A “Big Data” Analysis of Rocket Failure

Anthony Ozerov

2019

A “Big Data” Analysis of Rocket Failure

Introduction

The field of Big Data, which I have professional and hobbyist experience in, is emerging as one which has the possibility to do everything from making advertising more effective to improving crop yields. However, one field which I have not seen it applied to is rocketry, and I wondered how Big Data could be used to predict rocket failures. The aim of this investigation is to *predict one output value—the probability of failure of a rocket—based on as much data as is practically possible to obtain in a reasonably automated manner*.

Data in the investigation will include:

- Mean climate data for the day (e.g. surface temperature)
- The launch vehicle used (e.g. a Saturn V)
- The launch site (e.g. Cape Canaveral)

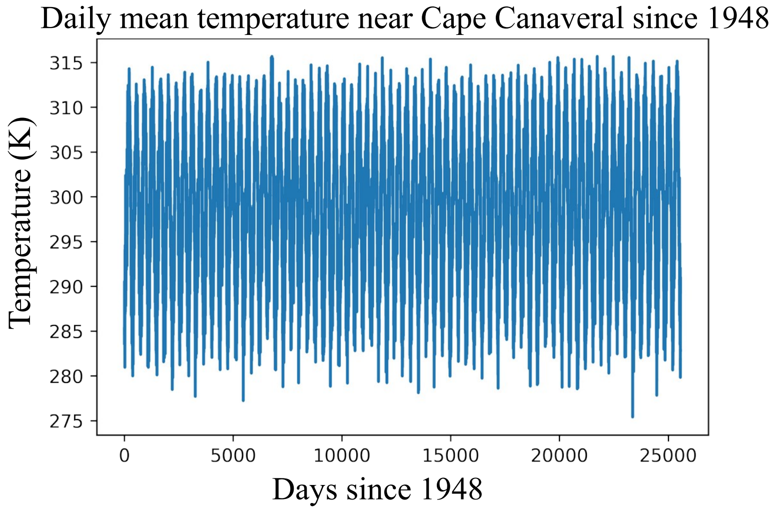
By applying a subset of statistics called “Statistical Learning” to said data, I think it will be possible to predict the probability of failure of any launch. This will provide interesting insights. For example, if there is little correlation between the climate data and the probability of failure, I could conclude that launch providers are scheduling launches at the correct times, when the weather provides no risk to the launch vehicle. The scope of this investigation is every rocket launch before 2018, as I was unable to find complete climate data for 2018 and after.

Data Sources

The only complete and sourced rocket launch dataset I found was at planet4589.org[1]. This dataset includes the date of a launch, the launch vehicle, the launch site, and if the launch was a failure or a success, and will clearly provide some of the desired **predictors** and the **output value**.

The only historical climate dataset I found that goes back far enough to cover the first rocket launches, back in 1957, and covers the entire planet, is the U.S. National Oceanic and Atmospheric Administration's (NOAA) Earth System Research Laboratory's (ESRL) National Centers for Environmental Protection/University Corporation for Atmospheric Research (NCEP/NCAR) Reanalysis I[2]. I took the daily mean data for every year from this dataset and collated into one large file to provide the quantitative climate predictors I want. One limitation of this data is that it is only the mean daily data, and the data at the time of launch could be very different. Moreover, its coordinate resolution is only 2.5°, so the data at the location of the launch site could be very different. However, due to a rocket's long path through the atmosphere, I think the coordinate resolution's effect on the accuracy is somewhat diminished. To connect this dataset to McDowell's, a dataset[3] providing the coordinates for every launch site was used. Together, these data can, as seen in the figure above, be used to find the temperature near any launch site (*e.g.* Cape Canaveral).

Figure 1



1 McDowell, Jonathan. Launchlog. Txt. August 2018. http://planet4589.org/space/log/launchlog.txt
2 "NCEP/NCAR Reanalysis 1: Summary." Noaa.Gov, https://www.esrl.noaa.gov/psd/data/gridded/data.nmc.reanalysis.html. Accessed 6 Feb. 2019.
3 McDowell, Jonathan. Sites. Txt. August 2018. http://planet4589.org/space/log/sites.txt

Preliminary Data Processing

The two fundamental kinds of variables are **categorical** and **quantitative**. Categorical data is data that cannot be represented by a number on a scale. In this investigation, it includes text data like the launch site. Quantitative data is data that *can* be represented by a number on a scale, and includes data like the daily mean temperature. To allow my categorical variables to function in various statistical learning methods, I must, effectively, make them into quantitative variables. This can be done by creating **dummy variables**. A simple example of this is shown below.

Diagram 1

m

Launch	Launch Site
1	Cape Canaveral
2	Baikonur
3	Vandenberg

$\Rightarrow$

m-1

Launch	Cape Canaveral	Baikonur
1	Yes	No
2	No	Yes
3	No	No

The effect of this is that it turns a categorical **predictor** with *m* possible values into *m*-1 quantitative predictors. I was startled at how my creation of dummy variables increased the **predictor space** (the number of predictors) from 3 to 428. As seen in this case, when *m* is large (*e.g.* many different kinds of launch vehicles have been launched), the creation of dummy variables results in many predictors and a very **sparse** (meaning that most values are 0, or "No") set of data. I am concerned by the fact that a sparse set of data creates potential for models to **over-fit** the data. Over-fitting means that the models perform very well on the data they are **trained** on (including any outliers, which don't fit the general trend of the data), but won’t necessarily perform well on any data used to **test** them (precisely because they gave weight to outliers which go against the trend). Furthermore, if, once the data is split into two for training and testing purposes, any of the predictors have a variance of zero, the models will not be able to fit using those predictors. I will have to carefully process the data and tune the models to avoid these limitations.

I made training and test sets by performing a **train-test split** on the original data—a separation which is necessary to accurately evaluate how the model would perform in a real-world scenario. I used a computer program to do this randomly, telling it to place a proportion of 0.7 of the launches into the training set and a proportion of 0.3 into the test set. This resulted in sets of 4011 launches and 1719 launches, respectively. The randomness is important as it makes it probable that the training and test sets will be representative of the whole set of data. The training set may be larger than the test set, because a large training set allows for an accurate model while a test set of 1719 still, as seen in the Model Evaluation section, provides for reliable evaluation of its accuracy.

1) $$Var(X)=\frac{\sum_{i=1}^{n}(x_i-\mu)^2}{n}$$

The above equation gives the **variance** of a set of *n* observations (x_i), that is, their average squared deviation from their mean μ. It is clear that it is possible for the variance to be zero if $\mu = 0$, and all $x_i = 0$. This means that, if, for example, no launches with a particular launch vehicle have occurred in the training set, the variance of that predictor will be zero and the model will not be able to fit to the data. To avoid this issue, I simply removed all predictors that had a variance of zero in the training set.

Over-fitting can occur when the number *n* of observations is low compared to the number of predictors *p* or if some of the predictors are very **sparsely** populated, both of which are the case here. To solve the second issue, I figured I could simply remove all predictors with a total of fewer than 10 positives (arbitrarily chosen, as I could not find a rigorous way to choose the threshold) in all *n* observations. Essentially, predictors were removed if:

2) $$10>\sum_{i=1}^{n}x_i$$

Conveniently, I found that this also effectively solved the other source of over-fitting (*n* low compared to *p*), as it left only 130 predictors with an *n* of 5730.

Model Creation

When predicting an output based on an input, the simplest mathematical model is arguably **linear regression**. It comes in the form:

3) $Y = \beta_0 + \beta_1 X_1 + \ldots + \beta_p X_p$

Y = output	X_p = *p*th input variable
β_0 = intercept	β_p = *p*th coefficient

I looked into this model first because it was the first one I was introduced to, many years ago. It is very effective at predicting a **quantitative** response variable. The first issue I came across with applying it in this investigation is that my response variable is **categorical** (though, for the purpose of computation, a launch success is encoded as a 1 and a failure is encoded as a 0). Thus, the prediction being made is the **probability** of a success, so the notation must be changed:

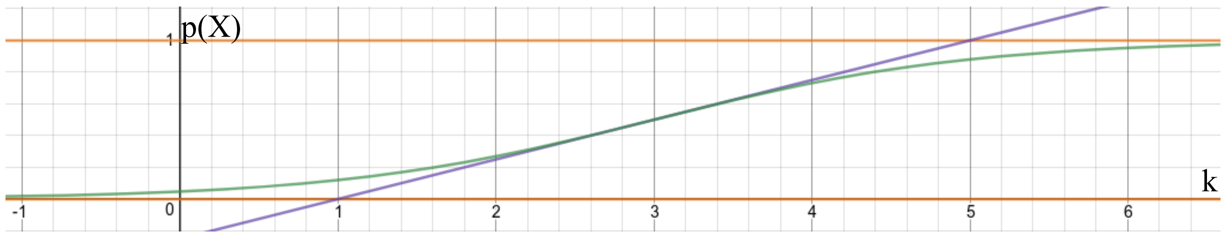
4) $p(X) = \beta_0 + \beta_1 X_1 + \ldots + \beta_p X_p$

This equation, giving probabilities, now technically works for **binary classification**. It will give the probability of being in one class (a launch success) and, because there are only 2 classes, the output can be subtracted from 1 to give the probability of being in the other class (a launch failure). But it is still a *linear* regression, and it is easy to see that it would be possible to output probabilities higher than one or lower than zero, which would not be real-world applicable. I figured that **asymptotes** must be put on the output so that it comes only in the range of zero to one. As $\lim_{k \to -\infty} \frac{e^k}{1+e^k} = 0$, and $\lim_{k \to \infty} \frac{e^k}{1+e^k} = 1$, it is evident that **logistic regression**, in the form of:

5) $p(X) = \frac{e^{\beta_0 + \beta_1 X_1 + \ldots + \beta_p X_p}}{1 + e^{\beta_0 + \beta_1 X_1 + \ldots + \beta_p X_p}} = \frac{1}{e^{-(\beta_0 + \beta_1 X_1 + \ldots + \beta_p X_p)} + 1}$

can be used to keep the probability predictions within the bounds of 0 and 1. This is illustrated in Figure 2, where the asymptotes are orange, the green curve is the logistic regression, and the purple line is the linear regression:

Figure 2



To bring the right side of Equation 5 back to the familiar format seen in the right side of Equation 3 which I've known for years, it can also be written in the form of **odds**:

6) $$\frac{p(X)}{1-p(X)}=e^{\beta_0+\beta_1 X_1+\ldots+\beta_p X_p}$$

Then, the logarithm of the function, *i.e.* the **log odds**, can be obtained, yielding:

7) $$\log\frac{p(X)}{1-p(X)}=\beta_0+\beta_1 X_1+\ldots+\beta_p X_p$$

I have learned in previous studies that linear regression can be expanded to include **non-linear terms**. This familiar format on the right side of the equation makes it clear to me logistic regression can also be expanded to include non-linear terms. For example, I hypothesized that rockets would succeed most often in some range of optimal climate values, making a linear relationship inappropriate. I was able to add a quadratic term by simply including an extra predictor which is the square of the original values. This predictor was given its own **coefficient** and treated just like any other quantitative predictor.

I now have an appropriate equation to make predictions with, but I need to find values for the coefficients. A **likelihood function** must be used to do this. Likelihood refers to how closely a model corresponds to its data, and is computed using the probability that the class outputted by the model is really p if $y_i=1$ or is really $1-p$ if $y_i=0$, where y_i is the known response. Essentially, it gives the probability of observing the real success/failure data if the data is assumed to follow the same pattern as the model. The likelihood function for binary classification is:

8)[4] $$L(\beta_0,\beta)=\prod_{i=1}^{n} p(x_i)^{y_i}(1-p(x_i))^{1-y_i}$$

n = number of observations
y_i = *i*th known response
$p(x_i)$ = output of logistic regression function for *i*th input

I, obviously, want to maximize how closely the model corresponds to the data, and, as such, values for the coefficients must be found that **maximize** the likelihood function. I know that, if I can find a **derivative** of the above function (which would give me its **slope**), I can set it to zero to find a maximum, because maxima always have slopes of 0. However, I am completely unfamiliar with

4 Shalizi, Cosma. "Chapter 12: Logistic Regression." http://www.stat.cmu.edu/%7Ecshalizi/uADA/12/lectures/ch12.pdf. Accessed 6 Feb. 2019.

working with $\prod$ notation. To get around that, I can find the **logarithm** of the likelihood function.

This, conveniently, makes the product of the sequence seen in Equation 8 into sums:

9) $$\log(L(\beta_0,\beta))=\sum_{i=1}^{n}(\log(p(x_i)^{y_i})+\log(1-p(x_i))^{1-y_i})$$

I need to further rearrange this equation before I can derive it:

10) $$\log(L(\beta_0,\beta))=\sum_{i=1}^{n}(y_i\log(p(x_i))+(1-y_i)\log(1-p(x_i)))$$

11) $$\log(L(\beta_0,\beta))=\sum_{i=1}^{n}(y_i\log(p(x_i))+\log(1-p(x_i))-y_i\log(1-p(x_i)))$$

12)[5] $$\log(L(\beta_0,\beta))=\sum_{i=1}^{n}(\log(1-p(x_i))+y_i\log(\frac{p(x_i)}{1-p(x_i)}))$$

What I am really looking for is an equation using the **coefficients**, because I want to find the coefficients that will maximize the function. As such, I need to remove the instances of $p(x_i)$. I recognize that, from Equation 7, $\log(\frac{p(x_i)}{1-p(x_i)})$ is an expression for $\beta_0+\beta x_i$, so:

13)[6] $$\log(L(\beta_0,\beta))=\sum_{i=1}^{n}(\log(1-p(x_i))+y_i(\beta_0+\beta x_i))$$

And, taking $p(x_i)=\frac{1}{e^{-(\beta_0+\beta_i x_i)}+1}$ from Equation 5:

14) $$\log(L(\beta_0,\beta))=\sum_{i=1}^{n}(\log(1-\frac{1}{e^{-(\beta_0+\beta_i x_i)}+1})+y_i(\beta_0+\beta x_i))$$

15) $$\log(L(\beta_0,\beta))=\sum_{i=1}^{n}(\log(\frac{e^{-(\beta_0+\beta_i x_i)}}{e^{-(\beta_0+\beta_i x_i)}+1})+y_i(\beta_0+\beta x_i))$$

16) $$\log(L(\beta_0,\beta))=\sum_{i=1}^{n}(\log(\frac{1}{1+e^{\beta_0+\beta_i x_i}})+y_i(\beta_0+\beta x_i))$$

17)[7] $$\log(L(\beta_0,\beta))=\sum_{i=1}^{n}(-\log(1+e^{\beta_0+\beta_i x_i})+y_i(\beta_0+\beta x_i))$$

5 Shalizi, Cosma. "Chapter 12: Logistic Regression." http://www.stat.cmu.edu/%7Ecshalizi/uADA/12/lectures/ch12.pdf. Accessed 6 Feb. 2019.

6 Ibid.

7 Ibid.

The equation is now finally in a form with only coefficients on the right side. To find the maximum, it must now be **derived** and set to 0. With t as $\log(L(\beta_0,\beta))$, the **partial derivative** of Equation 17 with respect to a component of β, β_j is:

18) $$\frac{\partial t}{\partial \beta_j}=\sum_{i=1}^{n}\left(\frac{-1}{1+e^{\beta_0+\beta x_i}}\times e^{\beta_0+\beta x_{ij}}\times x_{ij}+y_i(0+x_{ij})\right)$$

19) $$\frac{\partial t}{\partial \beta_j}=\sum_{i=1}^{n}\left(\frac{-e^{\beta_0+\beta x_{ij}}}{1+e^{\beta_0+\beta x_i}}\times x_{ij}+y_i x_{ij}\right)$$

I now recognize that, from Equation 5, $\frac{e^{\beta_0+\beta x_{ij}}}{1+e^{\beta_0+\beta x_i}}$ is the expression of the odds function in logistic regression. So this can be rewritten as:

20)[8] $$\frac{\partial t}{\partial \beta_j}=\sum_{i=1}^{n}(y_i-p(x_i;\beta_0,\beta))x_{ij}$$

With $p(x_i;\beta_0,\beta)$ as an expression of said odds function. The exact value of β_j, the jth coefficient in the logistic regression function, would be the **solution** of Equation 20 when set to zero, as that is when the likelihood function is at its maximum. However, this has **no closed-form solution** when set to zero[9] because β_j appears as an argument to the **transcendental** (*i.e.* non-polynomial) function $e^{\beta_0+\beta x_i}$ in the odds function, and the sum is then impossible to factor. I initially thought that this made it impossible to solve for β_j. However, I learned that the value of β_j can be **approximated**, though the numerous methods for doing so—such as stochastic gradient descent[10]—are outside the scope of this investigation. With the large data set involved in this investigation, I used a computer algorithm based on Newton's method to find the values of the coefficients.

8 Shalizi, Cosma. "Chapter 12: Logistic Regression." http://www.stat.cmu.edu/%7Ecshalizi/uADA/12/lectures/ch12.pdf. Accessed 6 Feb. 2019.

9 Ibid.

10 Elkan, Charles. Maximum Likelihood, Logistic Regression, and Stochastic Gradient Training. 10 Jan. 2014, https://cseweb.ucsd.edu/~elkan/250B/logreg.pdf.

Model selection

Now that I have a type of model I can apply to this problem, and a way to estimate its coefficients, I could just get right to fitting a model and predicting the probability of failure. However, I am still concerned about the effects of the numerous predictors on the model's ability to fit to the data. I need to figure out a way for the model to perform **feature selection** when it fits, essentially choosing the best predictors and eliminating useless ones. I think this is necessary due to the very **sparse** data set that I have—some rockets have only been launched 10 times, making those predictors less valuable compared to predictors for rockets launched hundreds of times. One way to perform feature selection is to have the model fit itself repeatedly, choosing the best predictor using the **Analysis of Variance (ANOVA) F-value**. This value identifies which model best fits the data by measuring how much of the variation in the data is accounted for by the model. A simple way to put it mathematically is:

21)[11] $\text{F-value} = \frac{\textit{explained variance}}{\textit{unexplained variance}}$

The formal algorithm I came up with for this process when applied to 130 predictors is:

Algorithm 1

```
1. Begin with $L_0$, a logistic regression model with 0 predictors
2. For $p = 0, 1, 2, \dots 128, 129$:
    a) Train 130-$p$ logistic regression models on the training set, each with a different additional
       predictor
    b) Use the ANOVA F-value (Equation 21) to select from these models the one that best fits
       the test sample, and define it as $L_{p+1}$
```

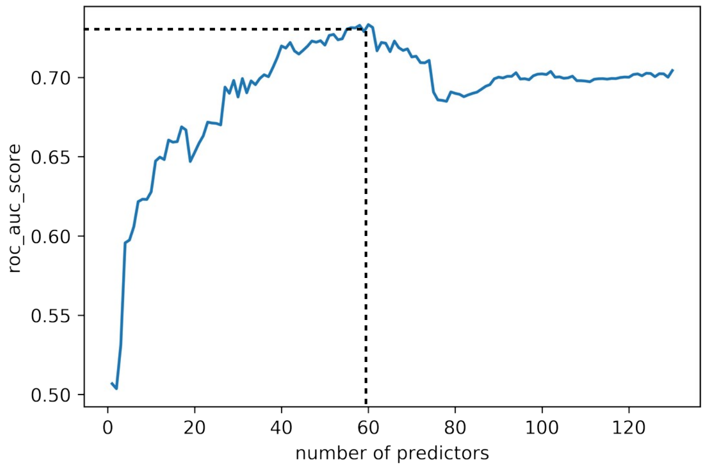
I understand that this algorithm is limited in that, for example, the first predictor added (*e.g.* daily mean temperature) will be valuable at first, but might become completely redundant with the addition of other predictors which are correlated with it (*e.g.* the rocket being a Soyuz rocket or not, as many Soyuz rockets are launched from warm areas). However, an algorithm taking that into account and testing *all* combinations of predictors would take orders of magnitude more time to compute.

11 "F Value." IBM, https://www.ibm.com/support/knowledgecenter/en/SS4QC9/com.ibm.solutions.wa_an_overview.2.0.0.doc/f_value.html. Accessed 6 Feb. 2019.

Now that I have 130 models with varying numbers of predictors, I need to figure out which number of predictors is the best. To do this, I can calculate the area under the **Receiver Operating Characteristic (ROC)** curve for each best model in Algorithm 1 step 2(b) and plot the results. As can be seen in the plot on the right, the optimal number of predictors is around 60 when maximizing for the area under the ROC curve (the formation and properties of which will be discussed later). Only those 60 predictors will be used in the rest of this investigation: any fewer predictors, and they are insufficient to predict accurately; any more, and the model begins over-fitting. It is clear in the plot that this method of using the ANOVA F-value is effective in cutting down the number of predictors while improving the model, but I wondered what additional methods could be used to further improve the model.

Figure 3

Area under model's ROC curve with varying *p*



I realized that what is called a "**cost function**" could be added to the logistic regression optimization function. The ***l*1 penalty** is a cost function which adds a certain "cost" to the model for having higher values on its **coefficients**. This cost forces the optimization function's solution to take into account how large the coefficients are, and thus decreases the size of the coefficients for less useful predictors. The decreased coefficient size also acts as a form of **feature selection**, thus preventing overfitting. This results in decreased **variance** (in this case meaning how much the created model would change if a different training sample were used).

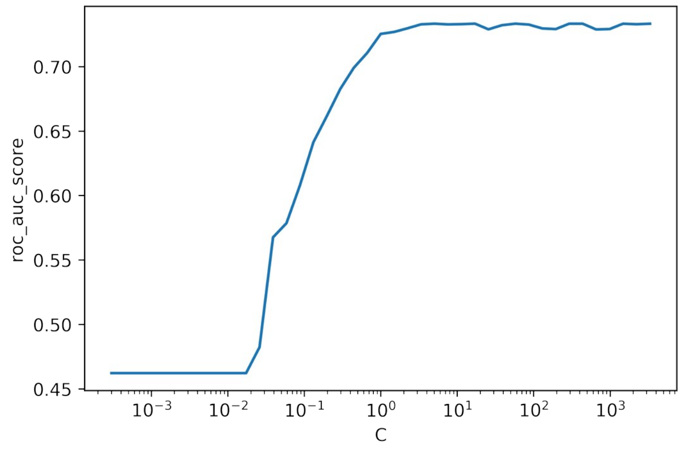
The $l1$ penalty takes the following form:

22)[12] $\frac{1}{C}\sum_{i=1}^{p}|\beta_p|$

The penalty simply sums the absolute values of the coefficients. C is referred to as the **tuning parameter**, because it can be changed to change the strength of the penalty. A small C value will lead to a significant **cost penalty** for high coefficients. At first, I didn't know how to choose a good value for C, and tried plugging in arbitrarily-chosen values. I then realized that I could, like in Figure 3, plot the area under the ROC curve for the different models created.

Figure 4

Area under model's ROC curve with varying C



As can be seen in the figure to the right, low values of C only worsen the area under the Receiver Operating Characteristic curve. I didn't consider this before, but using the $l1$ penalty with a low C would be redundant with the feature selection performed in Equation 2 and Algorithm 1, which already did a sufficiently good job performing feature selection.

The parameters of the final, best model I was able to produce are as follows:

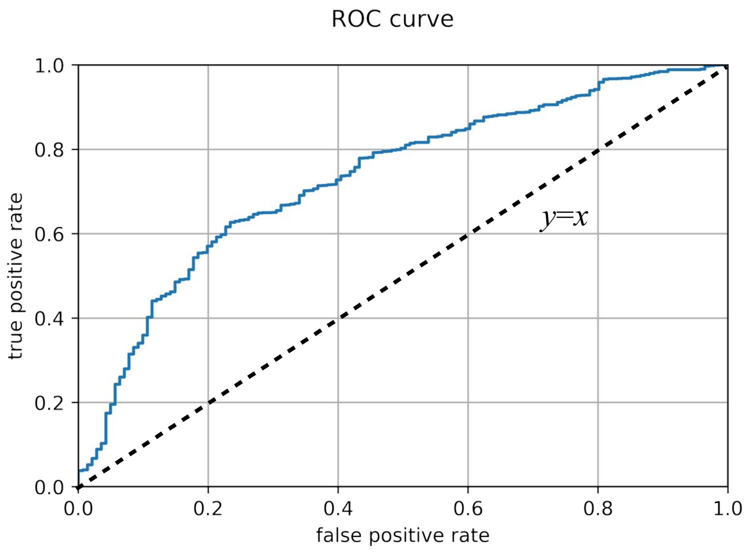
- Logistic regression
- $p = 60$
- C = 1

Coefficients for the various predictors are included in the appendix.

12 James, Gareth, et al. An Introduction to Statistical Learning. Springer New York Heidelberg Dordrecht London, 2013.

Model evaluation

Figure 5



I now have a model, but I wonder how it performs: does it do well on the test data? One method used to evaluate a model in the context of a binary classification problem is the **Receiver Operating Characteristic curve**. Because predictions need not be made solely based on which is class is more probable, the **threshold** can be changed depending on the needs of the data scientist. In this case, a higher threshold would only predict the most probable rocket failures. What I think would be more useful in a practical application is a lower threshold, such that any even slightly risky launches are caught. However, the real test of a good model is being able to fulfill the data scientist's needs at any threshold. To evaluate how good a model is, the ROC curve plots the **false positive rate** (the proportion of rockets that don't fail which are predicted to fail) against the **true positive rate** (the proportion of rockets that fail which are predicted to fail) as the threshold changes. The ROC curve for the model created in this investigation is plotted above. Just looking visually, my model clearly has a larger **area** under it than a random model, which would have an ROC curve along *y*=*x*. A higher area under the curve means that a higher true positive rate can be achieved with a lower false positive rate, so it is clear that my model's higher area means it is superior to a random model. To find the actual areas under the curves, I realized I could use the **integrals** of the two functions. Here, the functions are defined from 0 to 1, so the area under the ROC curve of my model is $\int_0^1 f(x)$, where $f(x)$ is the function for the curve representing my model. Computer calculations equate that integral to 0.733, significantly outperforming a random model whose area I calculated by taking the integral from 0 to 1 of *y*=*x*, giving an area of $\int_0^1 x = {}_0^1[\frac{x^2}{2}] = 0.5$.

Conclusion

I have thus shown that **statistical learning** and basic techniques of "Big Data" can be effectively applied to data on rockets. In terms of the practical applications of the model I created, it could be used by insurance companies which insure satellites to help determine the best rate for a given launch based on various data, like the launch vehicle, launch site, and climate data. However, I think such companies likely already have significantly better models based on much more detailed data, in addition to skilled workers capable of forecasting launch failure rates. The **predictor coefficients** in the Appendix also provide some valuable insights. For example, comparing the coefficient of -2.106 on the Space Shuttle (meaning that the rocket being a space shuttle increases its probability of failure), to a coefficient of 5.894 on the Soyuz-FG, it is easy to see why NASA retired the Space Shuttle and now uses the Soyuz-FG to send astronauts to the ISS. Comparing the coefficient of -1.57 on the Proton-K/D to that of, again, the Soyuz-FG, might suggest that the Proton-K/D should, too, be retired.

No significant **correlation** was found between the climate data and success and failure rates. I think the climate data found was not specific enough to the time and location of launches, making it a relatively useless predictor. For example, the coefficient on the launch vehicle being an Ariane 5 or not is about 700 times the product of the coefficient on temperature and the mean temperature. I think historical climate data specific to individual launches could greatly improve the model, but such data is largely unavailable, in part due to the historical secrecy of some nations' rocket programmes. It is also possible that launch providers simply do a good job at only launching during safe weather, thus leading to little correlation. The correlation seen, however minute it may be, could also be attributed to differing failure rates at launch sites in different climate zones caused by factors like different operators and the different rockets generally used by them.

In hindsight, there were methodological errors in my application of statistical learning. For example, I did not **normalize** the climate data, leading to some redundancy between its mean and the **intercept**, β_0. My methodological errors stemmed from a lack of care, and a rush to build a final model which I found too exciting to wait for. If I repeated this investigation with more care, I could get an even better, more statistically sound, model.

As an extension, a model based on the *properties* of launch vehicles (such as mass, height, payload capacity, etc.), the *properties* of launch sites (such as elevation, max. capacity, etc.), and the same climate data would have a much **denser feature set**, be based on a small number of **quantitative predictors**, and might be able to predict successes and failures of new rockets at new launch sites. The conclusions from this would be more universally applicable, and could provide valuable insight on rocket and launch site design.

Bibliography

Elkan, Charles. "Maximum Likelihood, Logistic Regression, and Stochastic Gradient Training." 10 Jan. 2014, https://cseweb.ucsd.edu/~elkan/250B/logreg.pdf.

“F Value.” IBM, https://www.ibm.com/support/knowledgecenter/en/SS4QC9/com.ibm.solutions.wa_an_overview.2.0.0.doc/f_value.html. Accessed 6 Feb. 2019.

James, Gareth, et al. "An Introduction to Statistical Learning." Springer New York Heidelberg Dordrecht London, 2013.

McDowell, Jonathan. "Launchlog. Txt." August 2018. http://planet4589.org/space/log/launchlog.txt

McDowell, Jonathan. "Sites. Txt." August 2018. http://planet4589.org/space/log/sites.txt

“NCEP/NCAR Reanalysis 1: Summary.” NOAA, https://www.esrl.noaa.gov/psd/data/gridded/data.nmc.reanalysis.html. Accessed 6 Feb. 2019.

Shalizi, Cosma. "Chapter 12: Logistic Regression." http://www.stat.cmu.edu/%7Ecshalizi/uADA/12/lectures/ch12.pdf. Accessed 6 Feb. 2019.

Appendix

Computer-generated array of predictors and their coefficients:

```
[['intercept' '2.304162932188252']
 ['surfacetemp' '-2.6768041498450674e-05']
 ['LV Type__Ariane 1' '-1.3810206834426708']
 ['LV Type__Ariane 44L' '5.584040052128485']
 ['LV Type__Ariane 44LP' '4.9502163355937565']
 ['LV Type__Ariane 44P' '4.420428190978936']
 ['LV Type__Ariane 5ECA' '5.806230404355989']
 ['LV Type__Ariane 5G' '-1.6342425784777634']
 ['LV Type__Atlas D' '-1.0370557311957218']
 ['LV Type__Atlas I' '-1.3714158462212438']
 ['LV Type__Atlas IIA' '5.080910480675659']
 ['LV Type__Atlas IIAS' '5.399553286639319']
 ['LV Type__Chang Zheng 2D' '5.604044310914068']
 ['LV Type__Chang Zheng 2F' '4.610122381363054']
 ['LV Type__Chang Zheng 3A' '5.01585928142424']
 ['LV Type__Chang Zheng 3C' '4.61014996627977']
 ['LV Type__Delta 4M+(4,2)' '4.610712670168166']
 ['LV Type__Delta 7420-10C' '4.610453138418614']
 ['LV Type__Delta 7920-10C' '5.016234708821623']
 ['LV Type__Delta 7925-9.5' '5.252935090358672']
 ['LV Type__Dnepr' '4.71980106035977']
 ['LV Type__Falcon 9' '0.8772986472158648']
 ['LV Type__H-IIA 202' '4.458425064397956']
 ['LV Type__Juno II' '-2.289364961025986']
 ['LV Type__Kosmos 11K65M' '0.12920745729881264']
 ['LV Type__Kosmos 63S1' '-2.2685156478367685']
 ['LV Type__Molniya 8K78' '-1.8590311075619825']
 ['LV Type__Molniya 8K78M' '0.3206429951255642']
 ['LV Type__Pegasus XL' '-0.24284931019334735']
 ['LV Type__Proton-K/D' '-1.5703514447475317']
 ['LV Type__Scout D-1' '4.94720490092654']
 ['LV Type__Soyuz-FG' '5.894152216777368']
 ['LV Type__Soyuz-U' '0.7646929265134461']
 ['LV Type__Soyuz-U-PVB' '0.6662682840519792']
 ['LV Type__Soyuz-U2' '6.158859497412102']
 ['LV Type__Space Shuttle' '-2.0158096386407625']
 ['LV Type__Thor Ablestar' '-1.7787486443435019']
 ['LV Type__Thor Agena A' '-2.1103745527449234']
 ['LV Type__Thor Agena D' '-1.3731728260758171']
 ['LV Type__Thor Delta M' '-0.9036496869005897']
 ['LV Type__Thorad SLV-2G Agena D' '5.017404484933509']
 ['LV Type__Titan IIID' '5.080729568038875']
 ['LV Type__Tsiklon-2' '1.9096114713823782']
 ['LV Type__Vanguard' '-3.3880436598602826']
 ['LV Type__Voskhod 11A57' '0.39843446992034914']
 ['LV Type__Vostok 8A92M' '1.4040853073650617']
 ['LV Type__Vostok 8K72K' '-1.1603382127620951']
 ['LV Type__Zenit-2' '-1.1262534212305033']
 ['Site__CSG' '0.1904010202267069']
 ['Site__GIK-5' '0.2633995883493779']
 ['Site__GNIIPV' '5.402474995867007']
 ['Site__GTsP-4' '0.2609243409009502']
 ['Site__KASC' '-1.4071476768138576']
 ['Site__KSC' '4.344940459189578']
 ['Site__NIIP-5' '-0.030841086034045243']
 ['Site__NIIP-53' '0.5523393919690384']
 ['Site__PA' '-1.601071402446986']
 ['Site__PALB' '-1.189741474687216']
 ['Site__PAWA' '-0.39548252427605995']
 ['Site__SHAR' '-0.4692475381033872']
 ['Site__TNSC' '0.621408127680282']]
```