

# Data Accountability and Uncertainty Analysis in Mars Science Lab

Ryan Alimo\*

*Jet Propulsion Laboratory, California Institute of Technology*  
sralimo@jpl.nasa.gov

Brian Kahovec\*

*Jet Propulsion Laboratory, California Institute of Technology*  
brian.kahovec@jpl.nasa.gov

Dylan Sam\* †

*Department of Computer Science, Brown University*  
dylan\_sam@brown.edu

Ameera Chowdhury ‡

*Jet Propulsion Laboratory, California Institute of Technology*  
ameerah@alumni.caltech.edu

Dariush Divsalar

*Jet Propulsion Laboratory, California Institute of Technology*  
dariush.divsalar@jpl.nasa.gov

**This paper presents machine learning based approaches to automate and optimize the detection of volume loss for the downlink process of telemetry data from the Mars Curiosity Rover. The Curiosity observes volume loss and data corruption, requiring re-transmits from the rover and Ground Data System Analysts (GDSA) to monitor the data flow. To resolve this issue, we created a data pipeline to accumulate data from various data sources in the downlink process and detect where the data is missed. In this paper, we benchmarked different methodologies based on the accuracy and excitability of them to identify whether a downlink data that is received to the ground system is complete or incomplete. Our results show that machine learning methods can improve the performance of the GDSA by 55% while the user can diagnose why a data is missed and provide explanation for the data accountability problem.**

## I. Introduction

The Mars Science Lab (MSL) Real-Time Operations team at NASA’s Jet Propulsion Laboratory monitors the downlink process of telemetry data from the Mars Curiosity Rover [1] back down to Earth. The current MSL downlink process includes the Mars Orbiters, the Deep Space Network (DSN) [2], JPL Data Control, and MSL Ground Data Systems. During the transmission of data through this process, time stamps, data volumes, and other metadata are recorded to make sure the data is being transmitted successfully. However, there are frequent losses in data as it is sent through these multiple locations, which sometimes require re-transmissions by the rover. There is a need for a better understanding of the cause of data loss and re-transmission, which can help the Ground Data System Analyst (GDSA) team better determine the root cause of the issues in the Ground Data System (GDS).

The use of machine learning models has become widespread in analyzing data from space [3–6]; however, the inability of scientists to understand these models seems problematic [7–10]. There are different interpretation about the interpretability or explainability of a model. Several works suggest that explainability means trust [11], but it is unclear what trust is.

Machine Learning approaches are particularly useful, as they can learn abstract and relational features in a dataset. Although the GDSA team has an expert understanding of the feature space, machine learning algorithms allow the model

---

\*First, second, and third authors have equal contribution and are listed in alphabetic order.

†This project has been completed while the author was a summer intern at Jet Propulsion Laboratory, California Institute of Technology.

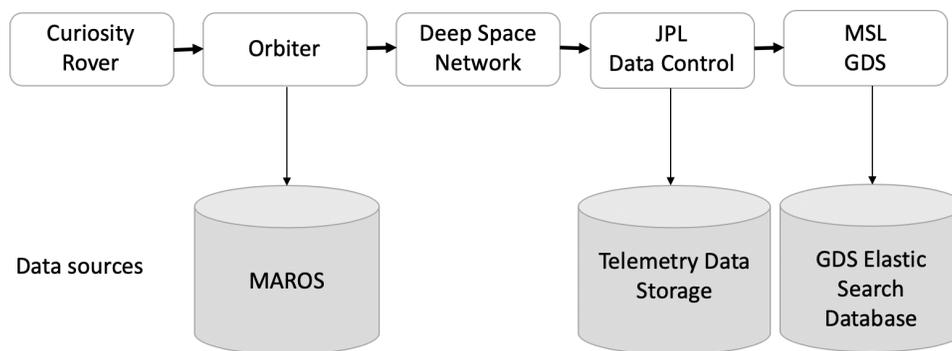
‡This project has been completed while the author was a summer intern at Jet Propulsion Laboratory, California Institute of Technology.

to possibly learn other important features that are obscure. Neural networks and random forests can utilize these learned relations to make accurate predictions on new data. These algorithms can be applied to determine whether future Mars Curiosity Rover transmissions are corrupted and require re-transmission based on the recorded historical data.

The paper is organized as follows. Following this introduction, in Section I.A provides some background for the problem and a short overview of the problem statement. Section II describes the proposed approaches to the data accountability problem. Then, the adopted algorithms are outlined in more details in Section III and discusses how to make the methods more explainable for the GDSA analyst. Section IV presents a preliminary analysis of the performance of the proposed strategy. Finally in Section V, we draw the conclusions.

## A. Background

Metadata about each downlink is recorded at each of the data sources in Figure 1. GDSAs access these data sources and view the metadata to determine if the downlink is being successfully transferred through the downlink process. We will gather the metadata from these data sources to use in our machine learning algorithms.



**Fig. 1 High-level architecture of the Mars Science Lab (MSL) Downlink Process, which is the process of sending telemetry data recorded by the rover back to the MSL team. First, the Curiosity Rover sends data to one of the Mars Orbiters. Then, the orbiter sends the data to one of the Deep Space Network stations. Afterward, the data is sent to the Jet Propulsion Laboratory (JPL), where it is received by Data Control and stored as transfer frames. Finally, the MSL team receives the data and converts the frames into packets and data products. GDSAs report the total data volume of the downlink in megabits.**

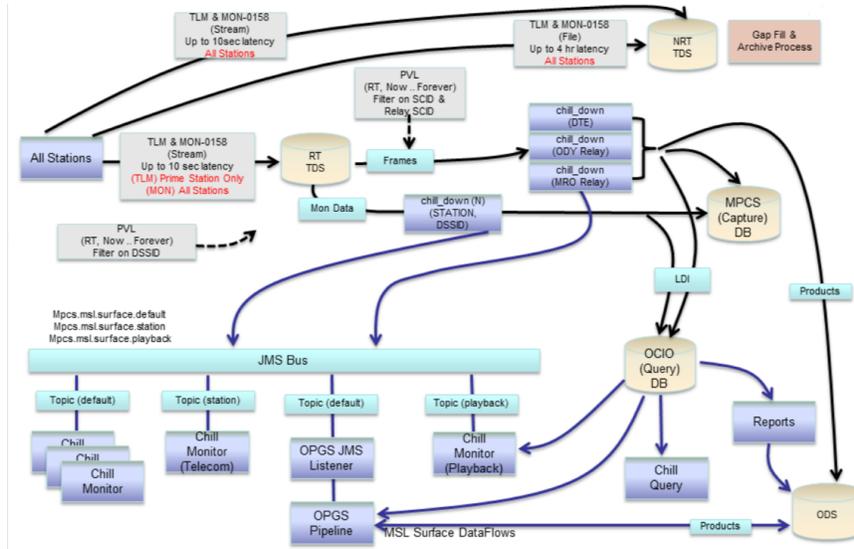
Figure 2 shows that the GDS is complex and there are many locations where data can be corrupted or lost. When a downlink is unsuccessful, it can take several hours to identify the root cause of the issue.

There have been attempts to automate the task of determining whether or not a downlink was successful, but these methods have proved to be inaccurate [12]. The GDSA Dashboard is the current automated system for identifying complete and incomplete downlinks. Table 1 shows that the GDSA Dashboard is unreliable, and therefore not suited for mission operations.

	Precision	Recall	f1-score	Support
Incomplete	0.74	<b>0.55</b>	0.63	1141
Complete	0.94	0.97	0.95	7867
Avg/Total	0.91	0.92	0.91	9008

**Table 1 Accuracy of the GDSA Dashboard**

A recall of 0.55 means that only 55% of incomplete passes are labelled as incomplete. Our machine learning algorithms hope to improve on these values.



**Fig. 2 Overview of the ground data system architecture [1]. This figure illustrates the complexity of the data flow from space to the ground data system.**

## II. Algorithms Description

### A. Supervised Learning Models

As shown in section I.A, the current software unreliably labels passes as successful or unsuccessful. We implemented various machine learning algorithms to classify each downlink as successful or unsuccessful and improve on GDS software. Although we may frame labelling the passes as a multi-class classification problem in the future, for now we only look to implement binary classification for simplification. Furthermore, binary classification is most usable to the GDSA team because the failed completion of a pass indicates the need for a re-transmission.

#### 1. Neural Networks

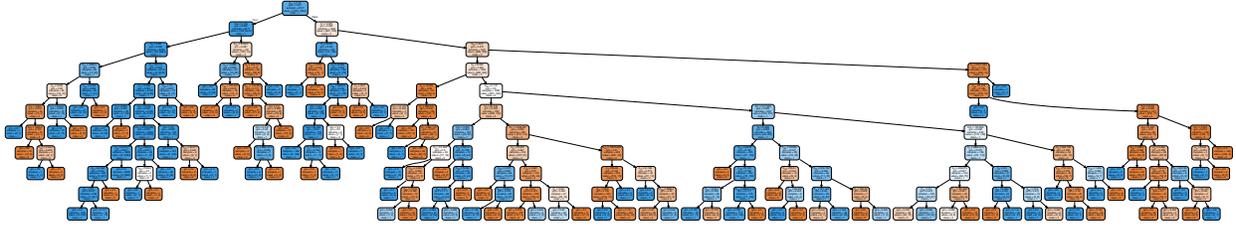
We implemented a feed-forward neural network to perform binary classification on our data points. The network architecture consists of five dense layers with a dropout rate of 0.5 between each of the first four layers. Each hidden layers used ReLU [13] as its activation function, and sigmoid was applied on the output layer to compute the probability of success.

The data was broken up into training, validation, and test sets. The split was 70%, 20%, and 10% respectively. When including categorical one-hot vectors, each data point consisted of 42 features. When omitting categorical features, each data point contained 18 features. We experimented with both min-max and z-score normalization for the dataset, and we chose z-score because it led to higher classification accuracy. The network was trained for 400 epochs with a batch size of 20, with data randomly shuffled between each epoch. A learning rate of 0.0005 and AdamOptimizer were used to update the model's weights.

#### 2. Random Forest

A random forest is a collection of decision trees. In each step of building a decision tree, we pick a feature and a real number  $R$ . Then we partition the dataset according to whether a sample's feature has value less than  $R$ . The goal is to end up with two smaller datasets that have less variance in their labels than the original dataset. Eventually, it is no longer possible to split the dataset as all samples will have the same label. To build and analyze a decision tree, we first split the data using 90% of it as a training set and 10% of it as a test set. We used a stratified split to take into account the class imbalance. The training data is further split into training data and validation data, using 10-fold cross validation, to find the best parameters for a decision tree classifier using a grid search. The decision tree classifier is evaluated using the 10% test data that was withheld. Figure 3 shows our most accurate decision tree.

While decision trees are locally accurate and have low bias, they suffer from high variance, which means that they



**Fig. 3 Decision Tree and branches. The above figure shows the complexity of the trees that makes the approach explainable.**

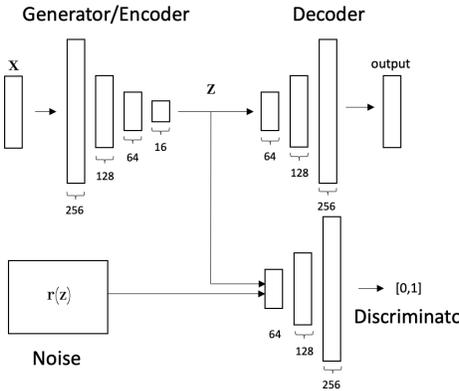
overfit to the training data and are sensitive to small changes in the training data. To reduce the variance, we can ensemble different decision trees to create a random forest classifier. To build and analyze a random forest, we again split the data using 90% as a training set and the other 10% as a test set. We used a stratified split to take into account the class imbalance. The training data is further split into training data and validation data, using 10-fold cross validation, to find the best parameters for a random forest classifier using a grid search.

**B. Model Explainability**

*1. Adversarial Autoencoders*

Adversarial autoencoders are standard encoders that constrain their latent space representation by using adversarial learning. Recall that autoencoders consist of an encoder and decoder that help project and recreate training examples to learn the most relevant features of the dataset. The encoder model projects some input  $x$  into a latent representation of that input  $z$ . In other words, it learns a function  $p(z|x)$  to map data points to latent representations. The decoder tries to recreate the original data point from this latent variable  $z$ . Thus, it learns the function  $q(x|z)$ .

The adversarial autoencoder uses adversarial learning to constrain the latent space representation  $z$ . Adversarial learning usually consists of a generator and a discriminator models, where the generator takes in randomly sampled noise to create synthetic data to fool the discriminator. For adversarial autoencoders, both the encoder and the generator are the same model; the generator learns to fool the discriminator by creating a latent space distribution similar that of the randomly selected noise. Let the randomly selected probability distribution be  $r(z)$  and the latent space distribution  $p(z)$ . The generator learns to make  $p(z)$  similar to  $r(z)$  to maximize the loss of the discriminative model. Therefore, this architecture imposes a chosen probability distribution on the latent space representation of the autoencoder.



**Fig. 4 Illustration of our adversarial autoencoder architecture.**

We implemented an Adversarial Autoencoder as illustrated in Figure 4. The encoder had four layers, which reduced the dimensionality of the input to 16, which was the chosen latent space size. The decoder consisted of four layers which reconstructed the input data point from the latent space representation. Each layer of the encoder and decoder uses the ReLU for the activation function, except for the output layer of the decoder. Since the data has been normalized by z-scoring, the decoder uses tanh as the activation function to make each dimension of the output space from  $[-1, 1]$ .

The discriminator has four layers, with the output layer representing the probability that the latent space input data was from the noise distribution. The discriminator uses ReLU for each layer except the output layer, which uses Sigmoid.

The model was trained for 50 epochs with a batch size of 50. During training, the generator and discriminator were trained by maximizing their likelihood. The encoder and decoder used the mean squared error on the reconstructed image as its loss function. The equations are as follows, with  $x$  representing the input data point,  $z$  representing randomly selected noise input, and  $\hat{x}$  representing the reconstructed image of  $x$ :

$$L_g(x) = \log(1 - D(G(x))) \quad (1a)$$

$$L_d(x, z) = \log(D(z)) - \log(D(G(x))) \quad (1b)$$

$$L_{ae}(x, \hat{x}) = \frac{1}{n} \sum_{i=1}^n (x_i - \hat{x}_i)^2 \quad (1c)$$

where  $D$  is the discriminator model and  $G$  is the generator model. The learning rates for the encoder, decoder, and discriminator were 0.0001, 0.01, and 0.01 respectively. The random noise distribution was a Gaussian distribution with  $\mu = 0.0$  and  $\sigma = 0.5$ . The GDSA team informed us that the passes should follow some Gaussian distribution. In other words, unsuccessful passes would correspond to values with very high z-scores, and successful passes would be close to the mean value.

## 2. *t*-SNE Visualization

The anomalies were visualized using t-Stochastic Neighbor Embedding (t-SNE). t-SNE is a machine learning algorithm for learning a two-dimensional representation of a dataset to aid in visualization [25]. The anomalies were visualized using a t-SNE model with a perplexity of 30, a learning rate of 200, and 1000 iterations.

## III. Data Description

We accumulated and analyzed historic data from the Curiosity Rover downlinks to best make future predictions on whether data is being correctly transmitted. We can create a predictive model with this data, which the GDSA team can use to upgrade current software in the downlink process.

### A. Data Pipeline

We collected data from the three different data sources in the downlink process using internal JPL APIs. As shown in Figure 1, MAROS, Telemetry Data Storage, and the GDS Elastic Search Database are the three different sources that store the metadata in the downlink process. MAROS receives metadata from the Mars orbiters, TDS from JPL's ground data systems, and GDS from the GDSA Team's data processing. These API's return data volumes, start and end time of data transmission, information about orbiter height and location, and other important features.

We created a data processing pipeline that queries each API and for all the data from a given sol. Then we combined the three datasets into one by matching the downlinks pulled from each of the three sources. Our dataset consists of downlink data from sol 337 to 2450, sent by six different orbiters. This resulted in 9008 data points, which is too small of a dataset for complex deep learning models. Therefore, we looked to implement data augmentation to produce more synthetic data for training our models.

### B. Signal Processing

After collecting and merging our data, we implemented signal processing to best compute relevant features from the dataset. We consulted with multiple Ground Data System Analysts to gain prior knowledge about feature importance. We computed the time-deltas between the three different data sources and between actual and predicted times for data arrival. We also calculated the disparities in data volumes between the different locations and between the actual and predicted volumes. We included both the differences in times and volumes and the raw amounts in our datasets. We constructed one-hot vectors to represent from which Mars Orbiter the data was transmitted and at which Deep Station Network station on Earth the data was received and appended those to the data points. We removed Overflight IDs and Relay Product IDs from the dataset as they are numerical IDs that have no impact on the pass.

### C. Labelling

To Label the dataset, we incorporated both heuristics and hand-labelling from experts on the GDSA team. We developed two different labelling schemes and combined them with a third set of labels labels that are automatically generated from GDSA software.

#### 1. Scheme 1: Volume Differences & Re-transmits

Our first labelling scheme utilized two different heuristics to label the dataset. From the computed differences in data volumes between different locations, the GDSA team set a threshold of 95% data retention for a successful pass. Also, the team considers all passes that are later re-transmitted as unsuccessful.

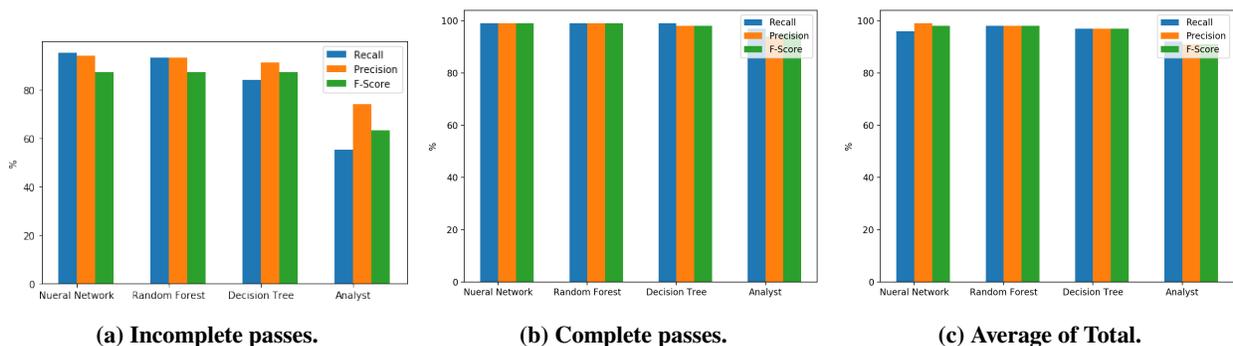
#### 2. Scheme 2: TDS Difference

The second labelling scheme compares the data volumes between MAROS, GDS, and TDS. It labels a pass as successful when the volume difference between any of the two locations is within 1 megabit or the TDS megabits data volume is larger than the other locations. Once we computed all three sets of labels, we analyzed which datapoints were labelled differently by any of the schemes. When all three schemes were in agreement, we used that label as our “ground truth”. A GDSA hand-labelled the remaining datapoints to create “ground truth” labels. We utilized these labels in our training and testing as well as our analysis on the importance of different features.

## IV. Results

The neural network achieved 95-96% accuracy on the training and validation datasets. On the test dataset, the model achieved 95.3% accuracy. When splitting the test results by orbiter, we saw that there was an accuracy of 95.5% on MRO over 337 data points, 97.3% on TGO for 38 data points, and 95.6% on ODY for 253 data points. When omitting the categorical variables in training, the neural network had an accuracy around 95% on the training set and validation set. The test accuracy was 94.5%, which is slightly worse than when the categorical one-hot vectors were included. The overall accuracy of the current GDS software is 91% accuracy.

The precision and recall of the neural network are shown in Table ?? . When comparing these values to the precision and recall in Table 1, we can see a significant improvement on the current GDS software.



**Fig. 5** Supervised learning results with different methods to identify complete and incomplete downlink pass for MSL data from Sol 337 to Sol 2450. The training data includes 9009 datapoints, but 1141 are incomplete and 7867 are complete passes. For the validation of our results we used 114 incomplete, and 787 complete passes which in total 901 were used for test set. The results from each method is averaged to be comparable with the GDSA analyst reports. The results from left to right are less explainable.

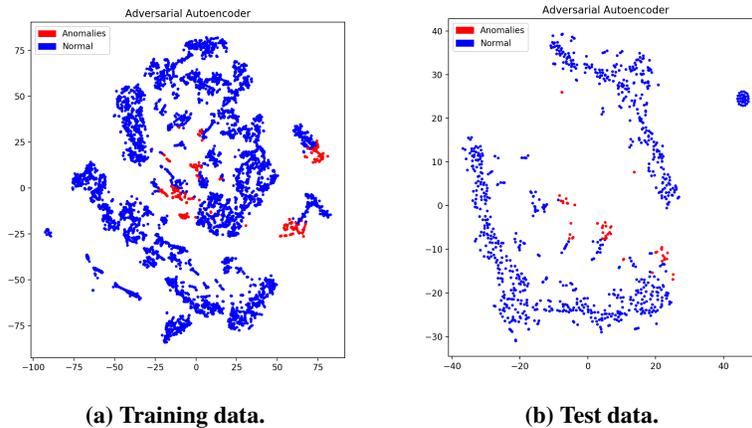
The random forest classifier is evaluated using the 10% test data that was withheld. The random forest classifier has 98% accuracy and its precision and recall are reported in Table ?? . The recall of the random forest classifier on incomplete classes is 93%, which dramatically improves upon the 55% recall of the GDSA Dashboard Labeller on incomplete passes.

We experimented with visualizing the results of training the adversarial autoencoder by displaying the latent space. Since the latent space consists of 12 dimensions, we used t-SNE to project this into two dimensions to visualize, as shown

in Figure ???. We were not able to distinguish any particularly useful clustering of data points that distinguish between the successful and the unsuccessful passes. We believe that this is due to the significant reduction of dimensionality, which implies large information loss.

The adversarial autoencoder model was trained and used to detect anomalies by selecting a reconstruction loss threshold. A loss threshold of 0.5 was selected, as shown by the black horizontal line in Figure ???. The average reconstruction loss on the test data points was 0.38, which is also skewed by the presence of data points with large loss.

The anomalies detected by the adversarial autoencoder and the one-class support vector machine (Figure 6a & 6b) illustrate multiple different groupings of anomalous data passes in the datasets.



**Fig. 6 Latent space representation of the training and test set using adversarial autoencoder.**

The areas that the autoencoder deemed anomalous were also considered anomalies by the one-class SVM. The reconstruction method seems to have learned an accurate reconstruction of the input data, because fewer points have been classified as anomalies by reconstruction loss threshold. The anomalies detected by both the autoencoder and the one-class SVM seem to be in distinct groups, which indicates that there are similar classes of anomalies.

The test set anomalies, Figure 6b, also show similar overlapping anomalies from both the AAE and the OC-SVM. These are contained within the center of the visualization and seem to be in three separate groups.

## V. Conclusion and Future Work

In this project, we have been able to apply a variety of machine learning algorithms to analyze real downlinks sent by the Mars Curiosity Rover. We implemented the following classification algorithms: neural networks, random forests, and t-Stochastic Neighbor Embedding. The neural network and random forest outperformed the previous automated methods, and the random forest is currently being used by the MSL Ground Data Systems Analysts to classify new downlinks. We also implemented the following anomaly detection algorithms: adversarial autoencoders and one-class support vector machines. The anomalies detected by these algorithms seemed to form distinct groups. In our future work for this project, we would like to gain a better understanding of these different classes of anomalies, and create classifiers that can determine which class an anomaly belongs to. Other future work includes identifying more data sources in the downlink process, and not only identify *if* there was an issue, but also identify where the issue occurred. Our goal is to simplify and automate the job of an expert analyst so that even an untrained team member can respond to issues in the Ground Data System.

## Appendix: Feature Selection Scheme

We analyzed the different features of our dataset to better understand their importance on successful and unsuccessful passes. We analyzed the effects of the features on the labels by calculating p-score and variance and using a random forest algorithm. We then compared those results with GDSA intuition.

Feature Name	p-score	variance	Feature Name	p-score	variance
GDS dataActual	0.08955028	0.001000709	TDS dssId 55	0.014390925	0.056779483
GDS dataPredict	0.055014329	0.015769806	TDS dssId 63	0.024266422	0.021671917
MAROS lander return value	0.204609025	0.013874329	TDS dssId 64	-0.050181142	0.001917114
MAROS max elevation	0.117533194	0.067001409	TDS dssId 65	5.20E-05	0.035710896
MAROS orbiter return value	0.095045297	0.014373769	TDS insync megabits	-0.064221131	0.000861176
MAROS rise elevation	0.177457655	0.046255406	TDS insync tf 0 frames	-0.035171744	0.003870473
TDS GDS end timedelta	-0.086843584	0.000906355	TDS insync tf 32 frames	-0.050215818	0.000793424
TDS GDS start timedelta	-0.1152615	0.000878692	TDS outasync tf frames	-0.05152732	0.000186214
TDS dssId 0	-0.006342271	0.011969046	TDS to GDS delta	-0.203290347	0.000425259
TDS dssId 14	-0.011767087	0.017558619	actual to predict begin timedelta	-0.096073465	0.000556861
TDS dssId 15	-0.010204555	0.012257306	actual to predict delta	-0.12248021	0.000379606
TDS dssId 24	0.01371384	0.027424613	actual to predict end timedelta	-0.088029217	0.000552953
TDS dssId 25	-0.024426821	0.043839993	orbiter DTE	-0.291881369	0.036257726
TDS dssId 26	0.003005157	0.035163367	orbiter MEX	-0.058706648	0.002358477
TDS dssId 34	0.013456289	0.029233313	orbiter MRO	0.072998623	0.24926088
TDS dssId 35	0.01450093	0.04811124	orbiter MVN	-0.044854024	0.017273589
TDS dssId 36	-0.024681067	0.019407052	orbiter ODY	0.095767076	0.240199936
TDS dssId 43	0.012240282	0.017558619	orbiter TGO	-0.021657679	0.055608649
TDS dssId 45	-0.033029234	0.016845718	orbiter nan	-0.14412848	0.009366844
TDS dssId 50	0.071154972	0.24903724	orbiter to TDS delta	-0.231028699	0.00052855
TDS dssId 54	-0.013985144	0.035984398	rover to orbiter delta	-0.271554212	0.003950459

**Table 2 Feature Selection Metrics (p-score, variance)**

From Table 2, we can see that the data volumes (GDS dataActual, MAROS lander return value, orbiter to TDS delta, and rover to orbiter delta) and time differences (actual to predict begin timedelta, actual to predict end timedelta, and TDS GDS start timedelta) have the most non-zero correlation coefficient. On the other hand, the categorical one-hot vectors representing orbiter and station (TDS dssID 55, orbiter TGO, etc.), other than orbiter DTE, were the least correlated to the data label. The DTE downlinks are sent directly to Earth via the rover’s high-gain antenna, so they do not appear in MAROS and only contain small amounts of data. Our decision to treat DTE as a separate orbiter was based on input provided by GDSA experts. The one-hot features also have the least variance, because they can only take on the discrete values of  $[0, 1]$ . Since these variables were the least correlated to the labels, we experimented by also training our neural networks with datasets without the categorical variables.

## Acknowledgments

The authors also gratefully acknowledge partial funding from Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration (NASA) in support of this work. The authors would also like to thank D. Hanks, B. Castano, and F. Y. Hadaegh.

## References

- [1] Dehghani, N., & Tankenson, M. (2006). A multi-mission event-driven component-based system for support of flight software development, ATLO, and operations first used by the Mars Science Laboratory (MSL) project.
- [2] Johnston, M., & Lad, J. (2018). Integrated Planning and Scheduling for NASA's Deep Space Network—from Forecasting to Real-time. In 2018 SpaceOps Conference (p. 2728).
- [3] Kerner, H. R., Wellington, D. F., Wagstaff, K. L., Bell, J. F., Kwan, C., & Amor, H. B. (2019, July). Novelty detection for multispectral images with application to planetary exploration. In Proceedings of the AAAI Conference on Artificial Intelligence (Vol. 33, pp. 9484-9491).
- [4] Bue, B. D., Wagstaff, K. L., Rebbapragada, U. D., Thompson, D. R., & Tang, B. (2014). Astronomical data triage for rapid science return. In Proc. 2014 Conf. on 'Big Data from Space (BiDS' 14) (Vol. 206).
- [5] James, M. L., Shapiro, A. A., Springer, P. L., & Zima, H. P. (2009). Adaptive fault tolerance for scalable cluster computing in space. The International Journal of High Performance Computing Applications, 23(3), 227-241.
- [6] Kolcio, K., Fesq, L., & Mackey, R. (2017, March). Model-based approach to rover health assessment for increased productivity. In 2017 IEEE Aerospace Conference (pp. 1-13). IEEE.
- [7] Kerber, L., Dickson, J. L., Head, J. W., & Grosfils, E. B. (2017). Polygonal ridge networks on Mars: Diversity of morphologies and the special case of the Eastern Medusae Fossae Formation. Icarus, 281, 200-219.
- [8] Thompson, D. R., Babu, K. N., Braverman, A. J., Eastwood, M. L., Green, R. O., Hobbs, J. M., ... & Mathur, A. (2019). Optimal estimation of spectral surface reflectance in challenging atmospheres. Remote Sensing of Environment, 232, 111258.
- [9] Lee, H., Kalashnikova, O. V., Suzuki, K., Braverman, A., Garay, M. J., & Kahn, R. A. (2016). Climatology of the aerosol optical depth by components from the Multi-angle Imaging SpectroRadiometer (MISR) and chemistry transport models. Atmospheric Chemistry and Physics, 16(10), 6627-6640.
- [10] Roberts, J. T., Xue, D. D., & Mandrake, L. (2018, December). Machine Learning for Data Validation and Natural Event Detection using NASA GIBS Earth Satellite Imagery. In AGU Fall Meeting Abstracts.
- [11] Lipton, Z. C. (2016). The mythos of model interpretability. arXiv preprint arXiv:1606.03490.
- [12] Ko, A., Maldague, P., Lam, D., Bui, T., & McKinney, J. (2010, April). The Evolvable Advanced Multi-Mission Operations System (AMMOS): Making Systems Interoperable. In SpaceOps 2010 Conference Delivering on the Dream Hosted by NASA Marshall Space Flight Center and Organized by AIAA (p. 2303).
- [13] Davidson, G., & Mozer, M. C. (2019). Sequential mastery of multiple tasks: Networks naturally learn to learn. arXiv preprint arXiv:1905.10837.
- [14] A. Schölkopf, R. Williamson, A. Smola, J.S. Taylor, J. Platt, *Support vector method for novelty detection*, in Neural Information Processing Systems, Elsevier, New York, 2000, pp. 582–588.
- [15] A. Makhzani, J. Shlens, and N. Jaitly. *Adversarial Autoencoders*. arXiv preprint arXiv:1511.05644, 2015.
- [16] D. Lowd and C. Meek. *Adversarial learning*. In Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining (KDD '05). ACM, New York, NY, USA, 2005.
- [17] D. P. Kingma and M. Welling. *Auto-Encoding Variational Bayes*. In ICLR, 2013.
- [18] Dorogush, A. V., Ershov, V., & Gulin, A. (2018). CatBoost: gradient boosting with categorical features support. arXiv preprint arXiv:1810.11363.
- [19] E. Nalisnick, A. Matsukawa, Y. W. Teh, B. Lakshminarayanan. *Detecting Out-of-Distribution Inputs to Deep Generative Models Using a Test for Typicality*. ArXiv e-print, 2019.
- [20] Friedman, J. H. (2002). Stochastic gradient boosting. Computational statistics & data analysis, 38(4), 367-378.
- [21] Guzewich, S. D., Lemmon, M., Smith, C. L., Martínez, G., de Vicente, Retortillo, Á., Newman, C. E., ... & Harri, A. M. (2019). Mars science laboratory observations of the 2018/mars year 34 global dust storm. Geophysical Research Letters, 46(1), 71-79.
- [22] Hundman, K., Constantinou, V., Laporte, C., Colwell, I., & Soderstrom, T. (2018, July). Detecting spacecraft anomalies using lstms and nonparametric dynamic thresholding. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (pp. 387-395). ACM.

- [23] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio. *Generative Adversarial Nets*. Advances in Neural Information Processing Systems 27: pp. 2672-2680, 2014.
- [24] J. An and S. Cho. *Variational Autoencoder based Anomaly Detection using Reconstruction Probability*. SNU Data Mining Center, Tech. Rep., 2015.
- [25] L. J. P. Van der Maaten and G. E. Hinton. *Visualizing high-dimensional data using t-SNE*. J. Mach. Learn. Res. vol. 9, pp. 2579–2605, 2008.
- [26] S. M. Erfani, S. Rajasegarar, S. Karunasekera, C. Leckie. *High-dimensional and large-scale anomaly detection using a linear one-class SVM with deep learning*. Pattern Recognition, vol. 58, pp. 121–134, 2016.
- [27] S. Rajput, Z. Feng, Z. Charles, P.L. Loh, D. Papailiopoulos. *Does data augmentation lead to positive margin?* arXiv preprint arXiv:1905.03177, 2019.
- [28] T. A. Tang, L. Mhamdi, D. McLernon, S. A. R. Zaidi, M. Ghogho, *Deep learning approach for network intrusion detection in software defined networking*, Proc. Int. Conf. Wireless Netw. Mobile Commun., 258-263, 2016.
- [29] Y. Li, R. Ma, R. Jiao, *A hybrid malicious code detection method based on deep learning*. Int. J. Security Appl., vol. 9, no. 5, pp. 205-216, 2015.