Jake Cupani

INST414

Final Project

Detecting Radiation Events from the NASA GEDI Satellite

Section I

Introduction

As an intern at NASA Goddard in Greenbelt, Maryland, one of my mentors, Bryan Blair, tasked me with creating a machine learning model that can identify radiation anomalies in data collected by the GEDI satellite. The GEDI satellite is a satellite operated by NASA which uses LIDAR, or "LIght Detection and Ranging", technology to create 3D topographic maps of the surface of the earth. This technology can be used for a variety of applications including: detecting forest heights, glacial surface elevation changes, and other topographic use cases (GEDI, 2020). The problem, however, is that there are mysterious radiation events, or anomalies, that happen over the South Atlantic. These anomalies cause the detector to register extremely high signals. Scientists are unsure of the cause of these radiation events or why they happen almost exclusively in the South Atlantic, but by developing a classifier we can determine the rate of occurrence of these events, which might give us more insight into the source of the anomalies. Lastly, these results will be given to the radiation team at NASA Goddard to compare their models with the ones outlined in this report. From the analysis, I can conclude that the rate of occurrence of these anomalies is around 0.029%, which clearly shows just how rare these occurrences are. The K Nearest Neighbors classifier was able to successfully identify anomalous data with an accuracy of 95.6%, with undersampling. I believe that a classification machine learning algorithm would work best for this dataset, since we are trying to place the dependent variable (sample) into two classes: radiation event and non-radiation event. Anomalous data in this report is defined as any samples that are 2 standard deviations away from the mean noise levels.

Section II

Data

The data used in this analysis was originally obtained from my internship mentor, Bryan Blair, with explicit permission to use within this class. Different versions of the data are available via the NASA EarthData Portal (EarthData, 2020). The data is approximately 4 gigabytes in size, meaning it is quite a large dataset to analyze, although large sample sizes are needed to find the anomalies. The quantity of the data compared to the number of actual anomalous data is shown below, with only 0.02% of the data being anomalous. However, the true rate of occurrence would need more data to properly calculate. The descriptive statistics in Figure 1 also show that the average anomalous value is around 790, which is a large deviation from the noise mean value of around 234.

Figure 1.

Noise Mean	
Count	13075414
Mean	234.214
Std	13.97808
Min	204.0625
25%	227.75
50%	240.6875
75%	245.1875
Max	254.0625

Anomalous Data	
Count	3786
Mean	790.3249
Std	260.7786
Min	616
25%	708
50%	738
75%	787
Max	4095

Furthermore, in order to correctly perform any kind of analysis on the variables in question, we need to first check if there are any correlations between the variables which might affect our results. Based on Figure 2, we can see that the majority seem to not be correlated, and the ones that are correlated, like temperature and altitude, make sense in this domain. The independent variables for this analysis are laid out according to Figure 3.

Figure 2: (To view larger: https://imgur.com/a/ymlQm2s)

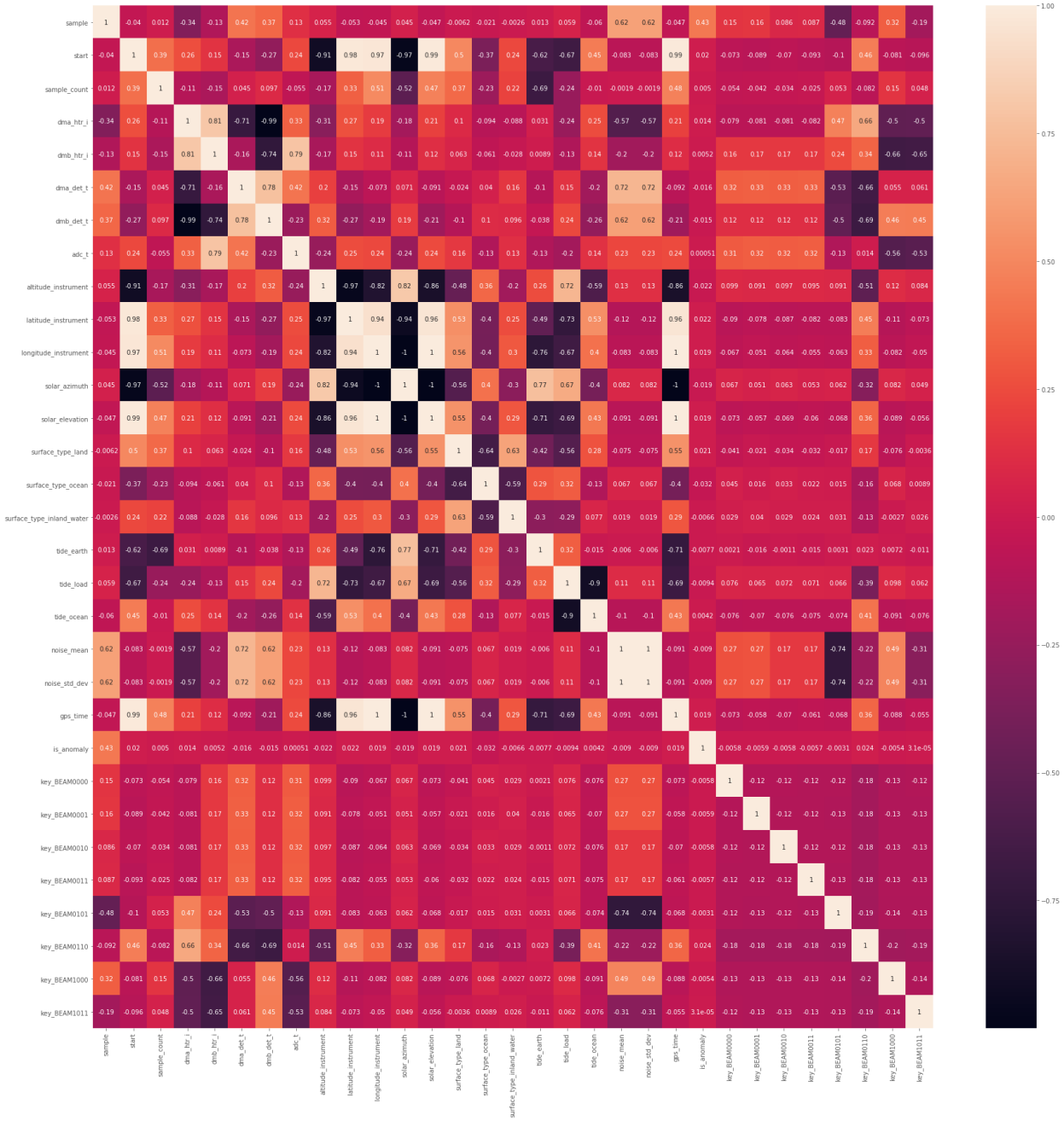


Figure 3:

Variable Name	Variable Description	DV/IV
sample	The return waveform sample from the lidar detector	DV
start	Variable used to indicate start of sample	IV
sample_count	Variable used to indicate the length of the sample	IV
dma_htr_i	Detector module #1 detector heater current.	IV
dmb_htr_i	Detector module #2 detector heater current.	IV
dma_det_i	Detector module #1 detector temperature	IV
dmb_det_i	Detector module #2 detector temperature	IV
adc_t	Analog to Digital Converter temperature	IV
altitude_instrument	Instrument altitude	IV
latitude_instrument	Instrument latitude	IV
longitude_instrument	Instrument longitude	IV
solar_azimuth	defines the Sun's relative direction along the local horizon,	IV
solar_elevation	Apparent elevation of the sun	IV
surface_type_land	Flag for surface type land	IV
surface_type_ocean	Flag for surface type ocean	IV
surface_type_inland_water	Flag for surface type inland water	IV
tide_earth	The displacement of the solid earth's surface caused by the gravity of the Moon and Sun	IV
tide_load	the deformation of the Earth due to the weight of the ocean tides	IV
tide_ocean	Ocean tide range	IV
noise_mean	Mean noise level of the sample	IV
noise_std_dev	Standard deviation of the sample	IV
gps_time	GPS timestamp	IV
datetime	Datetime Timestamp	IV

Methods

Since this data originated in what is known as HDF format, a Python module was needed in order to convert it into a Python object. For this, I used a module called h5py. Once I had the HDF file converted into a Python object, I needed to get a random sample from each of the detectors in the HDF file, as there is data collected for each of the 8 detectors. The “rxwaveform” variable which contains each of the return signals from the lidar sensor, is padded with zeros in order to separate sets of return signals. In order to take a random sample from each detector, while ignoring the padded zeros, the “rx_sample_start_index” and “rx_sample_count” variables were used to index through the “rxwaveform” variable. The sets were taken from a random sample of the data. The corresponding data from other variables was then all put into a dataframe list, which was then concatenated to form one final dataframe. Finally, to flag the anomalous data, I converted any anomalies to 1 and any other samples to 0. This flag is then used to classify whether or not a given sample is anomalous. The next step in preparing the data for analysis was to undersample the dataset. Due to the fact that there is so little anomalous data compared to the non-anomalous data, there exists a class imbalance. Therefore in order to balance the classes, we must create a new dataset that is undersampled. The undersampling process takes all of the anomalous data and their indices and concatenates it with an equal amount of randomly selected non-anomalous data. This ultimately creates a new dataset that is balanced and able to be put into a machine learning algorithm. If the data was not balanced and put into the machine learning models, it would be almost entirely trained on non-anomalous data, skewing results and creating

false conclusions. Figure 4 and Figure 5 show the relative class balances before and after undersampling, respectively.

Figure 4:

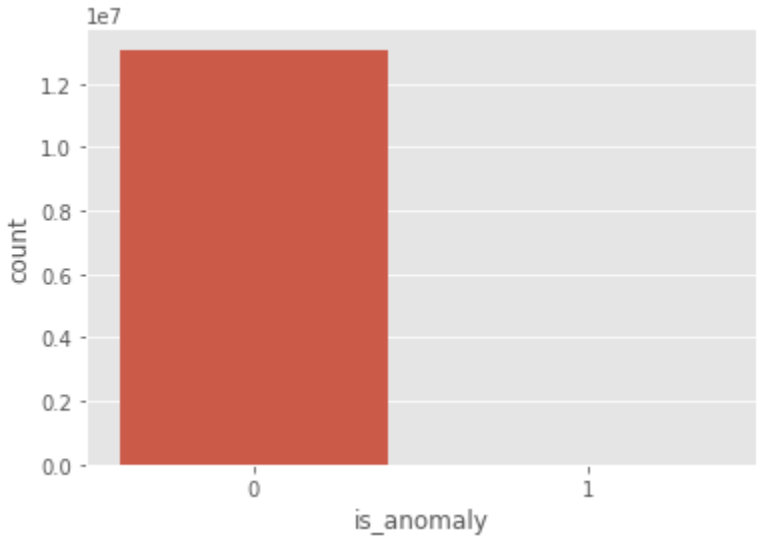
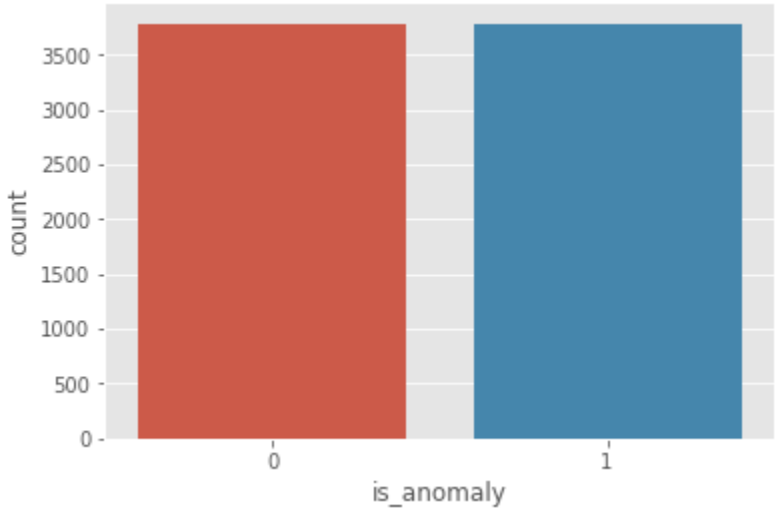


Figure 5:



Analysis

Now that the data has been manipulated to be able to be put into a machine learning model, all that is left in the analysis is to split the data into training and test sets, and run the different algorithms on the test set. The data needs to be split into training and test sets so that the

machine learning models are not predicting on data that it was previously trained on, otherwise the results would be skewed. The variable in this case that we are trying to predict is the "is_anomaly" variable. In other words, we want to be able to predict whether or not a sample is anomalous or not based on our independent variables. In this analysis I used Logistic Regression, Naïve Bayes, K Nearest Neighbors, and Random Forest algorithms to classify the "is_anomaly" variable. I decided to run each of these and then ultimately decide on whichever one performed the best.

Results

After running each of the algorithms, the following results were recorded by using the generate_model_report function (Figure 6).

Figure 6:

	Logistic Regression	KNN	Naïve Bayes	Random Forest
Accuracy	0.88	0.96	0.93	0.99
Precision	0.82	0.93	0.89	0.99
Recall	0.98	0.99	0.99	1
F1 Score	0.89	0.96	0.94	0.99

Based on these results, I have decided to go with the K Nearest Neighbors algorithm for a multitude of reasons. First off, due to the Random Forest model producing oddly accurate results of essentially 1.0, I have decided to not consider it in order to preserve the integrity of the results. Although this is a minor pitfall, I believe it is best to leave it out for now and look further into this in subsequent research. I believe that the problem with the Random Forest model is just the small amount of data it was fed. Ultimately, I believe the K Nearest Neighbors algorithm would work best for classifying the "is_anomaly" variable since it had the best overall metrics amongst the considered algorithms (besides Random Forest). Additionally, since we are mainly concerned with identifying anomalies when they are actually anomalous, I believe the recall metric is most important. Luckily, the K Nearest Neighbors algorithm performed quite well in terms of recall, with a score of 0.99, meaning it almost always predicted yes when the "is_anomaly" variable was 1, or in other words actually anomalous. Therefore, it is quite clear that the KNN algorithm is best suited for this dataset.

Section III

Conclusion

Clearly, this analysis has shown that given the data available, it is certainly possible to create a machine learning model that is able to successfully classify anomalous and non-anomalous radiation events. Furthermore, it has been shown that the K Nearest Neighbors algorithm performs the best out of the selected algorithms. The K Nearest Neighbors algorithm is often used in anomaly detection, which makes sense why it did so well in this application. I look forward to continuing to research how to improve these models, and how they compare to the radiation team's models at NASA Goddard.

Appendix

1. "imbalance_class_undersampling_oversampling.ipynb", Bhavesh Bhatt, 21 Apr. 2019, https://github.com/bhattbhavesh91/imbalance_class_sklearn/blob/master/imbalance_class_undersampling_oversampling.ipynb
2. "GEDI01_B v001." *LP DAAC - GEDI01_B*, NASA, 2020, lpdaac.usgs.gov/products/gedi01_bv001/.
3. "Science Overview." *GEDI*, NASA, 20 Apr. 2018, gedi.umd.edu/science/objectives-overview/.