SMU Data Science Review

Volume 2 | Number 1 Article 11

2019

Visualization and Machine Learning Techniques for NASA's EM-1 Big Data Problem

Antonio P. Garza III
Southern Methodist University, antoniog@smu.edu

Jose Quinonez
Southern Methodist University, jquinonez@smu.edu

Misael Santana
Southern Methodist University, mlsantana@smu.edu

Nibhrat Lohia
Southern Methodist University, nlohia@smu.edu

Recommended Citation

Garza, Antonio P. III; Quinonez, Jose; Santana, Misael; and Lohia, Nibhrat (2019) "Visualization and Machine Learning Techniques for NASA's EM-1 Big Data Problem," *SMU Data Science Review*: Vol. 2: No. 1, Article 11.
Available at: https://scholar.smu.edu/datasciencereview/vol2/iss1/11

Visualization and Machine Learning Techniques for NASA's EM-1 Big Data Problem

Antonio P. Garza, III[1], Jose Quiñónez[2], Misael Santana[3], Nibhrat Lohia[4]
. [1]antoniog@smu.edu, [2]jquinonez@smu.edu, [3]mlsantana@smu.edu , [4]Master of Science in Data Science, Southern Methodist University,
Dallas, TX 75275 USA, nlohia@smu.edu

Abstract. In this paper, we help NASA solve three Exploration Mission-1 (EM-1) challenges: data storage, computation time, and visualization of complex data. NASA is studying one year of trajectory data to determine available launch opportunities (about 90TBs of data). We improve data storage by introducing a cloud-based solution that provides elasticity and server upgrades. This migration will save $120k in infrastructure costs every four years, and potentially avoid schedule slips. Additionally, it increases computational efficiency by 125%. We further enhance computation via machine learning techniques that use the classic orbital elements to predict valid trajectories. Our machine learning model decreases trajectory creation from hours/days to minutes/seconds with an overall accuracy of 98%. Finally, we create an interactive, calendar-based Tableau visualization for EM-1 that summarizes trajectory data and considers multiple constraints on mission availability. The use of Tableau allows for sharing of visualization dashboards and would eventually be automatically updated upon generation of a new set of trajectory data. Therefore, we conclude that cloud technologies, machine learning, and big data visualization will benefit NASA's engineering team. Successful implementation will further ensure mission success for the Exploration Program with a team of 20 people accomplishing what Apollo did with a team of 1000.

Keywords: Big Data, Machine Learning, Regression, Visualization, Cloud Computing

1 Introduction

Exploring the Moon and space in general have been dreams of mankind for generations. In the 1960's NASA's Apollo program succeeded in making the dream of going to the Moon a reality. The project required a great effort and an extreme level of coordination between all participants to achieve a common goal. With a fraction of the budget and hence, a fraction of the manpower, NASA will attempt to return to the Moon in 2020. This paper will provide a framework to help NASA achieve their goals efficiently and cost effectively.

1.1 Going to the Moon

In the early stages of Apollo planning, the team identified more than 10,000 activities to be done, with no previous experience of this endeavor's magnitude to decide what to do sequentially or in parallel. Key questions that drove the entire project were: determination of the environment in cis-lunar space (volume of the lunar orbit), and on the lunar surface; design and development of the spacecraft and launch vehicles; the conduct of tests and flight missions to prove components and procedures; and the selection and training of flight crews and ground support to carry-on the mission [1].

Constraints on propulsion and performance efficiency triggered the project strategy in the Apollo project in the 60's. The same constraints and requirements are still valid for Exploration Mission-1 (EM-1). EM-1 will be the first integrated flight test of NASA's Deep Space Exploration Systems: the Orion spacecraft, the Space Launch System (SLS) rocket, and the Exploration Ground Systems at Kennedy Space Center in Cape Canaveral, Florida [3]. The SLS rocket includes the Interim Cryogenic Propulsion Stage (ICPS) providing the system with the required trans-lunar injection (TLI) burn to reach lunar orbit.

1.2 EM-1 Data Problem

EM-1 is currently planned for 2020. Due to rocket performance, orbital mechanics and the acceptable launch azimuth (direction angle from north pole clockwise) as allowed from the Florida coast, getting to the Moon is not simply a matter of picking a desired launch day any time of the year. In fact, during Apollo mission planning, engineers realized that having one lunar landing location, along with the performance limitations of the lunar lander, and the lighting constraints for landing left only ten viable days in which to launch each year. To increase the launch window for Apollo missions, NASA decided that more lunar landing locations were required to satisfy these constraints [21].

EM-1 will not be landing on the Moon. Even still, we will have all of the orbital mechanics, earth-moon geometry, and rocket performance challenges that were part of the Apollo problem. To study all the segments of an earth-moon trajectory, the JSC trajectory engineering team creates terabytes of data for both trajectory study and final submittal to the flight operations team and mission management. This data can be used to determine mission availability. A typical one-week launch window will generate about 2TB of data given the standard resolution of a trajectory per minute. Therefore, our first task is to determine how best to store and manipulate the data in order to find the days available to launch for EM-1. Trajectory scans are run at daily and sometimes minute resolutions. These trajectory scans will eventually include the entirety of the Exploration Mission Design Matrix which encompasses nominal, alternate mission, abort, and missed burn cases (total of 64 cases and 90 TB of data).

Currently, the engineering team stores their data in standard server file structures and in text, csv, or json formats that are the output formats of the trajectory calculation tool called Copernicus. While this has data management has sufficed for their current level of data (nominal launch scenarios only), the need to integrate alternative and other off-nominal scenarios will require further storage capabilities for data handling and elasticity.

1.3 EM-1 Launch Opportunity Discovery

Although EM-1 will only perform several orbits around the Moon, there are still many performance, communication and lighting constraints that require significant analysis in order to have a large enough launch period each month to successfully launch. By studying the trajectory and other pertinent mission and sun positioning data, we will determine when these launch opportunities exist. Orbital mechanics studies the motion of artificial satellites and space vehicles moving under the influence of forces such as gravity, atmospheric drag, and thrust [4].

The constraints to optimize spacecraft performance have designed trajectories to use gravitational forces and reduce fuel burns [5].

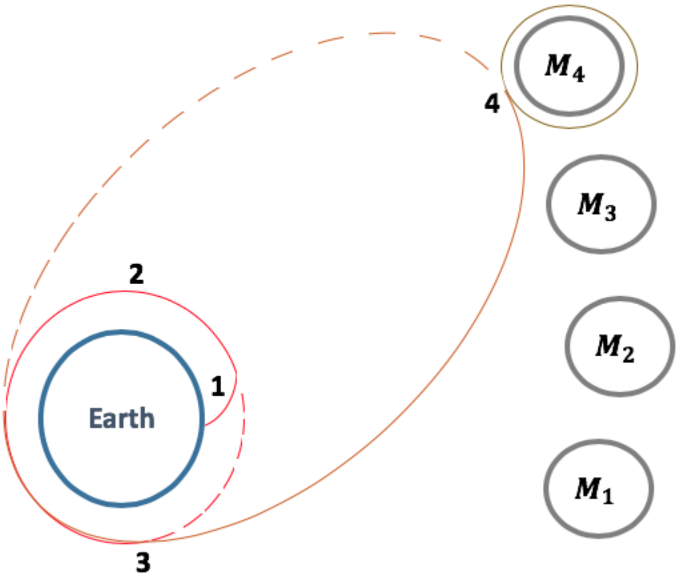


Fig. 1.The launch window must be calculated to optimize fuel consumption while getting to the Moon just on time.

The TLI burn is sized and timed to precisely target the Moon as it revolves around the earth. The launch window, therefore, is the range of time when the spacecraft can be launched so it will intercept the Moon's orbit at the correct time. The following series of events are depicted in Fig. 1:

1. Lift off. At this time the Moon is at a position M1, but it is calculated where it is going to be at TLI (M3) and the interception to translunar trajectory (M4).
2. The vehicle enters a parking orbit around the earth (low Earth orbit).
3. After the required checks have been performed, the TLI burn is the propulsive maneuver that increases the spacecraft's velocity changing its orbit from the parking orbit into its translunar trajectory in a highly eccentric orbit.
4. The spacecraft intercepts the Moon's sphere of influence, and enters into its orbit.

Our paper includes the ability to pull from the large trajectory data set the required data to find the available launch opportunities in a range of time, in order to have several options to manage risk.

1.4 EM-1 Launch Opportunity Visualization

Next, we need to find a way to visualize these launch opportunities and toggle on and off established requirements as well as nice-to-have requirements, or "desirements", for this EM-1 mission across the entire Exploration Mission Design Matrix. This matrix will eventually include nominal, alternate, abort, and missed burn mission trajectories. Also, we will need to be able to determine and visualize where "cut-outs" exist inside of a launch window. A "cut-out" is an amount of time where you either cannot launch due to a failure to meet a requirement or could signify a timeframe where you lose a "desirement".

Our approach requires a way to consolidate all of the trajectory data across the mission design matrix and create a subset of data that can be used to visualize launch opportunities. This visualization needs to be layered in such a way that the impacts of multiple requirements and "desirements" can be quickly demonstrated in order for decision-makers to make informed decisions on when to launch. For instance, if we were to choose to have a lighted launch to be able to have good imagery of the launch and the ascent phase of the mission, we would lose about 1/3 of all the available opportunities in a given year. Additionally, the tool would have to have the capability to perform more of a deep-dive into a particular day/week to determine whether there are times within an available launch day that preclude you from meeting either highly-desired objectives or where you do not have an available abort.

1.5 EM-1 Machine Learning Approach

Lastly, with the desired architecture in place that accommodates the storage, sharing, querying, and visualizing of the data, we are now able to apply machine learning techniques across the growing dataset in order to increase the trajectory scan resolution. Currently, when performing trajectory design, scans are run at a one-day resolution to cut down on overall computation time and get a "ballpark" trajectory. The drawback of this approach is that the subsequent trajectory scan initializes itself with the previous scan which is an inaccurate initial condition. The more inaccurate the initial condition, the more time the 3 degree of freedom (DOF) trajectory design tool, Copernicus, takes to resolve a valid trajectory.

We are looking at machine learning approaches using supervised regression that would ease the computational burden on different aspects of the overall trajectory calculations. Additionally, near flight, if increased resolution on the trajectory scans is requested, the machine learning approach could be utilized to provide valid classical orbital elements information that could be propagated in Copernicus to create additional valid trajectories (at much smaller resolution) but without running a full Copernicus set of scans that can take days to run.

1.6 Results and Conclusions

We found that using cloud technologies increases computational efficiency for trajectory generation by 125% from current methods. This increased efficiency could be crucial near launch when potential changes to the nominal plan could derail the schedule entirely due to the time it takes to compute new trajectories. Our machine learning model assists in decreasing trajectory creation from hours/days to minutes/seconds with an overall accuracy of 98%. The use of Tableau allows for sharing of visualization dashboards and would be automatically updated upon generation of a new set of trajectory data. Additionally, decision-makers could access the visualizations via an interactive dashboard in the cloud. An additional outcome here is that the entire process can be automated such that as soon as a new set of trajectories is run in the cloud, a new dashboard is available for studying the new data's effect on the launch opportunities. Therefore, we conclude that cloud technologies, big data visualization, and machine learning techniques will benefit NASA's engineering team.

The following is a brief description of the remaining sections of this paper. Chapter 2 describes the technical concepts around the mission, factors influencing launch opportunities, and operational requirements that the selected launch date should meet. It also introduces the architecture for the cloud, visualization and machine learning proposals to address the data challenges. Chapter 3 explores data structure related to mission concepts. Chapter 4 details the processes and methods we followed to solve the problems. Chapter 5 shows the results of our solutions for cloud data management, visualization, and machine learning. Chapter 6 discusses ethical considerations in machine learning projects. Chapter 7 talks about our conclusions and recommended next steps for NASA.

2 Background and Tutorial

SLS and Orion will take off from NASA's spaceport at Kennedy Space Center in Florida. The SLS rocket will produce about 8.8 million pounds of thrust during liftoff and ascent to loft a vehicle weighing nearly six million pounds to orbit, and it will reach the period of greatest atmospheric force within ninety seconds. After jettisoning the boosters, service module panels, and launch abort system, the core stage engines will shut down and the core stage will separate from the spacecraft. As the spacecraft orbits the Earth, it will deploy its solar arrays. Next, the Interim Cryogenic Propulsion Stage (ICPS) will give Orion the push required to leave Earth's orbit and travel toward the Moon. From there, Orion will separate from the ICPS about two hours after launch.

2.1 Exploration Mission-1 Overview

EM-1 is the first in a series of increasingly complex missions. It is a mission where new technologies will be tested as a foundation to support further missions. Fig. 2 illustrates the full trajectory of the EM-1 mission.

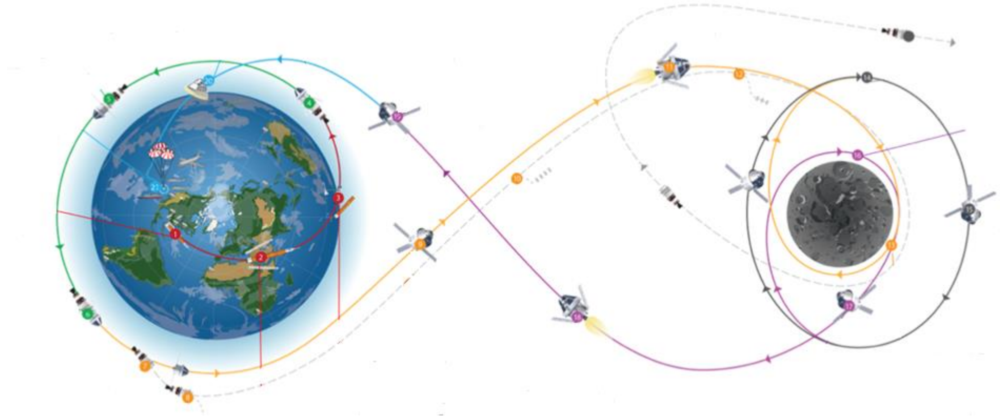


Fig. 2. This figure shows the EM-1 Mission overview broken out into segments showing the major/minor trajectory changing/correcting burns. (Source: NASA Exploration Mission Analysis Office)

The outbound trip to the Moon will take several days according to a predefined trajectory, which could be corrected as needed. Orion will fly about 62 miles (100 km) above the surface of the Moon, and then use the Moon's gravitational force to propel Orion into a distant retrograde (opposite the direction the Moon travels around the Earth) orbit about 40,000 miles (70,000 km) from the Moon. The spacecraft will stay in that orbit for approximately six days to collect data and allow mission controllers to assess the performance of the spacecraft.

For its return trip to earth, Orion will use another precisely timed engine firing service module in conjunction with the Moon's gravity to accelerate back toward Earth. This maneuver will set the spacecraft on its trajectory back toward Earth to enter our planet's atmosphere traveling at 25,000 mph (11 kilometers per second), producing temperatures of approximately 5,000 degrees Fahrenheit (2,760 degrees Celsius) – faster and hotter than Orion experienced during its 2014 flight test.

After more than three weeks and 1.3 million miles travelled, the mission will end with a test of Orion's capability to return safely to the Earth as the spacecraft makes a precision landing within the capabilities of the recovery ship off the coast of Baja California, Mexico [3].

2.2 Factors Affecting Launch Opportunities

Vehicle performance, launch location, launch azimuth along with earth-moon geometry and orbital mechanics all contribute to how many launch opportunities per month are available. Ideally, NASA looks for a continuous string of about a week of launch opportunities each month to give the best chance of not having to wait a full month until the next opportunity. If there is at least a one-minute window on a given day that is enough to plan a launch. Of course, it is preferable to have a few hours in which to launch. But even with no additional constraints levied on the mission than the agreed upon requirements, the opportunities are few. In fact, with only the aforementioned considerations, EM-1 will never have more than a three-hour launch window per day. Also, due to ground systems constraints only a two-hour window per day can be accommodated. Additional "desirements" have been added to the trade space that would make for more optimal conditions (e.g. lighting at launch, communication check outs, etc.). Each added criterion, however, only shortens the launch opportunities per month and the length of launch window per available day. Table 1 lists the EM-1 requirements, and Table 2 lists the EM-1 "desirements". Each of these plays a role in determining mission availability across a given year.

Table 1. Exploration Mission-1 List of Requirements

Target launch date
Maximum daily launch window of two hours
Variable launch azimuth—allows for constant post-TLI flight phase duration
ICPS will perform trans-lunar injection burn on first revolution
Flight duration between 21 and 42 days
Destination: Lunar Distant Retrograde Orbit (DRO)
Nominal and Abort Landing Sites
Lighting during Splashdown (for proper imagery conditions)
Orion solar arrays eclipse constraints

Table 2. Exploration Mission-1 List of “Desirements”

Daylight launch
Daylight ascent
RL-10 nozzle deploy lighting
Orion solar array deploy lighting
Orion/ICPS separation lighting
Crew module/Service Module separation lighting
Daylight for at least five hours post landing
Communication checkout
Secondary navigation checkout
Public Affairs Office imagery (requires good lighting)

To be successful in determining the available launch opportunities, data to calculate all of the above requirement and “desirement” effects on each day of the year will need to be extracted from the large trajectory data set.

2.3 Trajectory Data Flow

The JSC engineering trajectory team (EG) is tasked to find feasible earth-moon trajectories for each exploration mission. Using an in-house Fortran tool called Copernicus, the team feeds input parameters via a Python-coded tool called DAMOCLES that runs the Copernicus tool, takes the output and performs various post-processing tasks in order to make the final dataset more useful. It can also change the file format of the output as needed (see Fig. 3).

Currently, all EG tools used to manipulate Copernicus data are written in Python and utilize parallel processing techniques in order to utilize their blade server to compute trajectories. The computation of a precision lunar trajectory can only be done by numerical integration of the equations of motion, considering the oblate shape of the earth, solar perturbations, solar pressure, terminal attraction of the Moon, and ephemeris Moon data which gives the positions of the Moon’s orbital elements through time because they are constantly changing (primarily due to the perturbative effect of the sun) [6]. This requires intensive computation time and generates an incredible amount of data. The data generated with each run is stored as hdf5, csv, or json file formats in a simple server file structure to organize the data sets.

No attempt has been made to use a database, or cloud computing/storage. So far, the trajectory team’s consolidated trajectory runs or “scans” cover only the eight nominal launch cases. In mid-2019 EG begins to provide the final trajectory runs for EM-1 which will need to include all data against all Mission Design Matrix parameters. Since their data is in disparate, plaintext files, sharing, querying, and analyzing the data using statistical analysis or machine learning is next to impossible.

Additionally, these datasets are required by the Mission Analysis Office as well as the Flight Operations Directorate as inputs to their tools or in order to support decisions on what days are good to launch or how certain requirements must either be kept or dropped. These decisions affect SLS, Orion, Exploration Ground Systems, and NASA Headquarters where the Mission Management team makes the final call on when to launch. Our proposed architecture aims to bridge the gap between how engineering currently provides data and how organizations need the data in order to gain insights and visualize the resulting launch opportunities for EM-1 and future exploration missions.

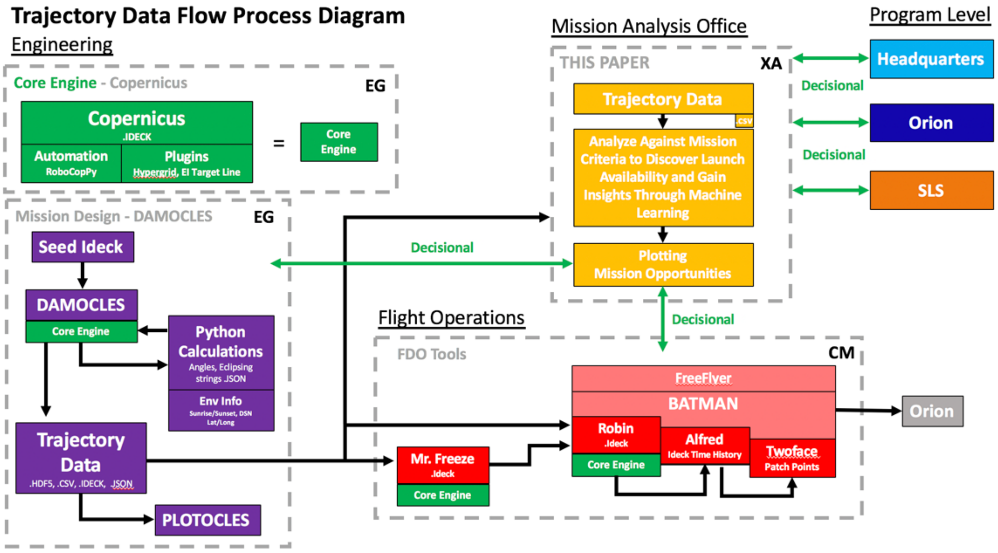


Fig. 3. This figure shows the trajectory data flow process. DAMOCLES seeds the core engine (Copernicus) and performs post-processing. This output data feeds the Mission Analysis Office where this paper's recommendations reside.

2.4 Architecture

The overall architecture discussed in this paper will reside within the Mission Planning Office section (Fig. 4). It consists of a repository of authoritative trajectory data that resides in the cloud and can easily be queried or studied. Additionally, an application we have developed that pairs down the raw trajectory data into a subset of data required to visualize launch availability will reside within the cloud. The application will pull out the dates/times across a time span (we will look at 2020) that meet all requirements as well as several highly desired conditions. Using the big data visualization tool, Tableau, connected to our cloud database, we will then visualize the mission availability and provide useful visualizations that will allow for easily choosing the best day to launch within a given month/year. Finally, the data stored in AWS will be pulled into a machine learning algorithm application to analyze the full data set.

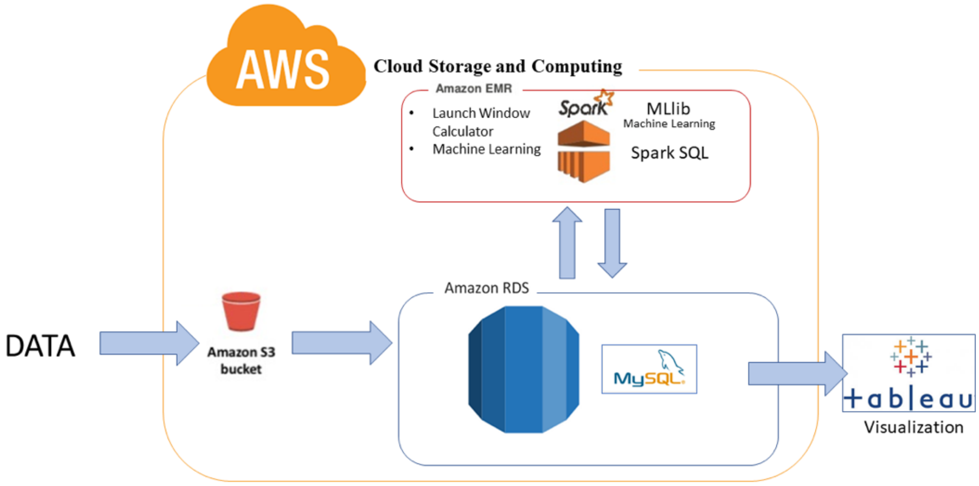


Fig. 4. Proposed Cloud Architecture for NASA's Mission Planning Office

Cloud Storage and Computing. Due to the sheer size of the data and the need for fast querying capability we chose to use Amazon Web Services (AWS) Elastic MapReduce (EMR) service offering. AWS EMR allows for the use of YARN with Spark. This cloud architecture (Fig. 5) accommodates elasticity of a database that will grow over time and allows the ability to perform machine learning tasks with the data co-located.

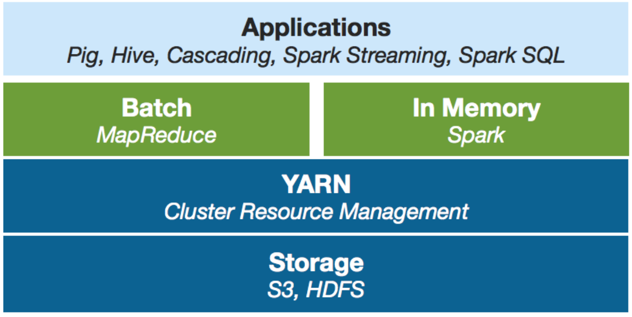

Applications
Pig, Hive, Cascading, Spark Streaming, Spark SQL
Batch
MapReduce
In Memory
Spark
YARN
Cluster Resource Management
Storage
S3, HDFS

Fig. 5. Shows the architecture to support our application using AWS S3 storage with a Hadoop Distributed File System along with MapReduce and Spark to help querying speed.

Launch Window Calculator. This application will consist of a python module that will reside in the cloud using YARN and Spark to assist in performing simple mathematical calculations and queries to extract the required data for determining launch opportunities and the length of the daily launch windows for a launch season of one year. It will provide some computation to calculate solar array eclipse times and lighting at particular mission times. In the end it will be an organized data set that will be placed in our database in order to visualize the launch opportunities.

Visualization in Tableau. With the complex data set now organized in a more optimal way that accommodates high performance querying, we can now link the data set to the commercially available visualization tool called Tableau. This will not only save time in developing custom visualization tools in python but will alleviate the trajectory team from having to respond to simple plotting requests on their data sets. More time spent on trajectory analysis means more time spent mitigating issues of meeting mission criteria. We expect to use this tool as not only our main source of launch window visualization but also a means by which the exploration mission community can plot variables against each other on their own time.

Machine Learning. Machine learning is a fresh concept at Johnson Space Center. Our hope for this paper is to not only position the engineering team to be able to better handle big data, but to also allow them to see what is in the realm of possibility using machine learning techniques and make the team and their suite of mathematical tools more efficient. Therefore, one component of the application will be pulling from the shared data set to perform machine learning tasks.

2.5 Machine Learning Methods

The data set consists of observed values for every COE element and elapsed times in every scan. We cannot assume linearity, so we will use regression algorithms that do not depend on linear structure, like K-Nearest Neighbors regression (KNN), Random Forest, Gradient Boosting, and Bayesian Ridge Regression (BRR). We will measure and compare accuracy between these models.

KNN estimates the regression value as the average of the K closest neighbors in the training responses. We will adjust for the number of K, and measure the bias-variance tradeoff to get to a resultant model [9].

The random forest regressor builds a number of decision trees on “boostrapped” training samples and averages their results. That may overcome the complexity of decision trees.

Extremely randomized trees is a tree-based ensemble method that randomizes both attribute and cut-point choice while splitting a tree node. In an extreme case, it builds randomized trees whose structures are independent from the learning sample [10].

Bayesian Ridge Regression (BRR), also known as Bayesian Linear Regression, assumes a normal distribution of the errors from a regression model, a prior distribution on the variables, and a posterior probability distribution from the model parameters [11].

Gradient boosting finds the function which best approximates the data, by taking numerous simple functions, and adding them together. While stochastic gradient descent (SGD) trains a complex model to find the best parameters, Gradient boosting trains the sum (ensemble) of simple models.

In the case of Gradient Boosted Trees, it is a gradient boosting model where the function is a decision tree. It is an additive model where trees are grown sequentially using learned information from previous grown trees, allowing the optimization of a loss function [9, 12, 13].

Extreme Gradient Boost (XGBoost) is an implementation of gradient boosted trees designed recently for speed and performance. XGBoost considers the distribution of features across all data points in a leaf and uses this information to reduce the search space of possible feature splits [14].

3 Data

First, the initial conditions are entered manually into the DAMOCLES tool. When DAMOCLES runs, it opens the Copernicus tool and performs a one-year trajectory scan given the inputted parameter set. In the case of nominal missions, this will generate a few hundred gigabytes of data artifacts. Once the abort cases are run, the total amount of data generated will come close to 90 terabytes. To be able to employ our machine learning tools we require the use of a summary file. In the case of the nominal missions, the summary file is about 100 megabytes and is one of the artifacts generated when DAMOCLES is run. It is a post-processed summary and includes only the minutes per day that meet the baseline requirements.

The dataset contains 18,980 rows and 480 columns. Each row represents an opportunity to launch at a certain date and time specified at column "Init Epoch (TDB)". Launch epochs are the feasible moments to successfully launch a spacecraft and start the mission. The EM-1 mission duration may be between 26 and 42 days in order to accommodate lighted landing requirements. Initial Epoch is measured in the astronomical time scale Barycentric Dynamical Time (or TDB, which is different by less than 0.002 seconds compared to the Terrestrial Time scale UTC). The launch epoch is then calculated from TDB to Eastern Standard Time for launching from Florida.

3.1 Summary File Rows

Each row or launch opportunity was calculated by NASA's DAMOCLES tool and saved to the filename logged at column "Directory". DAMOCLES computes the orbital elements and analyzes the feasibility for each possible Init Epoch. The system addresses the required constraints of performance (thrust requirements given the fuel capabilities and the earth-moon geometry), and lighting conditions at the selected Moon location during the expected arrival time. There are 18,980 opportunities that meet the required criteria between the selected dates of this particular scan. They represent the nominal cases only (assumes nothing goes wrong).

The first row in the dataset represents the first launch opportunity to start the mission on November 23, 2019 at 9:22am UTC, and the last one can start on December 14, 2020 at 3:17pm. Since the time resolution unit is one minute, there are 18,980 possible one-minute opportunities to launch, that spans 157 days within this date range; for instance, Nov 28, 2019 has 42 opportunities within that day, but the next day Nov 29 2019 does not have any. The more launch opportunities in a day, the more attractive the day (e.g. Oct 17, 2020 has 311 opportunities compared to a single opportunity on Dec 26, 2019). Also, the more consecutive days with feasible launch opportunities, the more attractive the date range because it can better mitigate operational risks. See the complete number of opportunities by day in Fig. 6 (only days with at least one launch opportunity).

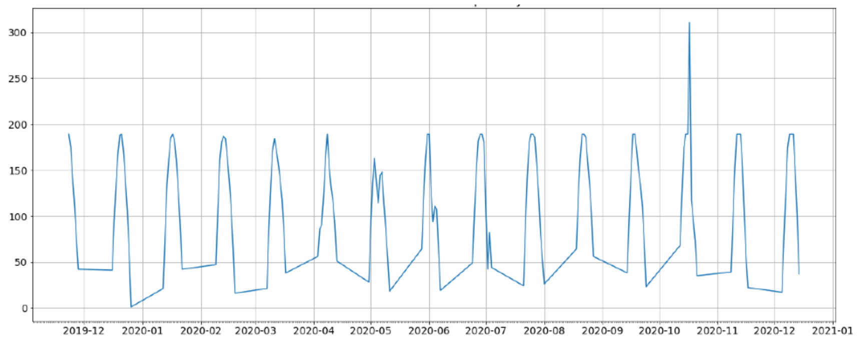


Fig. 6. Number of one-minute opportunities to launch. These make up the daily launch window.

Fig. 6 shows the number of one-minute launch opportunities for EM-1 (the daily launch window). Each month has a short launch window of a few days that typically peaks near 200 minutes. Unfortunately, Exploration Ground Systems requires a window of no more than 120 minutes, so the visualization tools assist with choosing the best 120 minutes to pick from.

3.2 Summary File Columns

Columns represent all the attributes and possible characteristics of the projected mission, assuming the mission starts at the given value of launch epoch. Column Launch Epoch uniquely identifies each opportunity in a data row.

From the 480 dataset columns, the explanatory variables are those 19 features that represent the mean elapsed times (MET) during certain points along the mission, for instance, 'TLI Start MET' is the number of days from the launch epoch until the Trans-Lunar Injection begins. See Table 3 for a complete description of explanatory variables used in the model.

Table 3. List of explanatory variables. All events are measured in number of days since launch epoch.

Name	Description
Init DRO MET (days)	Initial Distant Retrograde Orbit
CoreSep MET (days)	Core separation burn
PRM Start MET (days)	Perigee Raise Maneuver start time
PRM SteadyState MET (days)	Perigee Raise Maneuver steady state
PRM End MET (days)	Perigee Raise Maneuver end time
TLI Start MET (days)	Trans Lunar Injection burn
TLI SteadyState MET (days)	Trans Lunar Injection steady state
TLI End MET (days)	Trans Lunar Injection end time
ICPS-Orion SS MET (days)	Interim Cryogenic Propulsion Stage Separation
TIP MET (days)	Target Interface Point
USS MET (days)	Upper Stage Separation
OCO MET (days)	OMS Check-Out burn
OPF MET (days)	Outbound Powered Fly-by

Outbound Waypoint MET (days)	Outbound Waypoint time
DRI MET (days)	Distant Retrograde Insertion burn
DRD MET (days)	Distant Retrograde Departure burn
RPF MET (days)	Return Power Fly-by
Return Waypoint MET (days)	Return Waypoint time
EI MET (days)	Entry Interface time

The target variables are the classical orbital elements (COEs) through the mission segments (stages). COEs determine the orbit and the spacecraft's location in the orbit at any time. The set of COEs during the entire mission represents the key points that define the trajectory and allow Copernicus to fill in all the locations during the coasting phases.

One of this paper's purposes is to predict new trajectories based on previous calculated trajectories and save computing time using DAMOCLES/Copernicus. In the Machine Learning context, the goal is to regress COE variables from the explanatory MET variables at a time resolution shift of 30 seconds.

This is the list of COEs used in the dataset:

- SMA (semi-major axis, measured in Km)
- Ecc (eccentricity, no units are used)
- Incl (inclination measured in degrees)
- Raan (right ascension of the ascending node, aka longitude of ascending node, measured in degrees)
- AOP (argument of periapsis, measured in degrees)
- TrAnom (true anomaly, measured in degrees)

See a graphical representation of COEs at Fig. 7.

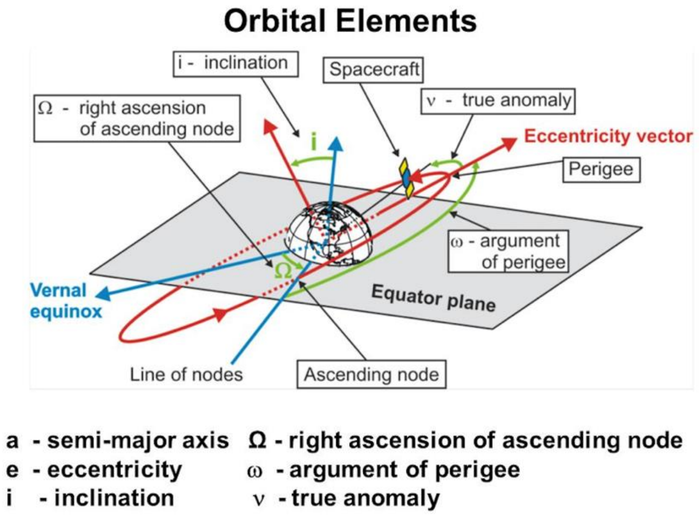


Fig. 7. Classical Orbital Elements (COEs). (Source: https://www.slideserve.com/bran/space-engineering-i-part-i-where-are-we)

This is the list of segments used in the dataset. It includes measurements from before and after the main events during the mission:

- Pre-PRM
- Post-PRM
- Pre-TLI
- Post-TLI
- Pre-OCO
- Post-OCO
- Pre-OPF
- Post-OPF
- Pre-DRI
- Post-DRI
- Pre-DRD
- Post-DRD
- Pre-RPF
- Post-RPF

From the 480 dataset columns, target variables are the 84 features that combine the six COEs multiplied by the 14 segments. For instance, "Pre-PRM TrAnom (deg) [J2000_Earth]" represents the measurement for orbital element True Anomaly (TrAnom) in segment Pre-PRM; "Post-RPF SMA (km) [J2000_Moon]" is the orbital element SMA in kilometers at the Post-RPF stage.

The combined six orbital elements define the trajectory at every stage. The combined 14 trajectories at each stage represent the complete mission trajectory for a specific initial epoch.

In summary, we have two columns that identify the initial and launch epoch in the summary source file from DAMOCLES, 19 explanatory variables from Mission Elapsed Times (MET) columns, 84 target variables from measurements of orbital elements at mission stages, and the remaining columns are not used for the purposes of this paper. We then applied supervised regression analysis to all COE variables on an individual basis assuming independence among them.

3.3 PyMAT Trajectory Table for Data Visualization

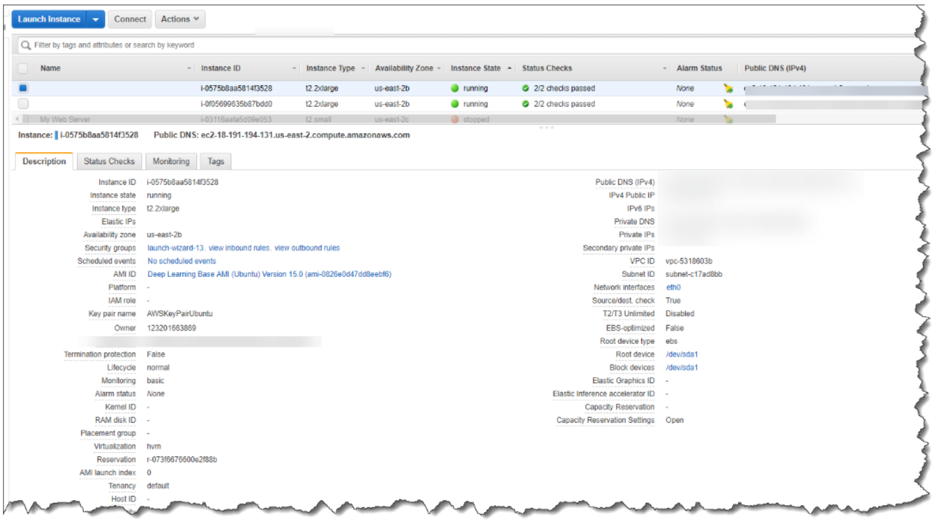
Much like the summary file pairs down the full set of outputs into one file, the trajectory table further pairs down the summary file data into the data required to visualize the launch opportunities and the influence of the requirements and "desirements" on different launch days. This trajectory table is created with the PyMAT tool discussed later in Chapter 4. Data from this table is used to create an interactive calendar visualization that provides both total launch opportunities given a set of requirements as well as a quick visual for launch windows per month and the ability to quickly see if some of the "desirements" can also be achieved.

4 Methods

The proposed strategy enhances the EM-1 status quo by creating a tool set with new methods for data storage and manipulation, visualization, and machine learning. We will discuss each in this section.

4.1 Cloud Storage and Computing

We used AWS as a platform to implement this solution. We used a cluster of two t2.2xlarge nodes with Linux/Ubuntu 18.04 with the following characteristics (Fig. 8):



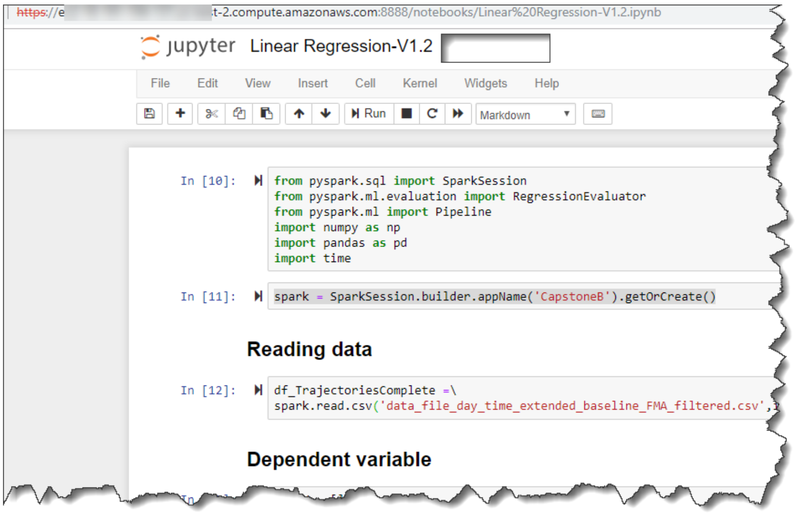
	Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status	Public DNS (IPv4)
☑		i-0575b8aa5814f3528	t2.2xlarge	us-east-2b	● running	✓ 2/2 checks passed	None	[redacted]
□		i-0f05699635b87bdd0	t2.2xlarge	us-east-2b	● running	✓ 2/2 checks passed	None	[redacted]
□	My Web Server	i-03116aafa5d09e053	t2.small	us-east-2c	● stopped		None	

Instance: i-0575b8aa5814f3528 Public DNS: ec2-18-191-194-131.us-east-2.compute.amazonaws.com

Instance ID	i-0575b8aa5814f3528	Public DNS (IPv4)	[redacted]
Instance state	running	IPv4 Public IP	
Instance type	t2.2xlarge	IPv6 IPs	
Elastic IPs		Private DNS	
Availability zone	us-east-2b	Private IPs	
Security groups	launch-wizard-13. view inbound rules. view outbound rules	Secondary private IPs	
Scheduled events	No scheduled events	VPC ID	vpc-5318603b
AMI ID	Deep Learning Base AMI (Ubuntu) Version 15.0 (ami-0826e0d47dd8eebf6)	Subnet ID	subnet-c17ad8bb
Platform	-	Network interfaces	eth0
IAM role	-	Source/dest. check	True
Key pair name	AWSKeyPairUbuntu	T2/T3 Unlimited	Disabled
Owner	123201663869	EBS-optimized	False
[redacted]		Root device type	ebs
Termination protection	False	Root device	/dev/sda1
Lifecycle	normal	Block devices	/dev/sda1
Monitoring	basic	Elastic Graphics ID	-
Alarm status	None	Elastic Inference accelerator ID	-
Kernel ID	-	Capacity Reservation	-
RAM disk ID	-	Capacity Reservation Settings	Open
Placement group	-		
Virtualization	hvm		
Reservation	r-073f6676600e2f88b		
AMI launch index	0		
Tenancy	default		
Host ID	-		

Fig. 8. Characteristics of AWS EC2 instance

We use PuTTY 0.70 to establish the connection to the Cloud and run Jupyter Notebook, then Jupyter Notebook is emulated through a local browser as shown below (Fig. 9).



~~https~~://e[redacted]t-2.compute.amazonaws.com:8888/notebooks/Linear%20Regression-V1.2.ipynb

In [10]:

```
from pyspark.sql import SparkSession
from pyspark.ml.evaluation import RegressionEvaluator
from pyspark.ml import Pipeline
import numpy as np
import pandas as pd
import time
```

In [11]:

```
spark = SparkSession.builder.appName('CapstoneB').getOrCreate()
```

Reading data

In [12]:

```
df_TrajectoriesComplete =\
spark.read.csv('data_file_day_time_extended_baseline_FMA_filtered.csv',
```

Dependent variable

Fig. 9. Running Jupyter Notebook in AWS from a local machine

4.2 Visualization

Table 1 and 2 above list all the requirements and "desirements" for the EM-1 mission. Our visualization of launch opportunities solution needs to easily show in a calendar format all the days available to launch in a given year that meet these requirements. Additionally, we want to enhance the visualization by showing where amongst these launch opportunities we can find days that also meet at least our highest priority "desirements" such as having lighting during launch and the proper sun angle to achieve a backup navigation system checkout. We would also like to see the influence these "desirements" have on the overall number of launch opportunities per year. This kind of visualization can assist with not only narrowing the choices of when the best time to launch is, but can help with requirements development to quickly see the influence certain requirements or "desirements" are having on our overall ability to launch, thus helping NASA properly categorize and rank them.

Conditional formatting is applied to the days with data pulled from the PyMAT-created "Trajectory Table". PyMAT imports a full trajectory "Summary Report" and creates an abridged subset of this data that can be used to visualize and analyze available launch opportunities.

A Tableau server is utilized to share a dashboard across the web with customers that do not have Tableau Desktop installed locally. Due to the interactive nature of the visualization, multiple conclusions can be drawn from a single visualization whenever the user would like to see it. As data matures, or requirements change, a new data set can be plugged into the PyMAT tool in order to generate a new trajectory table and hence, an updated launch opportunity visualization dashboard.

4.3 Applying Machine Learning

Now that we have the architecture in place to handle both the storage and the manipulation of large, complex data sets, we are free to explore ways to decrease the computational time required to find working trajectories. As discussed earlier, there are many factors required to compute a successful earth-moon trajectory. At the prescribed resolution of one minute, that is 525,600 trajectory scans per case (one launch window per minute for 365 days). With 8 nominal cases that equates to 4.2M scans. However, there are additional 56 off-nominal cases as part of the Mission Design Matrix. Each of these scans is separated into segments which represent the start and end positions of each burn (Fig. 2).

Each segment is calculated separately and then strung together in a final step to represent the entire earth-moon trajectory or off-nominal trajectory being run. Because this generates such a large amount of data in the mission planning stages, often only one scan is performed per day for the start of the launch window. Therefore, a "continuation method" is used in which the start of the next scan is initialized using the end of the previous scan. This initial guess is often not ideal and leads to increased computation time in trying to resolve the current trajectory being calculated.

Using machine learning techniques, we are in essence increasing the scan resolution by providing viable trajectories that can be used to initialize future scans and decrease overall computational time. Additionally, if in the future, a quick set of trajectories are required at a greater resolution than what the engineering team has data for (30 second rather than one minute), these machine learning techniques can be employed to offer quick solutions.

Classic Orbital Elements (COEs) can be used to describe the size, shape, and orientation of an orbit and the location of a spacecraft in that orbit [6]. By training a model on the COEs perifocal distance, eccentricity, right ascension, inclination, argument of perigee, and true anomaly as well as mission-elapsed time we can generate valid trajectories in between those we have data for thus improving the resolution of the scan. We then validate these trajectories by comparing against a solved data set.

4.4 Exploratory Data Analysis

In section 3.2 we organized the dataset as 19 explanatory variables from Mission Elapsed Times (MET) columns, and 84 target variables from measurements of orbital elements at mission stages. We want to train a model from known scans to regress COE target variables at different time resolutions.

We plotted the histogram of target variables to see how normal the data distribution was, and study the relationship between explanatory and target variables.

Fig. 10 shows histograms for two sample target variables: Post-PRM SMA and Post-PRM Incl.

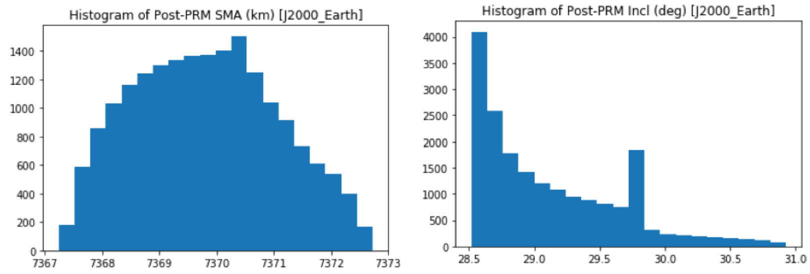


Fig. 10. Histograms of orbital elements SMA (left) and Incl (right) at stage Post-PRM.

Fig. 11 shows the relationship between the explanatory variable Init DRD MET and the target variable Pre-PRM SMA (left), and the relationship between other explanatory variable DRD MET to the same target variable Pfre-PRM SMA (right).

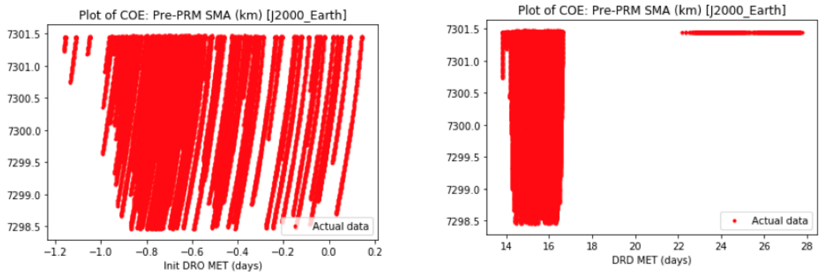


Fig. 11. Relationship between explanatory variables Init DRD MET and DRD MET to target variable Pre-PRM SMA.

Fig. 12 shows the relationship between explanatory variable Init DRD MET to other explanatory variable CoreSep MET (left), and the relationship between Init DRD MET to other explanatory variable Outbound Waypoint MET (right).

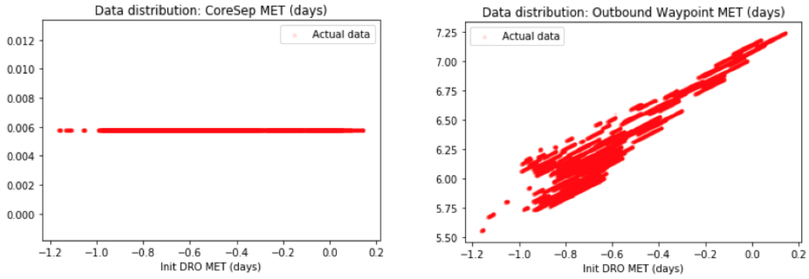


Fig. 12. Relationship between the explanatory variables Init DRD MET to CoreSep MET and to Outbound Waypoint MET.

From the preliminary observation, we can see several target variables highly skewed (e.g. Fig. 10 right), other variables seem not to be affected at certain stages (e.g. Fig. 11 right and Fig. 12 left), and also, we noticed correlation between some explanatory variables (e.g. Fig. 12 right).

The most significant findings were:

- Feature Pre-PRM TrAnom has only two possible values (-180 and 180), and so it is a candidate to be analyzed as a classification problem rather than as a regression (see Fig. 13 left). Another observation is that there is no visual distinction between when this variable switches from one value to the other (see Fig. 13 right).

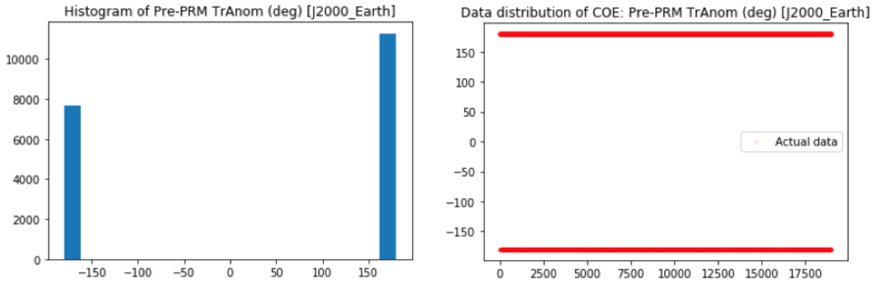


Fig. 13. Histogram and data distribution of target feature Pre-PRM TrAnom.

- Feature Post-DRD SMA has the longest skewness among all target variables. Its relative size of the left tail (skew = 94.69 in Fig. 14 left) led to the need of applying an inverse transformation to get a more manageable skew of 3.49 before the regression analysis (Fig. 14 right).

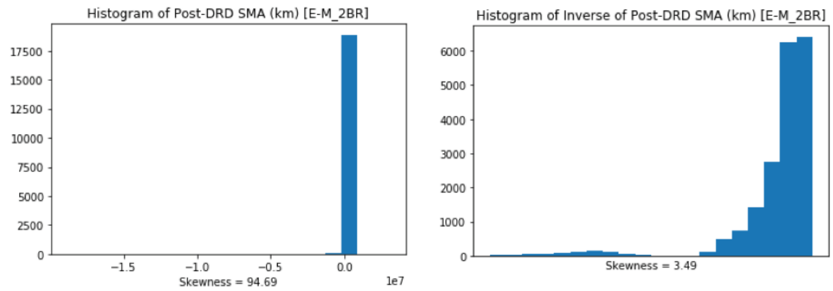


Fig. 14 Histogram of features Post-DRD SMA and Post-DRD Ecc.

4.5 Detailed Machine Learning Process

The first finding from the Exploratory Data Analysis revealed that feature “Pre-PRM TrAnom” (COE True Anomaly at stage Pre-PRM) is actually a binary variable, so we will use a classification analysis for handling it. The remaining 83 features will be managed as continuous variables with regression analysis. For the purpose of this paper, we will primarily focus on the regression analysis.

We have been through an iterative process trying different approaches to determine which algorithm returns the best results measured with R-squared (R2) scores as a goodness of fit [15] of regression. We also saved mean absolute error (MAE) during the process. Then the best fitting algorithm was associated for that specific feature for a later use to predict new trajectory data.

The regression algorithms used were: Bayesian Ridge, K-Nearest Neighbors, Extreme Gradient Boosting, Lasso, AdaBoost, Gradient Tree Boosting (or Gradient Boosted Trees), Extremely Randomized Trees, Random Forest, Stochastic Gradient Descent, Support Vector Machines, and Convolutional Neural Networks.

In the particular case of classification analysis, the classification algorithms used were: Gaussian Naive Bayes, K-Nearest Neighbors, Extreme Gradient Boosting, Logistic Regression, Random Forest, Support Vector Machines, AdaBoost, Gradient Tree Boosting, Stochastic Gradient Descent, Multi-Layer Perceptron, Linear Support Vector, and Convolutional Neural Networks. The performance score was measured by accuracy, rather than R2.

All of these algorithms were invoked through the scikit-learn machine learning library for Python, and in the case of neural networks, we used Keras deep learning library integrated to scikit-learn [16], to facilitate a single execution framework that browses through several configurable algorithms.

All the explanatory variables are measured in the same scale (number of days); however, we used a standard scaler to normalize data in order to improve the performance during the fit execution of each algorithm [17].

This is the process to generate the model:

1- Loop for each target variable in the list:
    a. If there is severe skewness, then: transform data using invert or log.
    b. If feature is Pre-PRM TrAnom, then transform data into a binary variable for classification analysis.

```
        c. Separate train and test datasets (with 20% data for testing).
        d. Scale train and test datasets.
        e. Loop for each algorithm in the list:
              i. Use GridSearchCV with 5 folds [18] to fit training data according
                 to current algorithm.
             ii. If R2 score is greater than current best score for this feature:
                    1. Save results (algorithm, parameters, R2, MAE, response
                       time, data transformation needed) as the best model.
                    2. Make R2 as the current best score.
        f. Repeat for another algorithm until R2 is greater than the minimum for early
           stop (.975) or until list completion.
  2- Repeat for next variable until list completion.
  3- Append results to a logfile.
```

This is the process to calculate new trajectories:

```
  1- Create a new trajectories dataset, copying column Launch Epoch, and explanatory
     MET variables from the initial dataset.
  2- Add 30 seconds to each column in the new dataset.
  3- Loop for each target variable in the. list:
        a. Load the best model associated for this specific variable from the results
           logfile.
        b. Transform data if needed (inverse/log or binary variable).
        c. Separate train and test datasets (with 20% data for testing).
        d. Scale train and the New Trajectories datasets.
        e. Use GridSearchCV with 5 folds to fit training data according to the best
           model algorithm.
        f. Predict target variable values using the fitted model on the new dataset.
        g. Transform back data if needed.
        h. Save target variable values into the new Trajectories dataset.
  4- Save Trajectories dataset to a CSV file.
  5- Save results to a logfile.
```

Alternative process using PySpark. In parallel, we also did the same regression and classification analysis using PySpark in order to have an independent analysis, and compare the results with Python+scikit-learn. Also, PySpark enables the possibility of using machine learning in a native cloud-based platform, in case we need to scale to higher volumes of data.

The regression algorithms used in PySpark were: Generalized Linear Regression, Linear Regression, Random Forest, Gradient-Boosted Tree Regression, and Decision Trees. The evaluation unit is also R Squared. The classification algorithms used in PySpark were: Decision Trees, Random Forest, and Gradient-Boosted Trees. The evaluation unit is also accuracy.

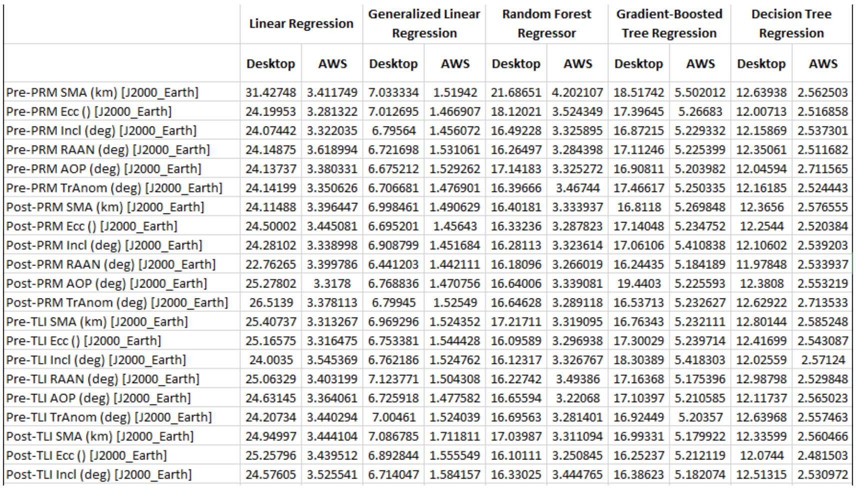
5 Results

This section discusses results in cloud computing, visualization and machine learning.

5.1 Results of Cloud Computing and Data Storage

In order to verify the advantages of using AWS and PySpark a preliminary test was performed running regression models with PySpark in a local machine, and later the same models in AWS. We compared the timing of training the data for every method for each feature mentioned above.

As mentioned above the regression algorithms used were: Linear Regression, Generalized Linear Regression, Random Forest Regressor, Gradient-Boosted Tree Regression and Decision Tree Regression. The table below shows only the first 20 variables with timing in seconds for the local Desktop and for the AWS cluster. It is clear that we have much better performance in AWS, with some cases ten times faster in AWS (Fig. 15).

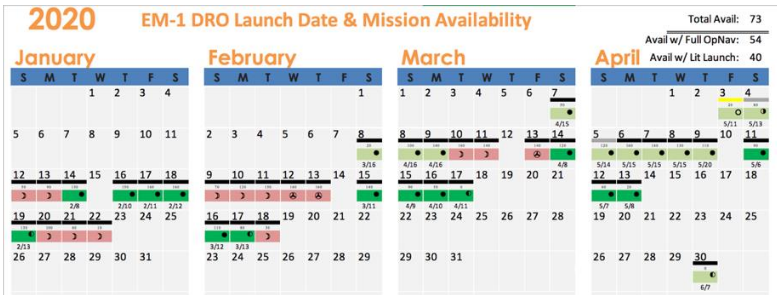


	Linear Regression		Generalized Linear Regression		Random Forest Regressor		Gradient-Boosted Tree Regression		Decision Tree Regression	
	Desktop	AWS	Desktop	AWS	Desktop	AWS	Desktop	AWS	Desktop	AWS
Pre-PRM SMA (km) [J2000_Earth]	31.42748	3.411749	7.033334	1.51942	21.68651	4.202107	18.51742	5.502012	12.63938	2.562503
Pre-PRM Ecc () [J2000_Earth]	24.19953	3.281322	7.012695	1.466907	18.12021	3.524349	17.39645	5.26683	12.00713	2.516858
Pre-PRM Incl (deg) [J2000_Earth]	24.07442	3.322035	6.79564	1.456072	16.49228	3.325895	16.87215	5.229332	12.15869	2.537301
Pre-PRM RAAN (deg) [J2000_Earth]	24.14875	3.618994	6.721698	1.531061	16.26497	3.284398	17.11246	5.225399	12.35061	2.511682
Pre-PRM AOP (deg) [J2000_Earth]	24.13737	3.380331	6.675212	1.529262	17.14183	3.325272	16.90811	5.203982	12.04594	2.711565
Pre-PRM TrAnom (deg) [J2000_Earth]	24.14199	3.350626	6.706681	1.476901	16.39666	3.46744	17.46617	5.250335	12.16185	2.524443
Post-PRM SMA (km) [J2000_Earth]	24.11488	3.396447	6.998461	1.490629	16.40181	3.333937	16.8118	5.269848	12.3656	2.576555
Post-PRM Ecc () [J2000_Earth]	24.50002	3.445081	6.695201	1.45643	16.33236	3.287823	17.14048	5.234752	12.2544	2.520384
Post-PRM Incl (deg) [J2000_Earth]	24.28102	3.338998	6.908799	1.451684	16.28113	3.323614	17.06106	5.410838	12.10602	2.539203
Post-PRM RAAN (deg) [J2000_Earth]	22.76265	3.399786	6.441203	1.442111	16.18096	3.266019	16.24435	5.184189	11.97848	2.533937
Post-PRM AOP (deg) [J2000_Earth]	25.27802	3.3178	6.768836	1.470756	16.64006	3.339081	19.4403	5.225593	12.3808	2.553219
Post-PRM TrAnom (deg) [J2000_Earth]	26.5139	3.378113	6.79945	1.52549	16.64628	3.289118	16.53713	5.232627	12.62922	2.713533
Pre-TLI SMA (km) [J2000_Earth]	25.40737	3.313267	6.969296	1.524352	17.21711	3.319095	16.76343	5.232111	12.80144	2.585248
Pre-TLI Ecc () [J2000_Earth]	25.16575	3.316475	6.753381	1.544428	16.09589	3.296938	17.30029	5.239714	12.41699	2.543087
Pre-TLI Incl (deg) [J2000_Earth]	24.0035	3.545369	6.762186	1.524762	16.12317	3.326767	18.30389	5.418303	12.02559	2.57124
Pre-TLI RAAN (deg) [J2000_Earth]	25.06329	3.403199	7.123771	1.504308	16.22742	3.49386	17.16368	5.175396	12.98798	2.529848
Pre-TLI AOP (deg) [J2000_Earth]	24.63145	3.364061	6.725918	1.477582	16.65594	3.22068	17.10397	5.210585	12.11737	2.565023
Pre-TLI TrAnom (deg) [J2000_Earth]	24.20734	3.440294	7.00461	1.524039	16.69563	3.281401	16.92449	5.20357	12.63968	2.557463
Post-TLI SMA (km) [J2000_Earth]	24.94997	3.444104	7.086785	1.711811	17.03987	3.311094	16.99331	5.179922	12.33599	2.560466
Post-TLI Ecc () [J2000_Earth]	25.25796	3.439512	6.892844	1.555549	16.10111	3.250845	16.25237	5.212119	12.0744	2.481503
Post-TLI Incl (deg) [J2000_Earth]	24.57605	3.525541	6.714047	1.584157	16.33025	3.444765	16.38623	5.182074	12.51315	2.530972

Fig. 15. Results comparing times of training data in a local computer vs AWS

5.2 Results of Data Visualization

First a prototype was created in MS Excel to show a proof of concept. In this prototype a representative calendar is created using formatted rows and columns to represent each day in a monthly calendar view. Data is pulled from the trajectory table and conditional formatting applied to show available launch opportunities in addition to highly ranked "desirements" that can be achieved on these available days. Short and long duration missions are represented by different shades of green. Launch lighting is represented by a yellow strip above each day, and the OpNav requirement is represented by a set of Harvey balls. Any changes to the overall requirements/"desirements" were handled via multiple tabs with their own calendar representation. The total days available for the year are calculated at the top of the spreadsheet, depending on the requirements selected for that particular analysis spreadsheet (tab in Excel Fig. 16).



2020 EM-1 DRO Launch Date & Mission Availability

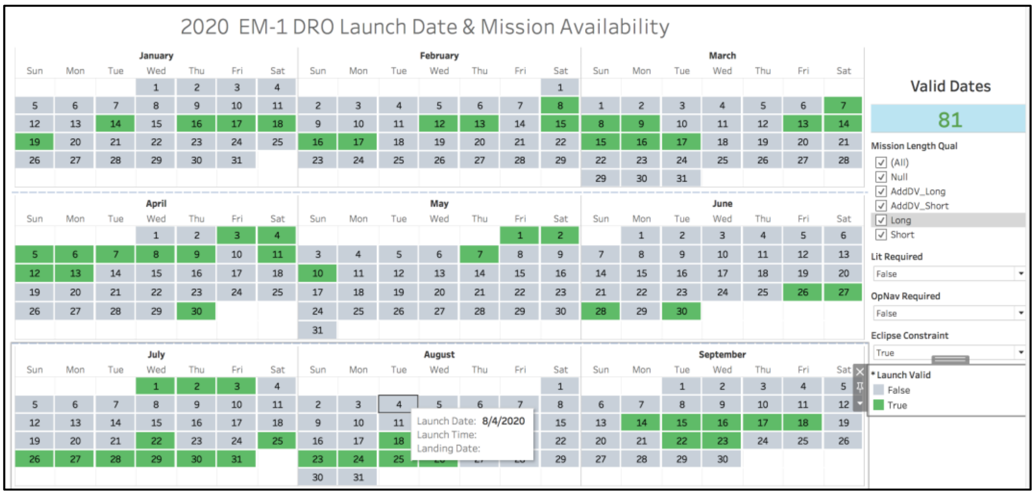
Total Avail: 73

Avail w/ Full OpNav: 54

Avail w/ Lit Launch: 40

Fig. 16. MS Excel Prototype of Launch Opportunities

Similarly, we ported over this method to Tableau, however we were able to use a built-in calendar visualization rather than building each day by hand as we did in MS Excel. We created an interactive dashboard rather than using multiple tabs and were able to make it such that a user could toggle requirements on and off and see the effect on each month's launch opportunities as well as the total opportunities for the year (Fig. 17).



2020 EM-1 DRO Launch Date & Mission Availability

Valid Dates

81

Mission Length Qual

☑ (All)
☑ Null
☑ AddDV_Long
☑ AddDV_Short
☑ Long
☑ Short

Lit Required: False

OpNav Required: False

Eclipse Constraint: True

Launch Valid

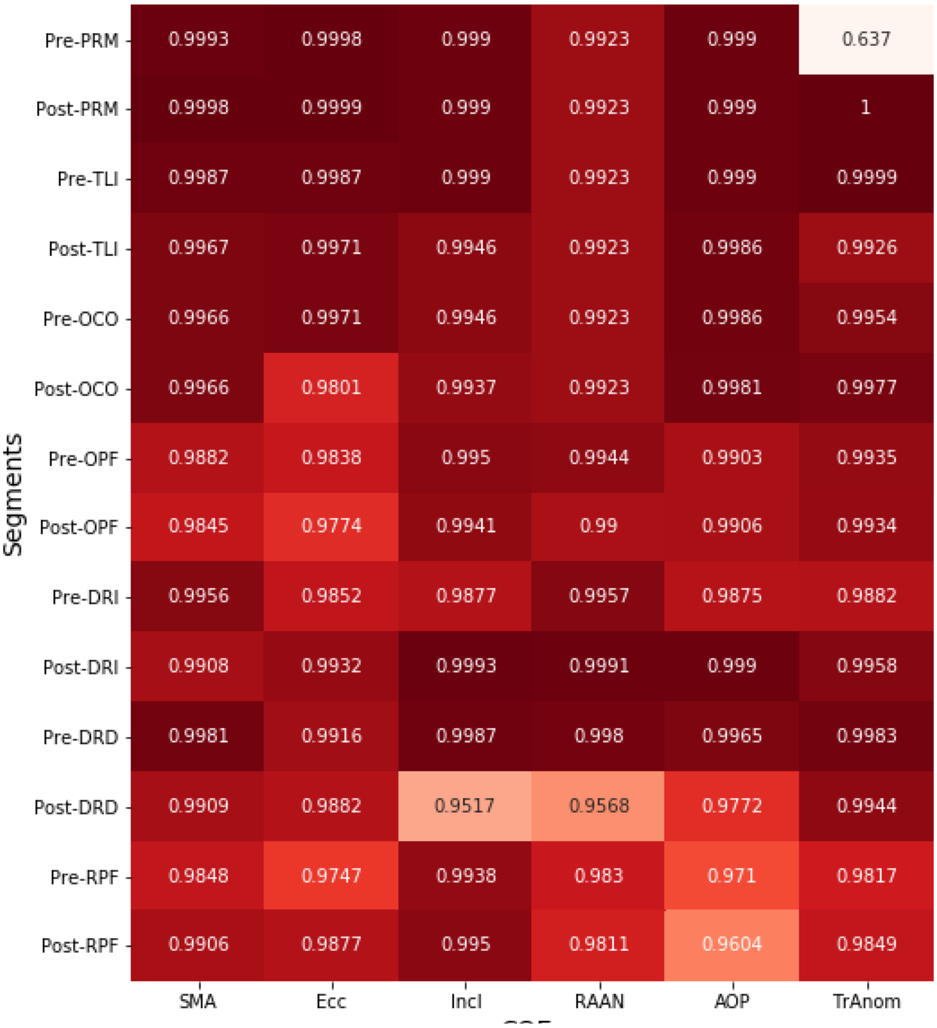
- False
- 🟩 True

Launch Date: 8/4/2020
Launch Time:
Landing Date:

Fig. 17. Tableau Launch Opportunity Interactive Dashboard

5.3 Results of Machine Learning Analysis using Python+sklearn

Accuracy. The exercise was successful in terms of fitting a model that produced a final average R2 around 0.98. Fig. 18 summarizes the R2 scores for all COE variables in all segments.



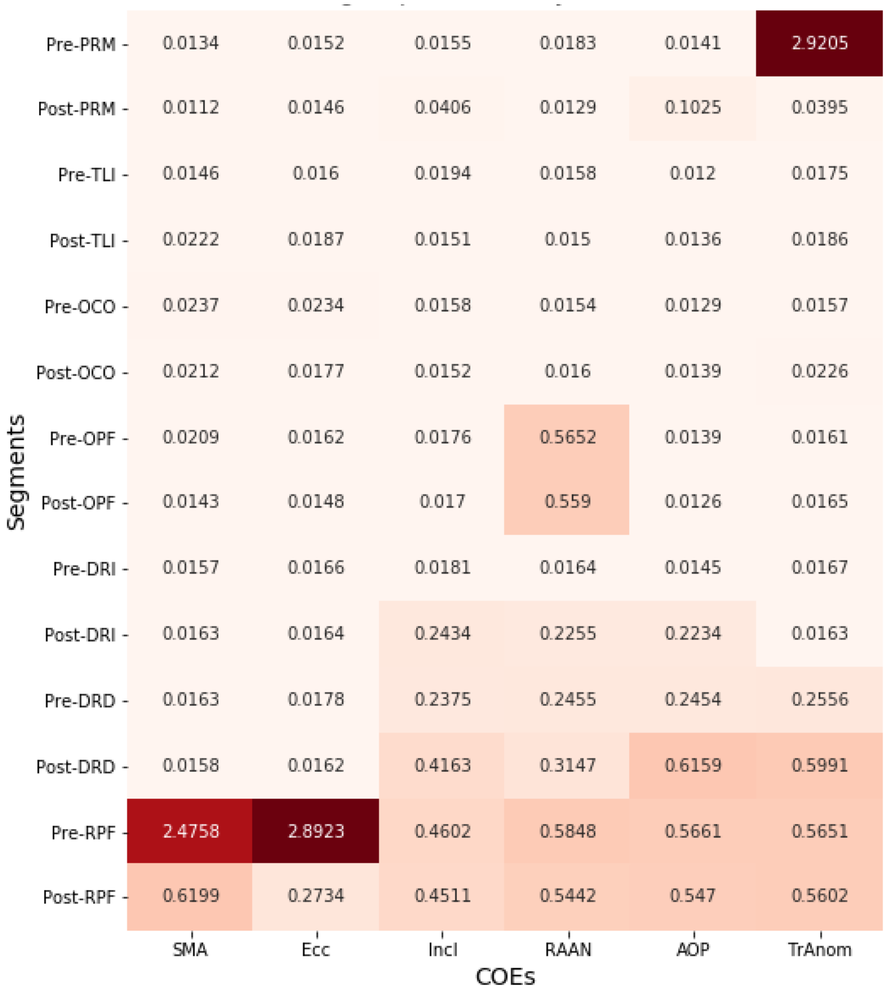
	SMA	Ecc	Incl	RAAN	AOP	TrAnom
Pre-PRM	0.9993	0.9998	0.999	0.9923	0.999	0.637
Post-PRM	0.9998	0.9999	0.999	0.9923	0.999	1
Pre-TLI	0.9987	0.9987	0.999	0.9923	0.999	0.9999
Post-TLI	0.9967	0.9971	0.9946	0.9923	0.9986	0.9926
Pre-OCO	0.9966	0.9971	0.9946	0.9923	0.9986	0.9954
Post-OCO	0.9966	0.9801	0.9937	0.9923	0.9981	0.9977
Pre-OPF	0.9882	0.9838	0.995	0.9944	0.9903	0.9935
Post-OPF	0.9845	0.9774	0.9941	0.99	0.9906	0.9934
Pre-DRI	0.9956	0.9852	0.9877	0.9957	0.9875	0.9882
Post-DRI	0.9908	0.9932	0.9993	0.9991	0.999	0.9958
Pre-DRD	0.9981	0.9916	0.9987	0.998	0.9965	0.9983
Post-DRD	0.9909	0.9882	0.9517	0.9568	0.9772	0.9944
Pre-RPF	0.9848	0.9747	0.9938	0.983	0.971	0.9817
Post-RPF	0.9906	0.9877	0.995	0.9811	0.9604	0.9849

Fig. 18. Summary of R2 by COE variable

Feature Pre-PRM TrAnom got the worst R2 (0.637). This is the variable that had to be managed as a classification problem. When we initially utilized regression analysis, as we did with the rest of the target variables, R2 score for this feature was barely 0.03.

Feature Post-DRD SMA, the one with the highest skewness, also had a low R2 score of 0.77, but after its inverse transformation, it finally reached 0.99 score. The rest of the target features scored R2 above 0.95.

Performance Time. In regards to response time, we discovered that using GridSearchCV with no tuned parameters turned out to be more time efficient than doing exhaustive iterations with several parameters for each variable. Fig. 19 summarizes the fit method response time in seconds.



	SMA	Ecc	Incl	RAAN	AOP	TrAnom
Pre-PRM	0.0134	0.0152	0.0155	0.0183	0.0141	2.9205
Post-PRM	0.0112	0.0146	0.0406	0.0129	0.1025	0.0395
Pre-TLI	0.0146	0.016	0.0194	0.0158	0.012	0.0175
Post-TLI	0.0222	0.0187	0.0151	0.015	0.0136	0.0186
Pre-OCO	0.0237	0.0234	0.0158	0.0154	0.0129	0.0157
Post-OCO	0.0212	0.0177	0.0152	0.016	0.0139	0.0226
Pre-OPF	0.0209	0.0162	0.0176	0.5652	0.0139	0.0161
Post-OPF	0.0143	0.0148	0.017	0.559	0.0126	0.0165
Pre-DRI	0.0157	0.0166	0.0181	0.0164	0.0145	0.0167
Post-DRI	0.0163	0.0164	0.2434	0.2255	0.2234	0.0163
Pre-DRD	0.0163	0.0178	0.2375	0.2455	0.2454	0.2556
Post-DRD	0.0158	0.0162	0.4163	0.3147	0.6159	0.5991
Pre-RPF	2.4758	2.8923	0.4602	0.5848	0.5661	0.5651
Post-RPF	0.6199	0.2734	0.4511	0.5442	0.547	0.5602

Fig. 19. Summary of fit response time by COE variable

Most of the variables were fit in the one to three second range. However, a few response times ranged from 40 to 120 seconds when we were still using GridSearchCV with different customized parameters.

During the iterative process of model refinement, we noticed that skewness negatively affects both R2 efficiency and fitting response time. The strategies to deal with data skewness included: standardizing the data, trying different algorithms, trying different sets of parameters, and applying data transformations like the inverse or log [19, 20]. At the end of this process, all COE variables except Pre-PRM TrAnom fit acceptable R2 and response times.

Selected Algorithms. The fastest algorithms were Bayesian Ridge (brr) and K-Nearest Neighbors (knn). They were intentionally moved up to the top of the list so they could be run before the other methods. In fact, we designed the process such that other algorithms are not run unless we were unable to achieve the minimum required R2 with knn or brr. Fig. 20 summarizes the number of times that each algorithm was used in all 84 target variables. Table 4 details the algorithms used by variable.

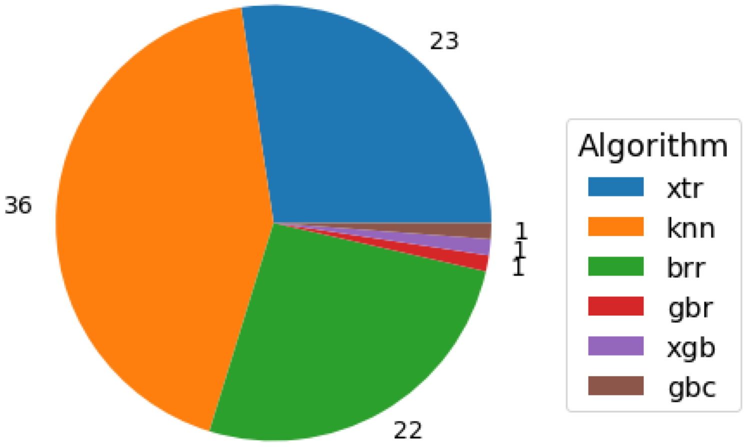


Fig. 20. Number of COE target variables fitted by algorithm.

Table 4. Detail of algorithms chosen by COE variable based off of R2 results.

Segment	SMA	Ecc	Incl	RAAN	AOP	TrAnom
Pre-PRM	brr	brr	brr	knn	brr	gbc
Post-PRM	brr	brr	brr	knn	brr	brr
Pre-TLI	brr	brr	brr	knn	brr	brr
Post-TLI	knn	knn	knn	knn	brr	brr
Pre-OCO	knn	knn	knn	knn	brr	brr
Post-OCO	knn	knn	knn	knn	brr	knn

Pre-OPF	knn	knn	knn	xtr	knn	knn
Post-OPF	brr	brr	knn	xtr	knn	knn
Pre-DRI	knn	knn	knn	knn	knn	knn
Post-DRI	knn	knn	xtr	xtr	xtr	knn
Pre-DRD	knn	knn	xtr	xtr	xtr	xtr
Post-DRD	knn	knn	xtr	xtr	xtr	xtr
Pre-RPF	xgb	gbr	xtr	xtr	xtr	xtr
Post-RPF	xtr	xtr	xtr	xtr	xtr	xtr

K-Nearest Neighbors (knn), Extremely Randomized Trees (xtr), and Bayesian Ridge (brr) were the most widely used regressors. Extreme Gradient Boosting (xgb), Gradient Tree Boosting regressor (gbr) and classifier (gbc) were used each in just one variable, mainly because knn, xtr or brr could not reach the minimum R2 required for early stop. The rest of the algorithms like Random Forest (rf) or Convolutional Neural Networks (cnn) were not really used in the last runs, because the first algorithms had already been successful in fitting the data.

Prediction. After we regressed all 84 target variables, we predicted new launch opportunities at the same initial epochs from the initial dataset plus a time shift of 30 seconds, thus increasing NASA's one-minute resolution to 30 second resolution without having to run a full DAMOCLES/Copernicus scan. We also had to consider that the last launch opportunity in a day should not be included to predict, because in theory, NASA could not assure that one minute after the day's last opportunity would still be a valid opportunity. Therefore, we removed 157 opportunities (one opportunity per day) from the new Trajectories dataset to get a final number of 18,823 launch epochs.

Since the shift time of half a minute is a very small difference compared to the number of days of the entire mission, and given that 83 out of 84 target variables got R2 score greater than 0.95, it was expected that predicted orbital elements were going to be close to the initial values.

As an example of the model accuracy in 83 out of 84 variables, Fig. 21 shows the actual values for orbital element SMA during all segments for launch epoch 11/23/2019 09:22:00 compared to the value predicted at 30 seconds later. As expected, most of these values are overlapped.

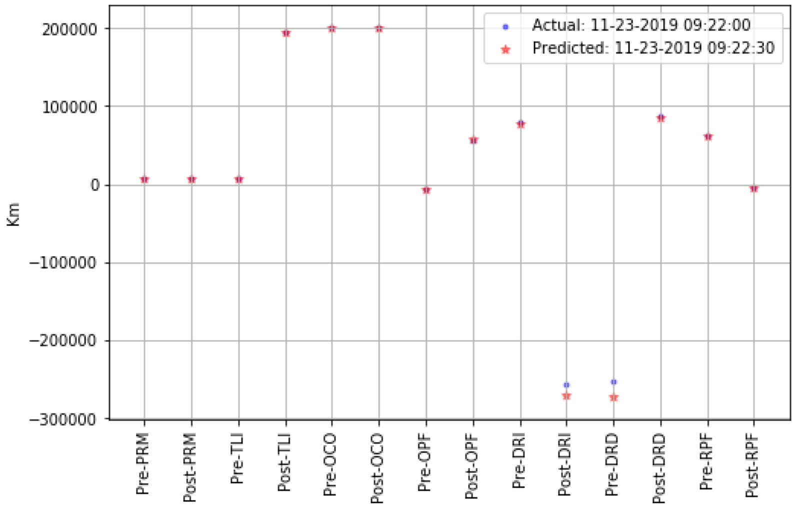


Fig. 21. Actual vs predicted values for orbital element SMA during all segments for launch epoch 11/23/2019 09:22:00 and 30 seconds later.

Feature Pre-PRM TrAnom was treated as a as a binary variable using a classification analysis. Its accuracy was only 0.64, and so it is expected that 64% of all launch opportunities overlap, but the remaining 36% show diverging values. As an example of the latter scenario, Fig. 22 shows the actual versus predicted values of this feature at a particular launch epoch with diverging values. It is advisable the need for further analysis regarding the behavior of this orbital element at this particular segment.

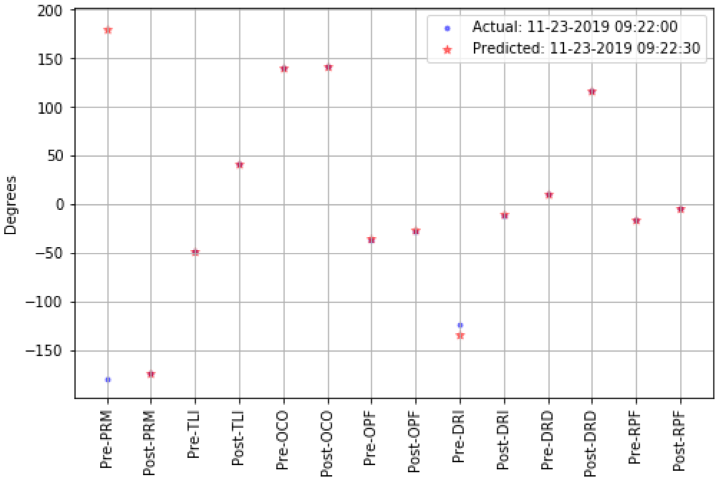


Fig. 22. Actual vs predicted values for orbital element true anomaly during all segments for launch epoch 11/23/2019 09:22:00 and 30 seconds later. See there is no overlap only at stage Pre-PRM

6 Ethics

As with medicine, data science projects must also consider ethics to ensure that we as data scientists “do no harm”. While our NASA project revolves around NASA data for NASA purposes, there are still some innate ethical considerations to ensure that our machine learning algorithms are understood. As data scientists we must be transparent with our methods and explain them well in order for customers of our data to believe in them and ensure that our methods were sound and can be counted on for their intended purposes. In this article we decided to not simply look at highly utilized, or highly ranked machine learning regression methods. In an effort to be more comprehensive, we performed a grid search that looked across a wide variety of methods. Additionally, we did not simply pick the best method overall and apply this method to all variables. Instead, we chose the algorithm that gave us the best R2 values for each variable. This way, no matter how the data changes in the future, whether it be small tweaks to input variables or an overall change to the mission itself, the logic put in place will always look for the best algorithm for the particular case in question. This approach eliminates any potential biases and ensures we accept the best case for the specific problem.

For this paper, we used preliminary trajectory data cleared for our use which was cleaned of export-controlled and/or proprietary vendor data. However, when implementing these solutions with the full and final dataset, NASA will need to ensure the safety of the data so that it does not fall into the wrong hands. While the data is unclassified, care must be taken with respect to export control. As a result, we recommend that the final NASA trajectory data be placed in a FedRamp Moderate secure location in the AWS Govcloud where it can be accessed by internal NASA personnel only via standard NASA authorization and authentication protocols. This level of security protocol guards export-controlled data at the desired International Traffic in Arms Regulations (ITAR) level.

7 Conclusions and Forward Work

This paper gives NASA a new framework with which to handle their big data problem. We have provided them with the architecture for a data and machine learning pipeline in the cloud. We have provided several ways to increase compute times on regular DAMOCLES/Copernicus scans via high performance computing in the cloud as well as increasing scan resolution via machine learning. Additionally, we created a means to go from a very large data set to just the required data for visualizing launch opportunities. We utilized Tableau software to create an interactive calendar visualization that can be used in the decision-making process in choosing the ideal time/day to launch.

Our machine learning approach using supervised regression techniques successfully predicted all the target variables. The mean R2 score was 0.98, and only one variable scored under 0.95. Machine learning can help to save computing time predicting trajectories at shorter resolution times than the given trajectory dataset without fully running a new scan. It can also serve as an initial guess generator that can be used to plug in a better starting point for a subsequent trajectory that requires less iteration than using the previous answer. Depending on the length of the scan, this can save hours and even days on trajectory scans spanning several months.

We regressed 84 target variables with different characteristics on their distribution, so we learned that when using Python + sklearn + GridSearchCV, it is better to start with default parameters rather than trying with an exhaustive combination of possible parameter values.

For most of our target variables it was useful to identify that some algorithms (like knn) were faster than others without compromising efficiency; efficiency vs accuracy trades can be made as some "slower" algorithms (like extreme gradient boosting) can further improve the learning rate and get higher R2 scores than "faster" algorithms. It may be possible to go with a much faster algorithm if one can determine that the accuracy is good enough.

For future work, we recommend looking further into the R2 score for variable Pre-PRM TrAnom. We obtained an R2 of 0.64 after its transformation to a binary variable (it was 0.03 before that). This is very different than the other variables, therefore we recommend further data analysis with subject matter experts from NASA. Additionally, we believe that our machine learning techniques can be applied to a large trajectory data set to validate the data and identify potential outliers that are out of family from the norm. We also recommend automating the visualization process in such a way as to automatically generate the new Tableau visualization every time a new DAMOCLES/Copernicus scan is run.

References

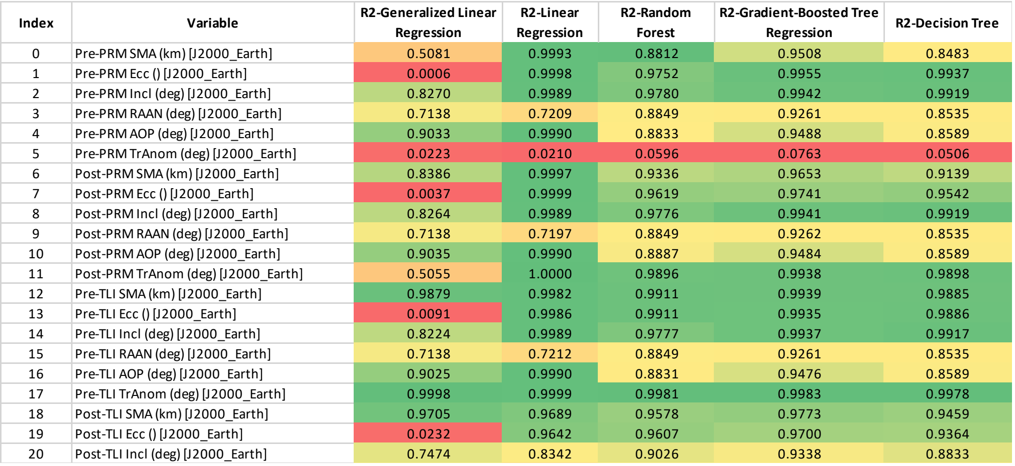
1. Cortwright, E.: Apollo Expeditions to the Moon. Scientific and Technical Information Office, NASA HQ Washington, DC, SP-350, (1975)
2. Lunar Orbit Rendezvous - NASA Documentary, NASA Mission Planning and Analysis Division (1968)
3. NASA: Around the Moon with NASA's First Launch of SLS with Orion, https://www.nasa.gov/feature/around-the-moon-with-nasa-s-first-launch-of-sls-with-orion
4. Braeunig, R: Rocket and Space Technology, Orbital Mechanics, http://www.braeunig.us/space/orbmech.htm (2013)
5. Hansen J. R.: "Enchanted Rendezvous: John C. Houbolt and the Genesis of the Lunar-Orbit Rendezvous Concept". Monographs in Aerospace History Series #4. Washington, D.C.: NASA. NASA-TM-111236 (1995)
6. Bate, R., Mueller, D., White, J.: Fundamentals of Astrodynamics, Dover Publications, New York (1971).
7. Apache Spark™ Homepage https://spark.apache.org/
8. Rezaul, K., Alla, S.: Scala and Spark for Big Data Analytics: Explore the concepts of functional programming, data streaming, and machine learning. Packt Publishing. Birmingham, UK (2017)
9. James, G., Witten, D., Hastie, T., Tibshirani, R.: An Introduction to Statistical Learning with Applications in R. Springer. New York (2013). Chapters 3, 8
10. Geurts, P., Ernst, D., Wehenkel, L. Machine Learning, Volume 63, Issue 1, https://doi.org/10.1007/s10994-006-6226-1, (2006) 3-42.
11. MacKay, David J. C.: Bayesian Interpolation. In Computation and Neural Systems, California Institute of Technology 139-74, Pasadena CA (1991)
12. Friedman, J.: Greedy Function Approximation: A Gradient Boosting Machine, IMS Reitz Lecture (1999)
13. Tseng, G.: Gradient Boosting and XGBoost, https://medium.com/@gabrieltseng/gradient-boosting-and-xgboost-c306c1bcfaf5 (2018)
14. Chen T, Guestrin C.: XGBoost. A Scalable Tree Boosting System, University of Washington, (2016)

15. Minitab Blog Editor: Regression Analysis: How Do I interpret R-squared and Assess the Goodness-of-Fit?. https://blog.minitab.com/blog/adventures-in-statistics-2/regression-analysis-how-do-i-interpret-r-squared-and-assess-the-goodness-of-fit (2013)
16. Brownlee, J.: Use Keras Deep Learning Models with Scikit-Learn in Python. https://machinelearningmastery.com/use-keras-deep-learning-models-scikit-learn-python/ (2016)
17. Raschka, S.: About Feature Scaling and Normalization and the effect of standardization for machine learning algorithms. https://sebastianraschka.com/Articles/2014_about_feature_scaling.html (2014)
18. Pedregosa, F. et al.: Scikit-learn: Maschine Learning in Python. Journal of Machine Learning Research 12, (2011) 2825-2830
19. Cox, N.: Transformations: An Introduction. Durham University. http://fmwww.bc.edu/repec/bocode/t/transint.html (2005)
20. Yeo, I., Johnson, R.: A New Family of Power Transformations to Improve Normality or Symmetry. Oxford University Press on behalf of Biometrika Trust https://www.jstor.org/stable/2673623 (2000)
21. Vox, Apollo 11's Journey to the Moon, Annotated, https://www.youtube.com/watch?v=OCjhCL2iqlQ, (2018)
22. Healy, K.: Artificial Intelligence Research and Applications. In NASA Johnson Space Center, The AI Magazine, Volume 7 (1986)

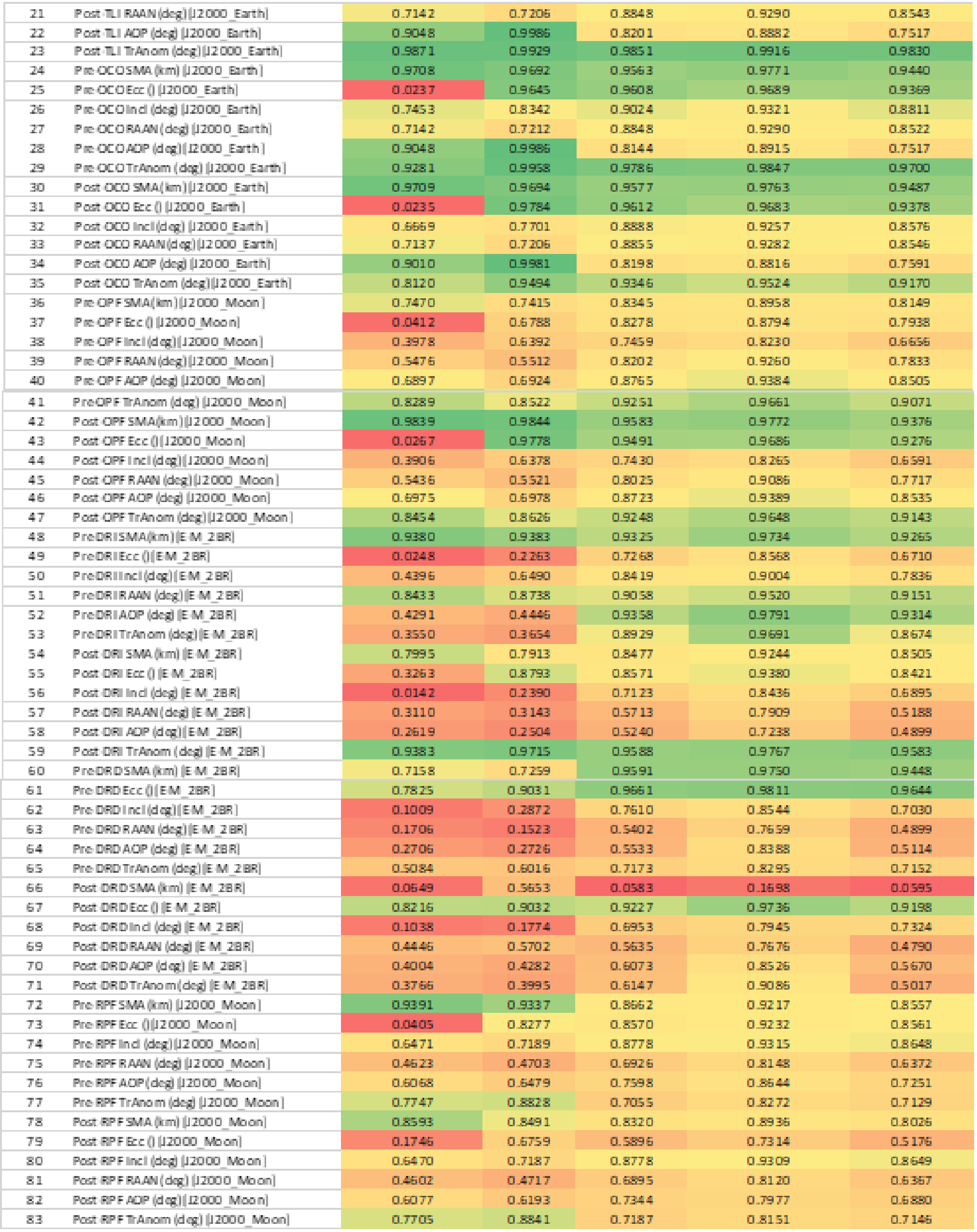
Appendix: Results of Machine Learning Analysis using PySpark

PySpark obtained similar results in the regression and classification analysis comparing the R-squared (R2) scores to those using Python and the scikit-learn API.

Fig. 23 shows a summary of the R2 scores using different algorithms in PySpark.



Index	Variable	R2-Generalized Linear Regression	R2-Linear Regression	R2-Random Forest	R2-Gradient-Boosted Tree Regression	R2-Decision Tree
0	Pre-PRM SMA (km) [J2000_Earth]	0.5081	0.9993	0.8812	0.9508	0.8483
1	Pre-PRM Ecc () [J2000_Earth]	0.0006	0.9998	0.9752	0.9955	0.9937
2	Pre-PRM Incl (deg) [J2000_Earth]	0.8270	0.9989	0.9780	0.9942	0.9919
3	Pre-PRM RAAN (deg) [J2000_Earth]	0.7138	0.7209	0.8849	0.9261	0.8535
4	Pre-PRM AOP (deg) [J2000_Earth]	0.9033	0.9990	0.8833	0.9488	0.8589
5	Pre-PRM TrAnom (deg) [J2000_Earth]	0.0223	0.0210	0.0596	0.0763	0.0506
6	Post-PRM SMA (km) [J2000_Earth]	0.8386	0.9997	0.9336	0.9653	0.9139
7	Post-PRM Ecc () [J2000_Earth]	0.0037	0.9999	0.9619	0.9741	0.9542
8	Post-PRM Incl (deg) [J2000_Earth]	0.8264	0.9989	0.9776	0.9941	0.9919
9	Post-PRM RAAN (deg) [J2000_Earth]	0.7138	0.7197	0.8849	0.9262	0.8535
10	Post-PRM AOP (deg) [J2000_Earth]	0.9035	0.9990	0.8887	0.9484	0.8589
11	Post-PRM TrAnom (deg) [J2000_Earth]	0.5055	1.0000	0.9896	0.9938	0.9898
12	Pre-TLI SMA (km) [J2000_Earth]	0.9879	0.9982	0.9911	0.9939	0.9885
13	Pre-TLI Ecc () [J2000_Earth]	0.0091	0.9986	0.9911	0.9935	0.9886
14	Pre-TLI Incl (deg) [J2000_Earth]	0.8224	0.9989	0.9777	0.9937	0.9917
15	Pre-TLI RAAN (deg) [J2000_Earth]	0.7138	0.7212	0.8849	0.9261	0.8535
16	Pre-TLI AOP (deg) [J2000_Earth]	0.9025	0.9990	0.8831	0.9476	0.8589
17	Pre-TLI TrAnom (deg) [J2000_Earth]	0.9998	0.9999	0.9981	0.9983	0.9978
18	Post-TLI SMA (km) [J2000_Earth]	0.9705	0.9689	0.9578	0.9773	0.9459
19	Post-TLI Ecc () [J2000_Earth]	0.0232	0.9642	0.9607	0.9700	0.9364
20	Post-TLI Incl (deg) [J2000_Earth]	0.7474	0.8342	0.9026	0.9338	0.8833



21	Post-TLI RAAN (deg) [J2000_Earth]	0.7142	0.7206	0.8848	0.9290	0.8543
22	Post-TLI AOP (deg) [J2000_Earth]	0.9048	0.9986	0.8201	0.8882	0.7517
23	Post-TLI TrAnom (deg) [J2000_Earth]	0.9871	0.9929	0.9851	0.9916	0.9830
24	Pre-OCO SMA (km) [J2000_Earth]	0.9708	0.9692	0.9563	0.9771	0.9440
25	Pre-OCO Ecc () [J2000_Earth]	0.0237	0.9645	0.9608	0.9689	0.9369
26	Pre-OCO Incl (deg) [J2000_Earth]	0.7453	0.8342	0.9024	0.9321	0.8811
27	Pre-OCO RAAN (deg) [J2000_Earth]	0.7142	0.7212	0.8848	0.9290	0.8522
28	Pre-OCO AOP (deg) [J2000_Earth]	0.9048	0.9986	0.8144	0.8915	0.7517
29	Pre-OCO TrAnom (deg) [J2000_Earth]	0.9281	0.9958	0.9786	0.9847	0.9700
30	Post-OCO SMA (km) [J2000_Earth]	0.9709	0.9694	0.9577	0.9763	0.9487
31	Post-OCO Ecc () [J2000_Earth]	0.0235	0.9784	0.9612	0.9683	0.9378
32	Post-OCO Incl (deg) [J2000_Earth]	0.6669	0.7701	0.8888	0.9257	0.8576
33	Post-OCO RAAN (deg) [J2000_Earth]	0.7137	0.7206	0.8855	0.9282	0.8546
34	Post-OCO AOP (deg) [J2000_Earth]	0.9010	0.9981	0.8198	0.8816	0.7591
35	Post-OCO TrAnom (deg) [J2000_Earth]	0.8120	0.9494	0.9346	0.9524	0.9170
36	Pre-OPF SMA (km) [J2000_Moon]	0.7470	0.7415	0.8345	0.8958	0.8149
37	Pre-OPF Ecc () [J2000_Moon]	0.0412	0.6788	0.8278	0.8794	0.7938
38	Pre-OPF Incl (deg) [J2000_Moon]	0.3978	0.6392	0.7459	0.8230	0.6656
39	Pre-OPF RAAN (deg) [J2000_Moon]	0.5476	0.5512	0.8202	0.9260	0.7833
40	Pre-OPF AOP (deg) [J2000_Moon]	0.6897	0.6924	0.8765	0.9384	0.8505
41	Pre-OPF TrAnom (deg) [J2000_Moon]	0.8289	0.8522	0.9251	0.9661	0.9071
42	Post-OPF SMA (km) [J2000_Moon]	0.9839	0.9844	0.9583	0.9772	0.9376
43	Post-OPF Ecc () [J2000_Moon]	0.0267	0.9778	0.9491	0.9686	0.9276
44	Post-OPF Incl (deg) [J2000_Moon]	0.3906	0.6378	0.7430	0.8265	0.6591
45	Post-OPF RAAN (deg) [J2000_Moon]	0.5436	0.5521	0.8025	0.9086	0.7717
46	Post-OPF AOP (deg) [J2000_Moon]	0.6975	0.6978	0.8723	0.9389	0.8535
47	Post-OPF TrAnom (deg) [J2000_Moon]	0.8454	0.8626	0.9248	0.9648	0.9143
48	Pre-DRI SMA (km) [E-M_2BR]	0.9380	0.9383	0.9325	0.9734	0.9265
49	Pre-DRI Ecc () [E-M_2BR]	0.0248	0.2263	0.7268	0.8568	0.6710
50	Pre-DRI Incl (deg) [E-M_2BR]	0.4396	0.6490	0.8419	0.9004	0.7836
51	Pre-DRI RAAN (deg) [E-M_2BR]	0.8433	0.8738	0.9058	0.9520	0.9151
52	Pre-DRI AOP (deg) [E-M_2BR]	0.4291	0.4446	0.9358	0.9791	0.9314
53	Pre-DRI TrAnom (deg) [E-M_2BR]	0.3550	0.3654	0.8929	0.9691	0.8674
54	Post-DRI SMA (km) [E-M_2BR]	0.7995	0.7913	0.8477	0.9244	0.8505
55	Post-DRI Ecc () [E-M_2BR]	0.3263	0.8793	0.8571	0.9380	0.8421
56	Post-DRI Incl (deg) [E-M_2BR]	0.0142	0.2390	0.7123	0.8436	0.6895
57	Post-DRI RAAN (deg) [E-M_2BR]	0.3110	0.3143	0.5713	0.7909	0.5188
58	Post-DRI AOP (deg) [E-M_2BR]	0.2619	0.2504	0.5240	0.7238	0.4899
59	Post-DRI TrAnom (deg) [E-M_2BR]	0.9383	0.9715	0.9588	0.9767	0.9583
60	Pre-DRD SMA (km) [E-M_2BR]	0.7158	0.7259	0.9591	0.9750	0.9448
61	Pre-DRD Ecc () [E-M_2BR]	0.7825	0.9031	0.9661	0.9811	0.9644
62	Pre-DRD Incl (deg) [E-M_2BR]	0.1009	0.2872	0.7610	0.8544	0.7030
63	Pre-DRD RAAN (deg) [E-M_2BR]	0.1706	0.1523	0.5402	0.7659	0.4899
64	Pre-DRD AOP (deg) [E-M_2BR]	0.2706	0.2726	0.5533	0.8388	0.5114
65	Pre-DRD TrAnom (deg) [E-M_2BR]	0.5084	0.6016	0.7173	0.8295	0.7152
66	Post-DRD SMA (km) [E-M_2BR]	0.0649	0.5653	0.0583	0.1698	0.0595
67	Post-DRD Ecc () [E-M_2BR]	0.8216	0.9032	0.9227	0.9736	0.9198
68	Post-DRD Incl (deg) [E-M_2BR]	0.1038	0.1774	0.6953	0.7945	0.7324
69	Post-DRD RAAN (deg) [E-M_2BR]	0.4446	0.5702	0.5635	0.7676	0.4790
70	Post-DRD AOP (deg) [E-M_2BR]	0.4004	0.4282	0.6073	0.8526	0.5670
71	Post-DRD TrAnom (deg) [E-M_2BR]	0.3766	0.3995	0.6147	0.9086	0.5017
72	Pre-RPF SMA (km) [J2000_Moon]	0.9391	0.9337	0.8662	0.9217	0.8557
73	Pre-RPF Ecc () [J2000_Moon]	0.0405	0.8277	0.8570	0.9232	0.8561
74	Pre-RPF Incl (deg) [J2000_Moon]	0.6471	0.7189	0.8778	0.9315	0.8648
75	Pre-RPF RAAN (deg) [J2000_Moon]	0.4623	0.4703	0.6926	0.8148	0.6372
76	Pre-RPF AOP (deg) [J2000_Moon]	0.6068	0.6479	0.7598	0.8644	0.7251
77	Pre-RPF TrAnom (deg) [J2000_Moon]	0.7747	0.8828	0.7055	0.8272	0.7129
78	Post-RPF SMA (km) [J2000_Moon]	0.8593	0.8491	0.8320	0.8936	0.8026
79	Post-RPF Ecc () [J2000_Moon]	0.1746	0.6759	0.5896	0.7314	0.5176
80	Post-RPF Incl (deg) [J2000_Moon]	0.6470	0.7187	0.8778	0.9309	0.8649
81	Post-RPF RAAN (deg) [J2000_Moon]	0.4602	0.4717	0.6895	0.8120	0.6367
82	Post-RPF AOP (deg) [J2000_Moon]	0.6077	0.6193	0.7344	0.7977	0.6880
83	Post-RPF TrAnom (deg) [J2000_Moon]	0.7705	0.8841	0.7187	0.8151	0.7146

Fig. 23. Scores using PySpark

PySpark also struggled with feature Pre-PRM TrAnom. In the regression exercise, R2 was barely 0.07, but it increased to 0.36 after having managed the problem as a classification analysis. In general, the exercise with PySpark has helped to confirm the results and findings with Python scikit-learn with most R2 scores above 0.88.