need to keep enough rows to train our program.

The final training dataset will be 4626 records and 42 features. With the help of Bootstrapping or Bagging this amount of records will be a sufficient amount for future machine learning models.

Random Forest is a beast of a program, a math bomb of epic proportions. It is a successful machine learning algorithm because it provides a good predictive performance, low overfitting, and easy interpretability. It is a tree based model that doesn't require us to scale all of the features and we are able to set the number of decision trees within the forest.



BENCHMARK **SCORES**

Random Intensification of Hurricanes happens 10% of the time. A binary solution of 'YES' this hurricane will intensify or 'NO' this hurricane will not intensify... Would give us a 5% accuracy.

A competition was recently held and teams were assembled with the purpose of predicting when hurricanes would randomly intensify. The best Machine Learning models during the competition predicted at between **25% & 30% accuracy.**

TRAINING **THE ALGORITHM**

We need to train_test_split the data and sklearn makes this an easily achievable process. After we call the function train_test_split, it is quite simple to call RandomForestClassifier().fit(X_train, y_train) and our program will officially take off in a massive decision tree of our wildest imaginations.

After we run the program we are able to achieve scores of:

- ~16.8% PSS(Pierces Skill Score)
- ~ 52.7% FAR(False Alarm Ratio)
- ~18.8% POD(Probability of Detection)

These numbers are quite a bit below the number we know is possible. We know that a 25% PSS score is something that others have achieved with the same SHIPS dataset. The fact that we were able to get relatively close to that score without having adjusted the hyperparameters is a testament to the power of The Random Forest.

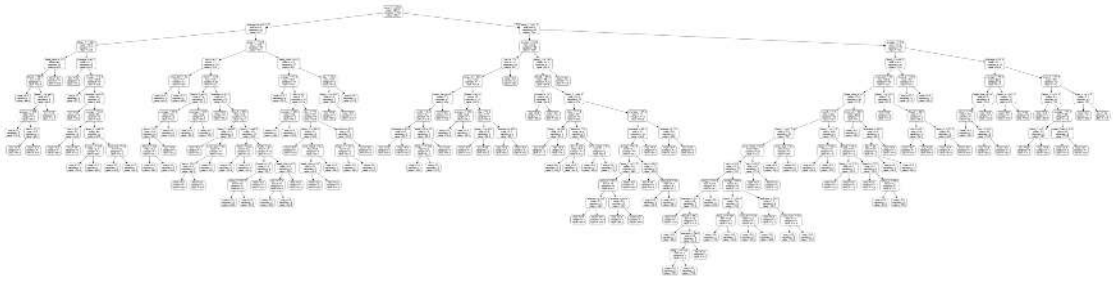
We are going to need to adjust the hyperparameters in order to achieve a higher score.

HYPERPARAMETER **ADJUSTMENTS**

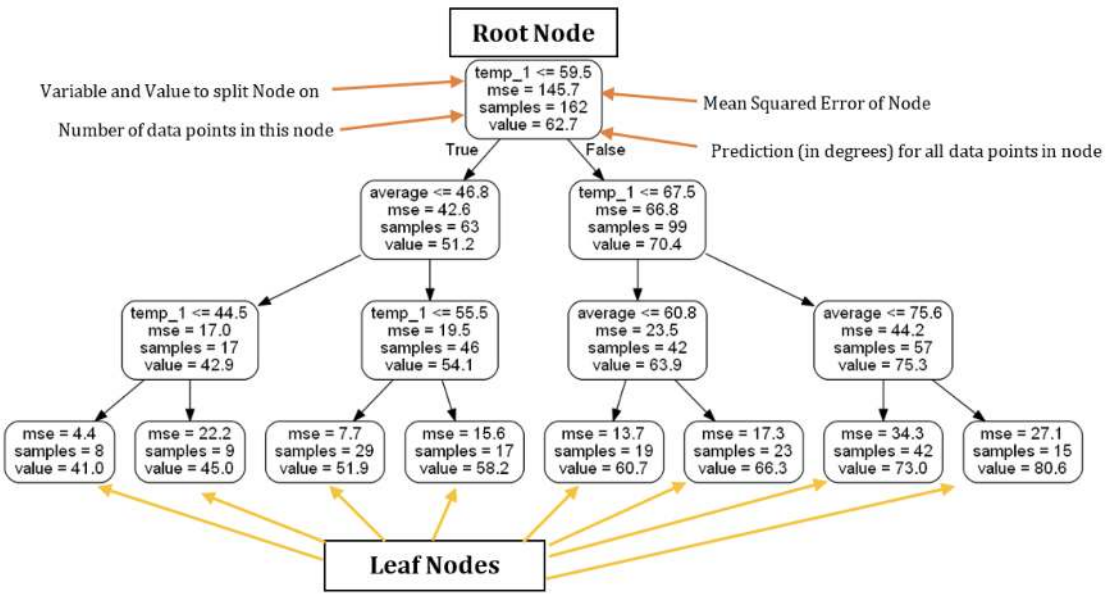
- n_estimators = 2000
- max_depth = 304
- max_features = 'sqrt'
- min_samples_leaf = 1
- min_samples_split = 2
- warm_start = 'True'
- criterion = 'gini'

I can only imagine how tedious it must have been to tune hyperparameters in the past. The RandomizedSearchCV() has made things exponentially easier.

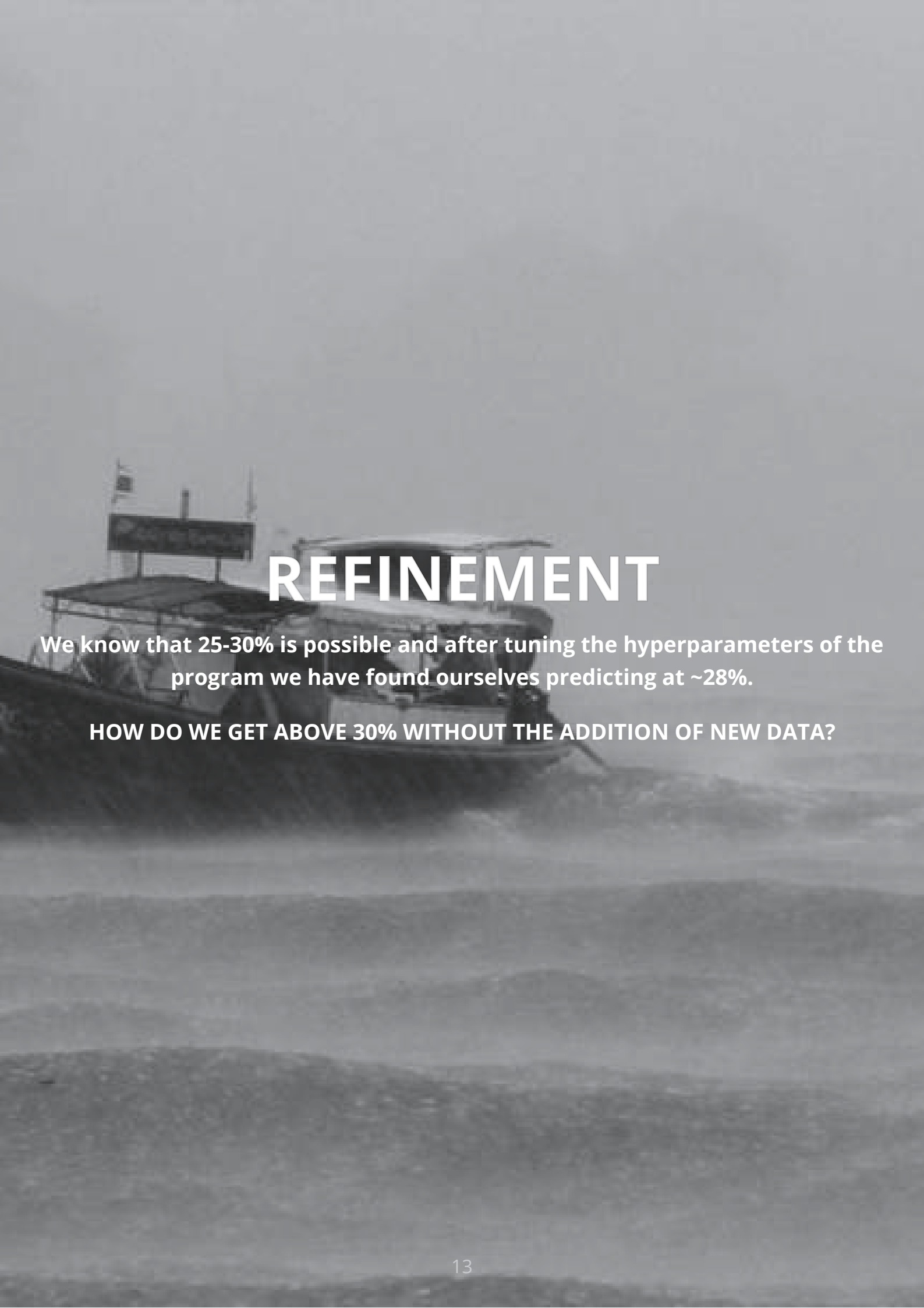
It is quite intricate what is happening within the random forest. Below is a better visualization of the mathematical forest that is being created within a program.



Single Full Decision Tree in Forest



THE ABOVE IMAGES ARE AN EXAMPLE OF RANDOM FOREST TREES AND NOT OF THE RANDOM FOREST WE CREATED. WE WOULD BE LOOKING AT 2000 FORESTS AND THAT IS TOO MANY TO SHOWCASE. :)



REFINEMENT

We know that 25-30% is possible and after tuning the hyperparameters of the program we have found ourselves predicting at ~28%.

HOW DO WE GET ABOVE 30% WITHOUT THE ADDITION OF NEW DATA?

FEATURE EXTRACTION

We are in a situation where there are 2 separate possibilities.

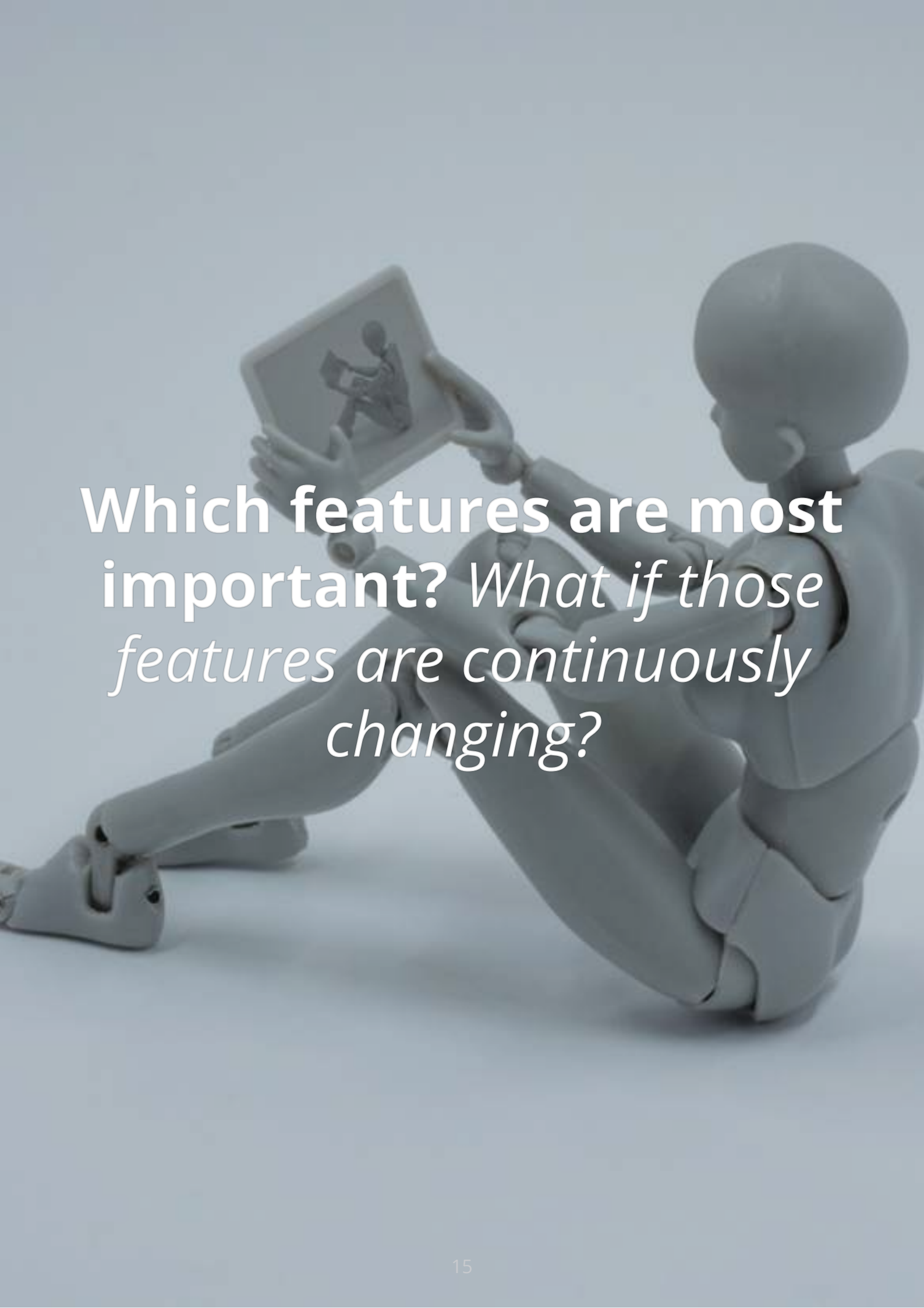
- 1) There is not enough data:
- 2) The algorithm isn't learning in the most efficient way

Lets assume there to be sufficient data to achieve a higher with a few adjustments.

Lets use feature extraction throughout this program and see which features are most important in determining the outcome.

The issue we keep running into though is that the features keep varying in importance. The top 10 most important features consistently change throughout each programs iteration. So which features do we need to keep? So far... The PSS score hasn't been able to achieve above ~28%.

It seems the program is having issues determining which features are most important. We can presume that the bootstrap method within Random Forest has something to do with this. The program uses duplicate data when the bootstrapping parameter is turned on.


Which features are most important? *What if those features are continuously changing?*

BEST PERFORMING FEATURES:

Lets create a function to find the top performing feature within the dataset.

We have a problem where the top features are consistently changing. So where do we cut off the amount of features to train on? Which features are really the best?

LETS FIND THE TOP PERFORMING COLUMNS IN A DIFFERENT WAY

In sports there is often a players ranking system that is voted on by the players. This ranking system tends to carry the most weight within the sports world as the players are being ranked by other players. In other words... the players are being ranked by the people that play against or with them.

We need the best features. We need to see what features can get to first place. We need the players that are able to get to the top of the program as the most important player.

HOW TO:

- Find the top performing FEATURE/COLUMN/PLAYER in dataframe.
- Take the top performing FEATURE/COLUMN/PLAYER in the dataframe out. Store it in a list as first place. :)
- Run the program again without the first place performer and we will be forced into having a new top performer. Store this new top performer in the same list as above. When appended it will become second place.
- Keep taking away the programs most important feature which will force it to find a new feature to rely on. We can get an understanding of the order of things with this. We will end up with a top 40 features in order of importance.

AFTER RUNNING THE FUNCTION

New feature importance with 1 being most important and 40 being least important:

- 1) 'ENSS'
- 2) 'SHGC'
- 3) 'SHRD'
- 4) 'clon'
- 5) 'SHRG'
- 6) 'SHRD2'
- 7) 'U200'
- 8) 'VS'
- 9) 'MSLP'
- 10) 'RSST'
- 11) 'POT'
- 12) 'TPWC'
- 13) 'vmax'
- 14) 'SHTD'
- 15) 'clat'
- 16) 'UMOV'
- 17) 'PSLV'
- 18) 'T200'
- 19) 'T250'
- 20) 'EPOS'
- 21) 'EPSS'
- 22) 'Z850'
- 23) 'DIVC'
- 24) 'D200'
- 25) 'RHMD'
- 26) 'RHLO'
- 27) 'PER'
- 28) 'DTL'
- 29) 'PC2'

- 30) 'AVBT2'
- 31) 'AVBT'
- 32) 'BTAV'
- 33) 'TBMX'
- 34) 'SDBT2'
- 35) 'SDBT'
- 36) 'PX50'
- 37) 'PX30'
- 38) 'PX10'
- 39) 'PX20'
- 40) 'PC1'

HOW MANY FEATURES SHOULD WE USE?

We now have a nice order to our feature list. But how many features are the correct amount of features to use within this model?

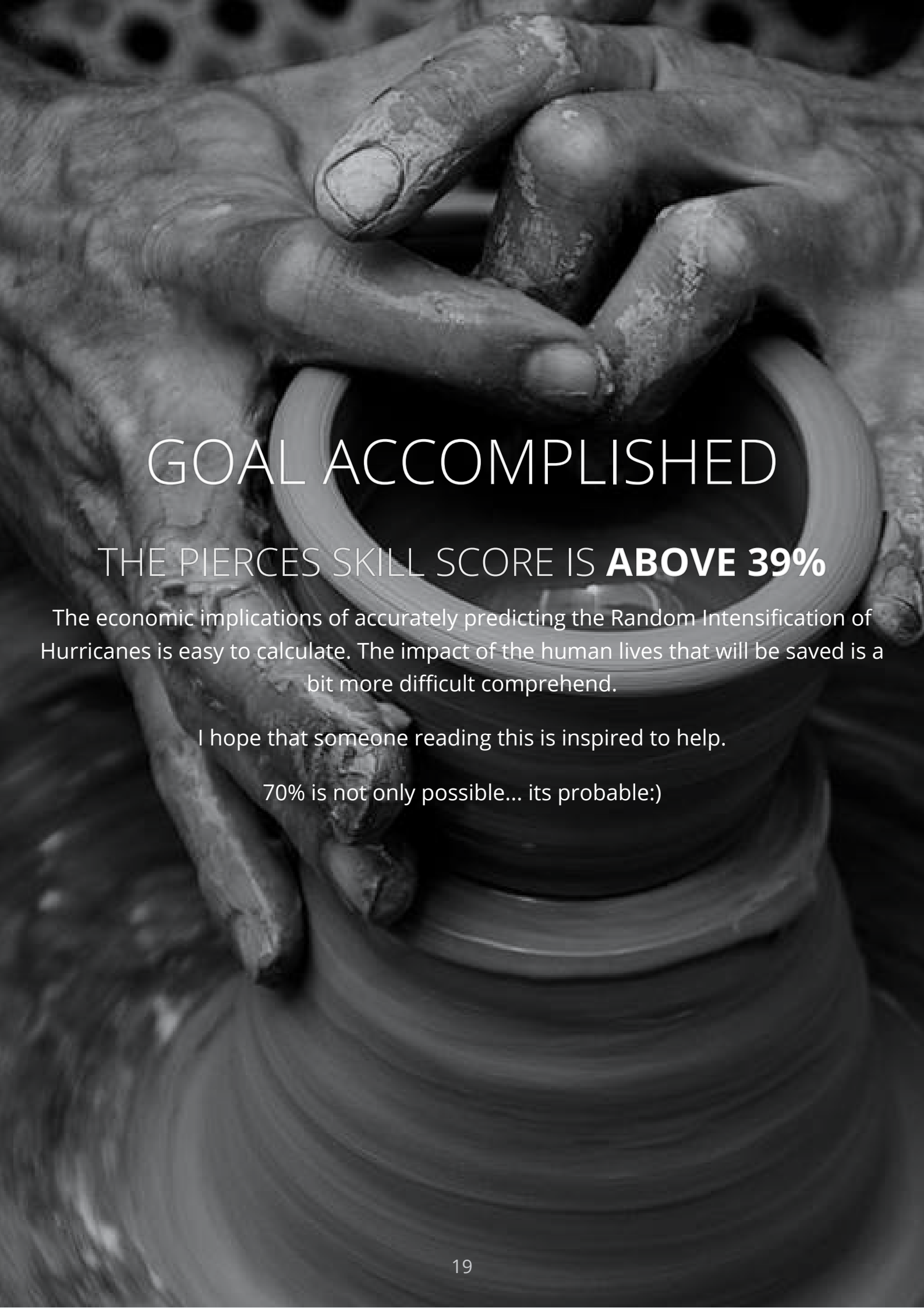
We are going to create another function to run over the feature list. We will be running the program with the top 2 features and seeing what the scores are. Then with the top 3 features. Then with the top 4 features. and so on...

We will be sure to take the warm_start method off as to make sure we are training on a new classifier.

LETS LOOK AT THE SCORES:

THE HIGHEST PERFORMING NUMBER OF FEATURES IS THE TOP 27 FEATURES

PSS(Pierces Skill Score) ~39.8% FAR(False Alarm Ratio) is: ~34.0% POD(Probability of Detection) is: ~ 42.0%



GOAL ACCOMPLISHED

THE PIERCES SKILL SCORE IS **ABOVE 39%**

The economic implications of accurately predicting the Random Intensification of Hurricanes is easy to calculate. The impact of the human lives that will be saved is a bit more difficult comprehend.

I hope that someone reading this is inspired to help.

70% is not only possible... its probable:)

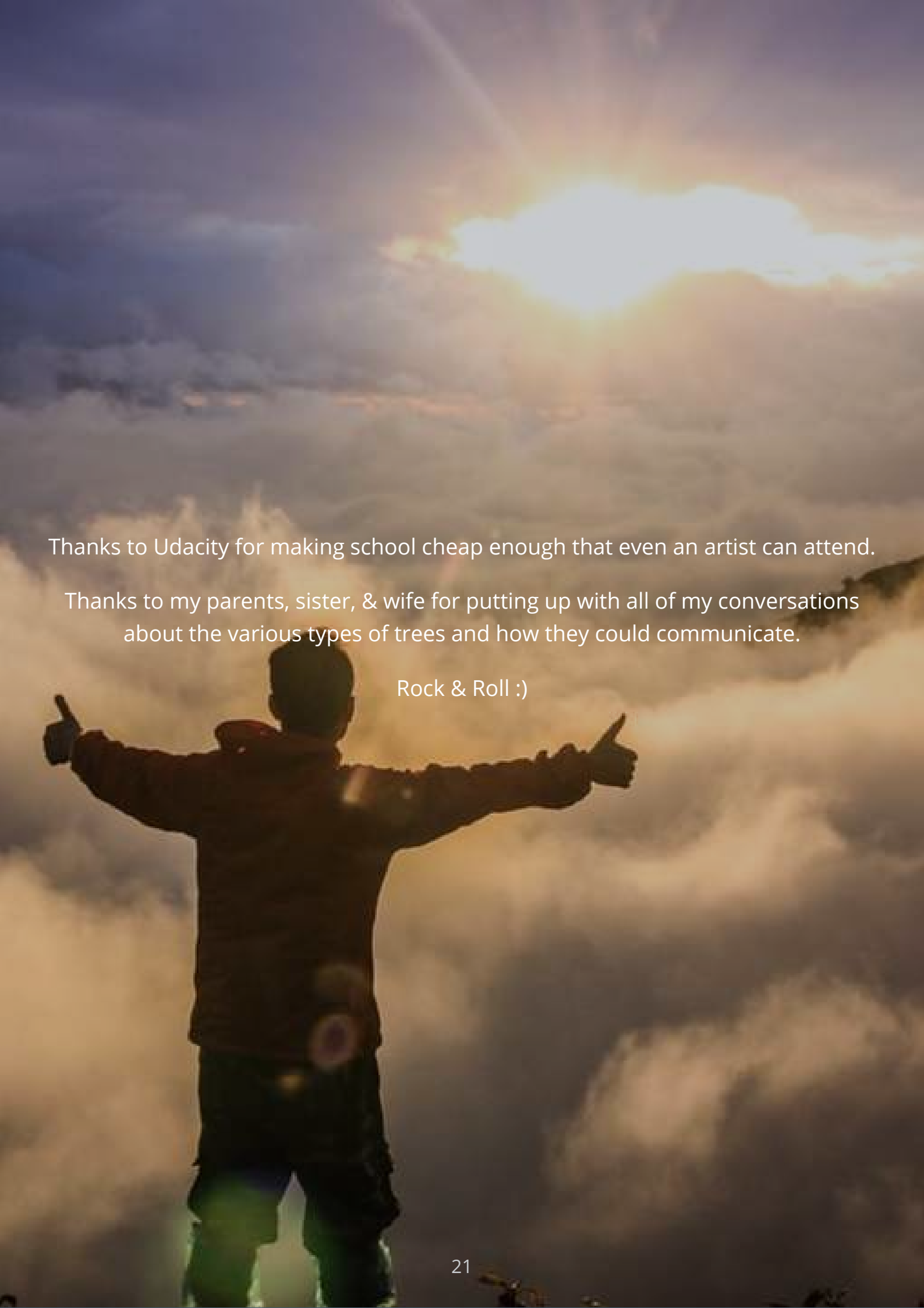
FOLLOWING **INTEREST**

FOR WHATEVER REASON I HAVE GROWN interested ABOUT FORESTS

I first got into machine learning in hopes of better understand trees and how they communicate with one another. I was initially going to work on reforestation but the datasets don't appear to be vast enough yet. I have found a few datasets that showcase how much forests are depleting but I would prefer to work on a solution to the issue versus another program showcasing how much damage we are doing to the earth. :) There are a few companies acquiring data on how trees communicate and that is really interesting to me but the datasets will understandably take time and for this project we didn't really have 20 years to wait for the trees data to come in accurately. One thing that could be beneficial to reforestation is that plants perform better in high CO2 locations.

The knowledge in this field is incredibly complicated. The types of trees that work best together isn't completely understood. But... The CO2 does move freely within the wind patterns.

That is one of the reasons that I wanted this project to be focused on wind.



Thanks to Udacity for making school cheap enough that even an artist can attend.

Thanks to my parents, sister, & wife for putting up with all of my conversations about the various types of trees and how they could communicate.

Rock & Roll :)

- A MACHINE LEARNING PROJECT WITH -

Todd Brueggemann

REFERENCES:

https://www.kdnuggets.com/2019/06/random-forest-vs-neural-network.html

https://towardsdatascience.com/feature-selection-using-random-forest-26d7b747597f

https://www.sciencedaily.com/releases/2019/09/190926114009.htm

https://grmds.org/sites/default/files/inline-files/dataset_guide.pdf

https://scikit-learn.org/dev/developers/develop.html

https://medium.com/@hjhuney/implementing-a-random-forest-classification-model-in-python-583891c99652

https://towardsdatascience.com/random-forest-in-python-24d0893d51c0

https://towardsdatascience.com/gradient-boosting-using-random-forests-for-application-on-the-new-york-taxi-fare-prediction-f6101c592bf9