Machine Learning Engineer

Nanodegree

Capstone Project Report

Catapult distance prediction

Abdullah Ali

8th of March 11, 2021

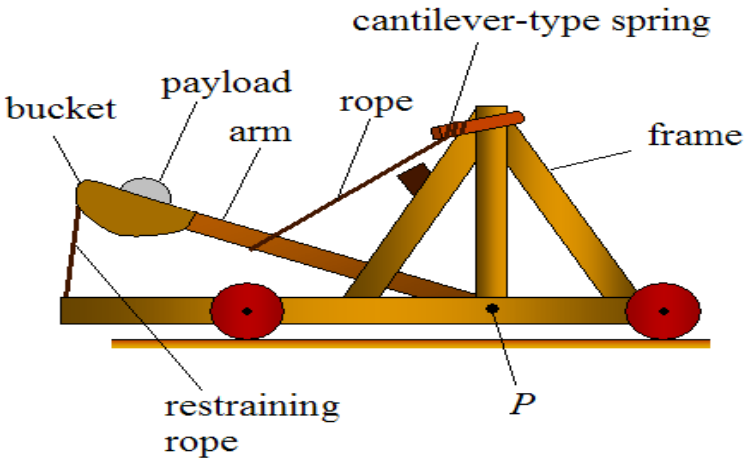
Contents

1 Definition

1.1 Project Overview

Old Day Catapults

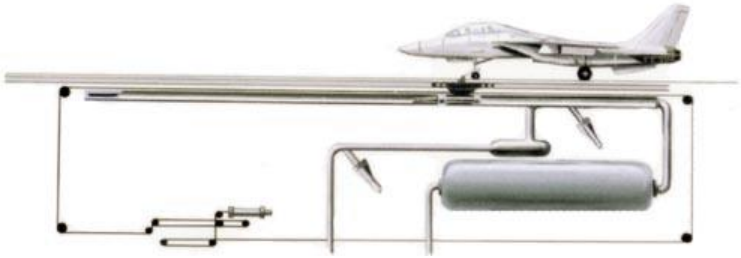
A catapult is a ballistic device used to launch a projectile a great distance without the aid of gunpowder or other propellants – particularly various types of ancient and medieval siege engines. A catapult uses the sudden release of stored potential energy to propel its payload. Most convert tension or torsion energy that was more slowly and manually built up within the device before release, via springs, bows, twisted rope, elastic, or any of numerous other materials and mechanisms. The counterweight trebuchet is a type of catapult that uses gravity. In use since ancient times, the catapult has proven to be one of the most persistently effective mechanisms in warfare. In modern times the term can apply to devices ranging from a simple hand-held implement (also called a "slingshot") to a mechanism for launching aircraft from a ship. The earliest catapults date to at least the 4th century BC with the advent of the mangonel in ancient China, a type of traction trebuchet and catapult. Early uses were also attributed to Ajatashatru of Magadha in his war against the Licchavis. Greek catapults were invented in the early 4th century BC, being attested by Diodorus Siculus as part of the equipment of a Greek army in 399 BC, and subsequently used at the siege of Motya in 397 BC.



Old Day Catapult

Modern Day Catapults

Catapults as siege weapons became ineffective in 885-886 AD rendered useless by new defense technology, but they continue to be used in military operations. The last large-scale use of catapults as a weapon delivery device was in World War 1. Catapults were used to throw hand grenades across No Man's Land and into enemy trenches. Unfortunately for catapults they were soon replace with small mortars. Now catapults are used in target practice to shoot clay pigeons in the air, to launch food at siblings, and the most common use to launch planes into the air. An aircraft carrier doesn't have the runway space to allow for a plane to accelerate to take off as it would on the ground so a catapult is used to propel the aircraft into the air in a very short distance. There is no an academic paper where machine learning was applied to this type of problem. The idea came to my mind when I was completing the final project and I saw there is an option that you can implement your project. I have studied the catapult as part of the Design of Experiment subject in the college and then I thought that I can build a Machine Learning model to predict the distance. It will be much easier



Modern Day Catapult

1.2 Problem Statement

In this Project The main goal is to predict the distance between the projectile when it hit the ground and the catapult depending on several inputs which are release angle, firing angle, cup elevation, pin elevation, and bungee position.

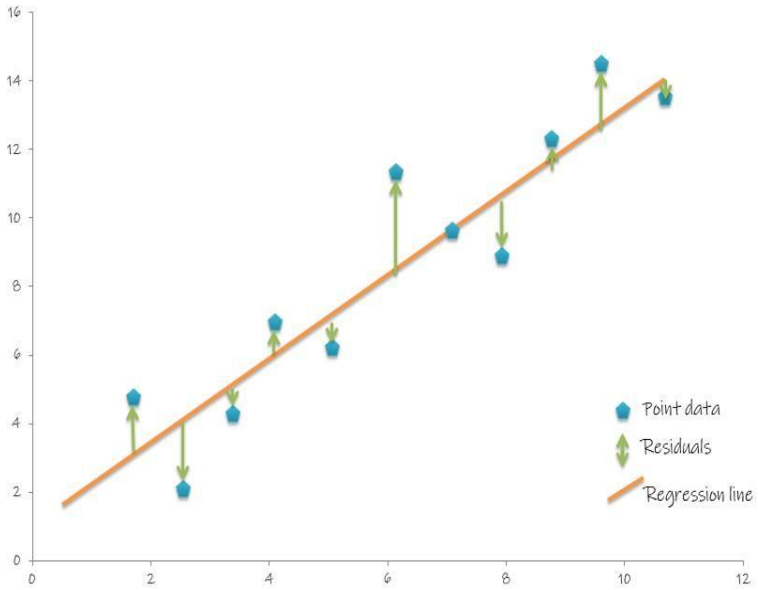
1.3 Metrics

The performance of the models in this project are evaluated by using of Root Mean Square Error (RMSE) Which is frequently used measure of the difference between the predicted values and observed values. Root Mean Square Error (RMSE) is the standard deviation of the residuals (prediction errors). Residuals are a measure of how far from the regression line data points are. Also is a measure of how spread out these residuals are. In other words, it tells you how concentrated the data is around the line of best fit. Root mean square error is commonly used in climatology and regression analysis to verify experimental results. That's why RMSE is a good choice to be the metrics of the models in this project.

Equation of the Root Mean Square Error.

$$RMSE = \sqrt{\sum_{i=1}^{n} \frac{(\hat{y}_i - y_i)^2}{n}}$$

Where $\hat{y}i$ is the predicted value and yi is the actual value.



2 Analysis

2.1 Data Exploration

For this project, the input data is collected from an online simulator of a catapult by changing. Could be also collected from a real Catapult but will take a lot of time for time saving I collected it from the online simulator. link for the simulator is provided in the last page of the report. By changing the five inputs of the experiment (release angle, firing angle, cup elevation, pin elevation, and bungee position) dataset is generated. Noise is considered and added to the experiment output. By taking the average of Distance without noise and Distance with noise. The average distance is the output data. About 460 experiments are done to collect the datasets for this project.

2.2 Exploratory Visualization

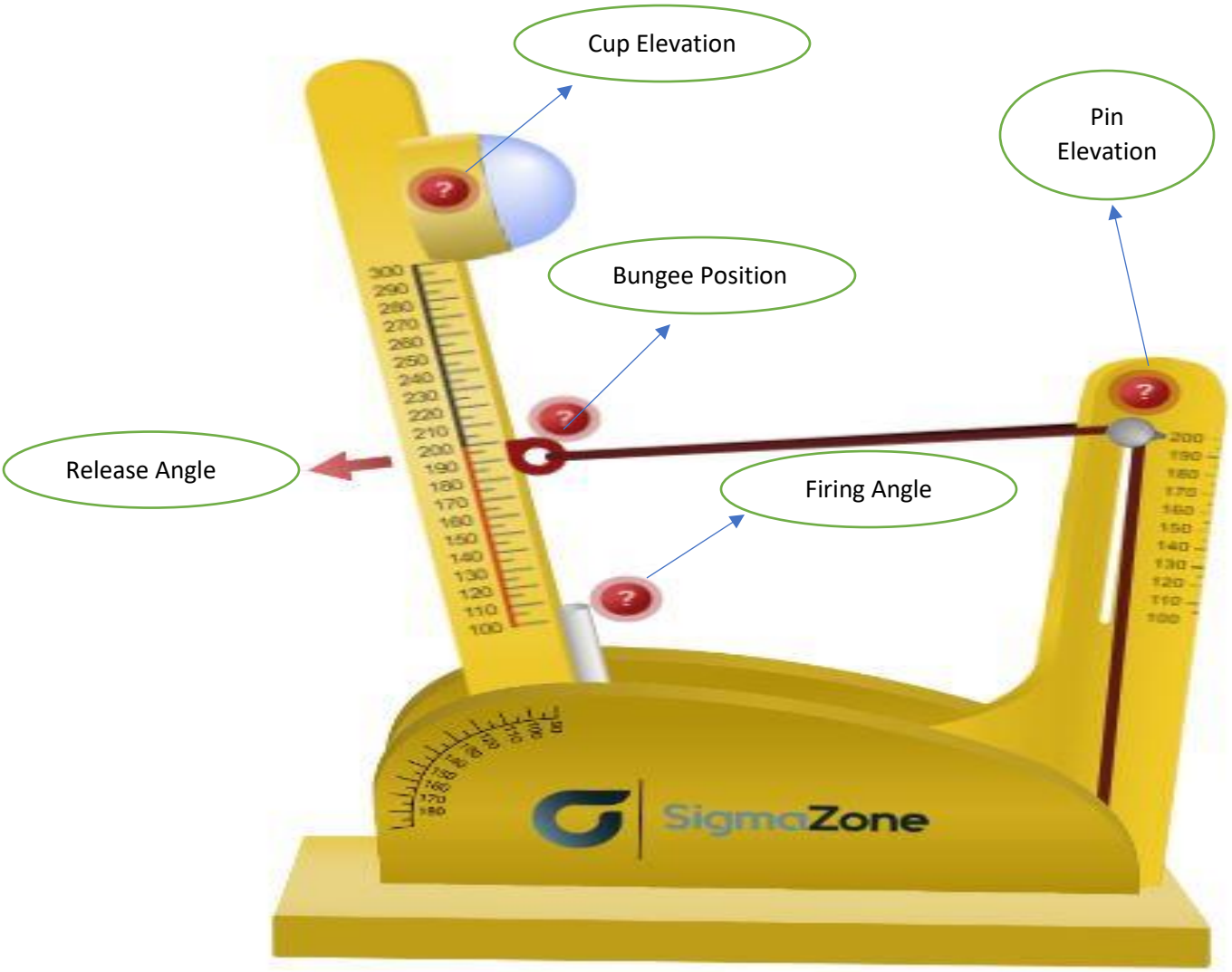
Example of the dataset with the noise distance.

Release angle	Firing angle	Cup elevation	Pin elevation	Bungee pos	Distance With out noise	Distance with noise	Average distance
185	90	300	200	100	196.99	207.89	202.44
185	95	300	200	100	243.40	251.52	247.16
185	100	300	200	100	287.92	305.08	296.5
185	105	300	200	100	331.42	345.32	338.37

Example of the dataset after taking the average of the distance.

Release angle	Firing angle	Cup elevation	Pin elevation	Bungee pos	distance
185	90	300	200	100	202.44
185	95	300	200	100	247.16
185	100	300	200	100	296.5
185	105	300	200	100	338.37

Image of the simulator to explain the input feature



2.3 Algorithms and Techniques

The solution addresses the multi regression supervised learning. The XGBoost model is a good supervised learning approach and is applied to this project.

XGBoost is an optimized distributed gradient boosting library designed to be highly efficient, flexible, and portable. It implements machine learning algorithms under the Gradient Boosting framework. XGBoost provides a parallel tree boosting (also known as GBDT, GBM) that solve many data science problems in a fast and accurate way.

2.4 Benchmark

K-Nearst Neighbors model is used as the benchmark model. Knn algorithm is an index-based algorithm. It uses a non-parametric method for classification or regression. For regression problems, the algorithm queries the *k* closest points to the sample point and returns the average of their feature values as the predicted value. Training with the k-NN algorithm has three steps: sampling, dimension reduction, and index building. Sampling reduces the size of the initial dataset so that it fits into memory.

3 Methodology

3.1 Data Preprocessing

First step is to explore our datasets and see the distribution of the input features. As the dataset is collected by me not a standard dataset then it's important to clean the data and remove any empty rows.
Second step is to sperate the input data from the output data to prepare them for splitting.

	FA	PE	BP	RA	CE
0	90	200	100	185	300
1	95	200	100	185	300
2	100	200	100	185	300
3	105	200	100	185	300
4	110	200	100	185	300
...	...	...	...	...	...
454	90	200	100	185	300
455	140	200	150	185	200
456	140	200	200	185	300
457	115	200	100	185	250
458	90	200	150	185	300

459 rows × 5 columns

	distance
0	202.44
1	247.16
2	296.50
3	338.37
4	364.61
...	...
454	199.54
455	187.45
456	450.42
457	300.05
458	234.43

459 rows × 1 columns

XGBoost is not sensitive to monotonic transformations of its features for the same reason that decision trees and random forests are not: the model only needs to pick "cut points" on features to split a node, So there is no need now to scale the dataset.

Third step is to split the data to Train, Validation and Test data to fit them into XGBoost model.
As the XGBoost is implemented in amazon sage maker the separated data are saved as csv files and uploaded to s3 bucket.

Before fitting the data to K-Nearst Neighbors model data is scaled using MinMaxScaler.

3.2 Implementation

Implementation of XGBoost model in SageMaker is done using train instance type: 'm1.m4.xlarge' with hyperparameter: max_depth=5 , eta=0.2 , gamma=4, min_child_weight=6, subsamples=0.8, early_stopping_round= 10, num_rounds=200, and using objective=regression linear.

Second model is created is K-Nearst Neighbors model by trying different values of K to find the least Root Mean Square Error (RMSE).
The least Mean Square Error found is 34.16 at K=6 as you can see at the image below, so the model need to be improved

```
RMSE value for k=  1 is: 34.95626054727931
RMSE value for k=  2 is: 35.553087788096775
RMSE value for k=  3 is: 35.896538624012
RMSE value for k=  4 is: 36.59602919949499
RMSE value for k=  5 is: 34.42029261764139
RMSE value for k=  6 is: 34.16060420557599
RMSE value for k=  7 is: 34.762336094774255
RMSE value for k=  8 is: 35.68191287612106
RMSE value for k=  9 is: 37.37582015261836
RMSE value for k=  10 is: 39.30196694924709
RMSE value for k=  11 is: 41.2752624066877
RMSE value for k=  12 is: 42.94294990722107
RMSE value for k=  13 is: 45.90368588228085
RMSE value for k=  14 is: 47.028466227926856
RMSE value for k=  15 is: 48.79810148395123
```

3.3 Refinement

The initial solution was a baseline of XGBoost model with max dept=5 as we can see the image below is the output of the model. The initial Root Mean Square Error (RMSE) was very high but it start to descend till the training rmse reached 16.75 while the validation rmse stopped at 23.22 and the Root Mean Square Error (RMSE) is no longer getting down the model stopped at these values which is still quiet high which reflect the accuracy is not high.

```
[5]#011train-rmse:91.88368#011validation-rmse:97.09563
[6]#011train-rmse:77.42004#011validation-rmse:82.56467
[7]#011train-rmse:66.31223#011validation-rmse:71.54710
[8]#011train-rmse:55.69284#011validation-rmse:60.09089
[9]#011train-rmse:48.48734#011validation-rmse:51.98401
[10]#011train-rmse:41.79656#011validation-rmse:44.97516
[11]#011train-rmse:36.83006#011validation-rmse:39.83061
[12]#011train-rmse:32.69567#011validation-rmse:35.64982
[13]#011train-rmse:29.69037#011validation-rmse:32.74931
[14]#011train-rmse:27.06215#011validation-rmse:29.63918
[15]#011train-rmse:25.18512#011validation-rmse:27.96710
[16]#011train-rmse:23.75727#011validation-rmse:26.25259
[17]#011train-rmse:22.50885#011validation-rmse:25.29181
[18]#011train-rmse:21.65473#011validation-rmse:24.78233
[19]#011train-rmse:21.08096#011validation-rmse:23.62229
[20]#011train-rmse:20.14702#011validation-rmse:23.48614
[21]#011train-rmse:19.70800#011validation-rmse:23.64281
[22]#011train-rmse:19.02225#011validation-rmse:23.58634
[23]#011train-rmse:18.84436#011validation-rmse:23.06558
[24]#011train-rmse:18.69344#011validation-rmse:22.48679
[25]#011train-rmse:18.59953#011validation-rmse:22.28549
[26]#011train-rmse:18.34987#011validation-rmse:22.19265
[27]#011train-rmse:17.97172#011validation-rmse:22.49504
[28]#011train-rmse:17.56084#011validation-rmse:22.71674
[29]#011train-rmse:17.29011#011validation-rmse:22.95303
[30]#011train-rmse:17.14390#011validation-rmse:22.72234
[31]#011train-rmse:16.92236#011validation-rmse:22.92964
[32]#011train-rmse:16.89583#011validation-rmse:22.65861
[33]#011train-rmse:16.74309#011validation-rmse:22.53364
[34]#011train-rmse:16.52520#011validation-rmse:22.78348
[35]#011train-rmse:16.37380#011validation-rmse:23.10651
[36]#011train-rmse:16.17508#011validation-rmse:23.22104
Training seconds: 65
Billable seconds: 65
```

Trying different hyperparameters are always a good solution to improve your model and to see which parameters is fit better for the input data so a hyperparameter tuner model is implemented based the on the XGBoost model defined earlier to find the best training job which have the east rmse as defined in the image below.

```
(estimator = xgb, # The estimator object to use as the basis for the training jobs.
 objective_metric_name = 'validation:rmse', # The metric used to compare trained models.
 objective_type = 'Minimize', # Whether we wish to minimize or maximize the metric.
 max_jobs = 20, # The total number of models to train
 max_parallel_jobs = 3, # The number of models to train in parallel
 hyperparameter_ranges = {
      'max_depth': IntegerParameter(3, 15),
      'eta'      : ContinuousParameter(0.05, 0.6),
      'min_child_weight': IntegerParameter(2, 12),
      'subsample': ContinuousParameter(0.5, 0.9),
      'gamma': ContinuousParameter(0, 10),
 })
```

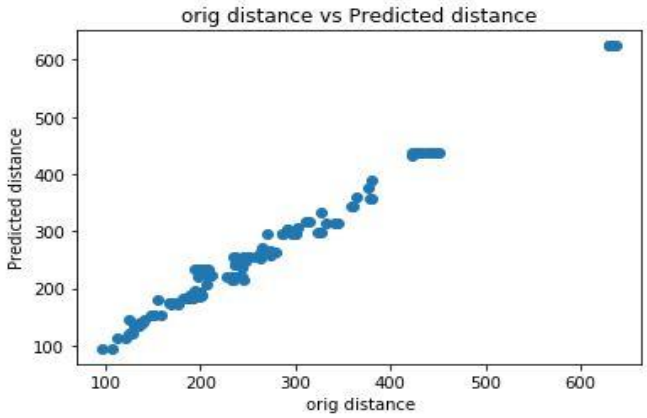
The model finds the best training job which have the least validation rmse 16.935 which is slightly lower than the original XGBoost model.

```
[17]#011train-rmse:23.91851#011validation-rmse:19.74479
[18]#011train-rmse:23.34899#011validation-rmse:18.23616
[19]#011train-rmse:22.96990#011validation-rmse:16.99404
[20]#011train-rmse:22.34405#011validation-rmse:17.02051
[21]#011train-rmse:21.86799#011validation-rmse:17.93920
[22]#011train-rmse:21.73258#011validation-rmse:17.04286
[23]#011train-rmse:21.58776#011validation-rmse:16.56973
[24]#011train-rmse:21.49809#011validation-rmse:16.26522
[25]#011train-rmse:21.48163#011validation-rmse:15.61484
[26]#011train-rmse:21.28518#011validation-rmse:16.49027
[27]#011train-rmse:21.13434#011validation-rmse:16.08038
[28]#011train-rmse:20.89516#011validation-rmse:17.67162
[29]#011train-rmse:20.22129#011validation-rmse:17.56289
[30]#011train-rmse:19.81745#011validation-rmse:17.14425
[31]#011train-rmse:19.12803#011validation-rmse:17.89457
[32]#011train-rmse:19.20220#011validation-rmse:17.19794
[33]#011train-rmse:19.02950#011validation-rmse:17.01191
[34]#011train-rmse:18.75238#011validation-rmse:17.53011
[35]#011train-rmse:18.77600#011validation-rmse:16.93518
Training seconds: 69
Billable seconds: 69
```

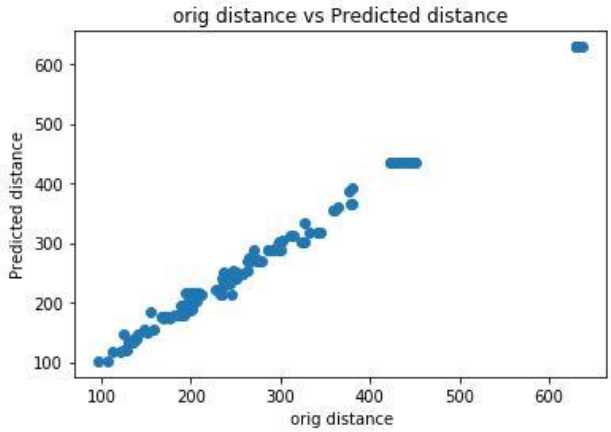
4 Results

4.1 Model Evaluation and Validation

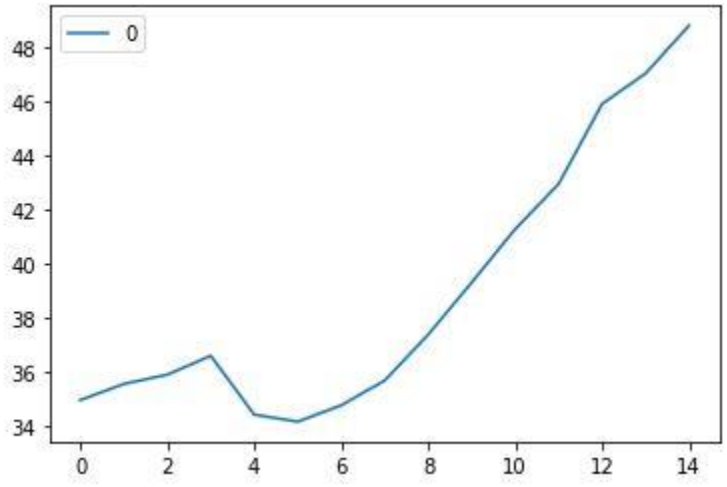
Simple scatter plot between the predicted and actual distance values by XGBoost model it's not fitted as linear line due to the rmse



Simple scatter plot between the predicted and actual distance values by XGBoost after adjusting the hyperparameter model is more linear



Scatter to show the relation between the number of k and rmse developed by the Knn model



The accuracy of the model developed by Knn model

```
#print the accuracy
model.score(X_train_s,Y_train_s)
```

```
0.8145896940834727
```

4.2 Justification

The model's performance needs to be improved I have written some suggestion below in the improvement section to improve the performance of the model

5 Conclusion

5.1 Free-Form Visualization

On this occasion I would like to show the predicted output of the developed model knn on some inputs.

```
#test the model and predict the distance
Xt = [[100,200,200,185,300]]; # ['FA','PE','BP','RA','CE']
print("Distance predicted by model: \t\t\t\t%.0f" % np.round(model.predict(scaler.transform(Xt)),
1))
```

```
Distance predicted by model:                    304
```

5.2 Reflection

The process used for this project can be summarized using the following steps:

Step 1:

- Collect the data using the catapult simulator.
- Take the average of the output distance using excel.

Step 2:

- Load the catapult dataset to the Jupiter notebook.
- Clean the data and remove any repeated or empty rows.
- Distribute the value for each feature.
- Separate the input data and the output data.
- Separate the data to train and test using Sklearn.

Step 3:

- Locate the data directory and save the test, train, and validation to csv files.
- Upload the csv files to s3 to use them later in the model.
- Build a XGBoost model using SageMaker.
- Set the hyperparameter and fit the model using input and validation data
- Deploy the model for prediction.

- Create a simple scatter plot between the predicted and actual distance values

Step 4:

- Prepare the input dataset and scale it.
- Create the K-Nearst Neighbors model using Sklearn Library.
- Fit the model with input and test data.
- Plotting the rmse values against k values
- Find the value of K which have the least value of Root Mean Square Error (RMSE).
- Test the model and predict the distance.

Step 5:

- Implementing Knn model in SageMaker.
- Locate the data directory and save the test, train, and validation to csv files.
- Upload the csv files to s3 to use them later in the model.
- Build a Knn model and set the hyperparameter.
- Deploy the model and test the model (not included in the Capstone project)

Step 6:

- Use SageMaker hyperparameter tuning for XGBoost model which defined earlier.
- Find the best training job.
- Deploy the adjusted model based on the best training job.
- Create a simple scatter plot between the predicted and actual distance values

5.3 Improvement

- Increase the number of the dataset used for this project
- Mean the value of train data for prediction to get a better RMSE
- Write a function to do cross validation that will help to get a better RMSE estimate of the model.

References

Simulator link: https://sigmazone.com/catapult/

Catapult grid interference to collect the data: https://sigmazone.com/catapult-grid/

Catapult Background: https://en.wikipedia.org/wiki/Catapult

https://docs.aws.amazon.com/sagemaker/latest/dg/k-nearest-neighbors.html

https://www.statisticshowto.com/probability-and-statistics/regression-analysis/rmse-root-mean-square-error/

https://sites.google.com/site/physicsofcatapults/home/how-a-catapult-works-the-basics

https://www.statisticshowto.com/probability-and-statistics/regression-analysis/rmse-root-mean-square-error/

https://xgboost.readthedocs.io/en/latest/