IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Adam Carrell
10/7/25

Outline

- Executive Summary
- Introduction
- Methodology
- Results
- Conclusion
- Appendix

Executive Summary

Our methodology adheres to an iterative, end-to-end data science lifecycle, beginning with Problem Understanding and rigorous Data Acquisition & Preparation by utilizing SQL and Python's Pandas. Key insights are extracted through Exploratory Data Analysis (EDA) and advanced visualization. The core involves applying diverse Machine Learning (ML) techniques, including classification, regression, and clustering, using Scikit-learn, to build high-performance predictive models. The process concludes with essential Model Evaluation and the principles of operationalizing models for business value. This comprehensive, tool-agnostic approach ensures scalable and actionable data-driven solutions.

The disciplined application of the iterative data science methodology has yielded successful outcomes across multiple projects. We established robust data foundations through rigorous acquisition and preparation using SQL and Pandas. This led to the generation of deep, actionable insights derived from EDA and advanced visualizations. Critically, we successfully developed and evaluated high-performance ML models, including classification, regression, and clustering that consistently meet business objectives and are ready for operational deployment.

Introduction

A new company, SpaceY, is looking to break into the space race theater. My job as a data scientist working for SpaceY was to determine the price of each launch by gathering data, creating dashboards for my team, and to determine if SpaceX would reuse the first stage of their rocket. I used ML models to determine the viability of this.

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Using the open-source SpaceX RestAPI, I collected launch data of several years
- Perform data wrangling
  - I processed the data and converted Outcomes to a y-class for ML viability
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - I built a ML pipeline for a streamlined process that consisted of data preprocessing and splitting the data for test-training to determine the highest accuracy model to employ

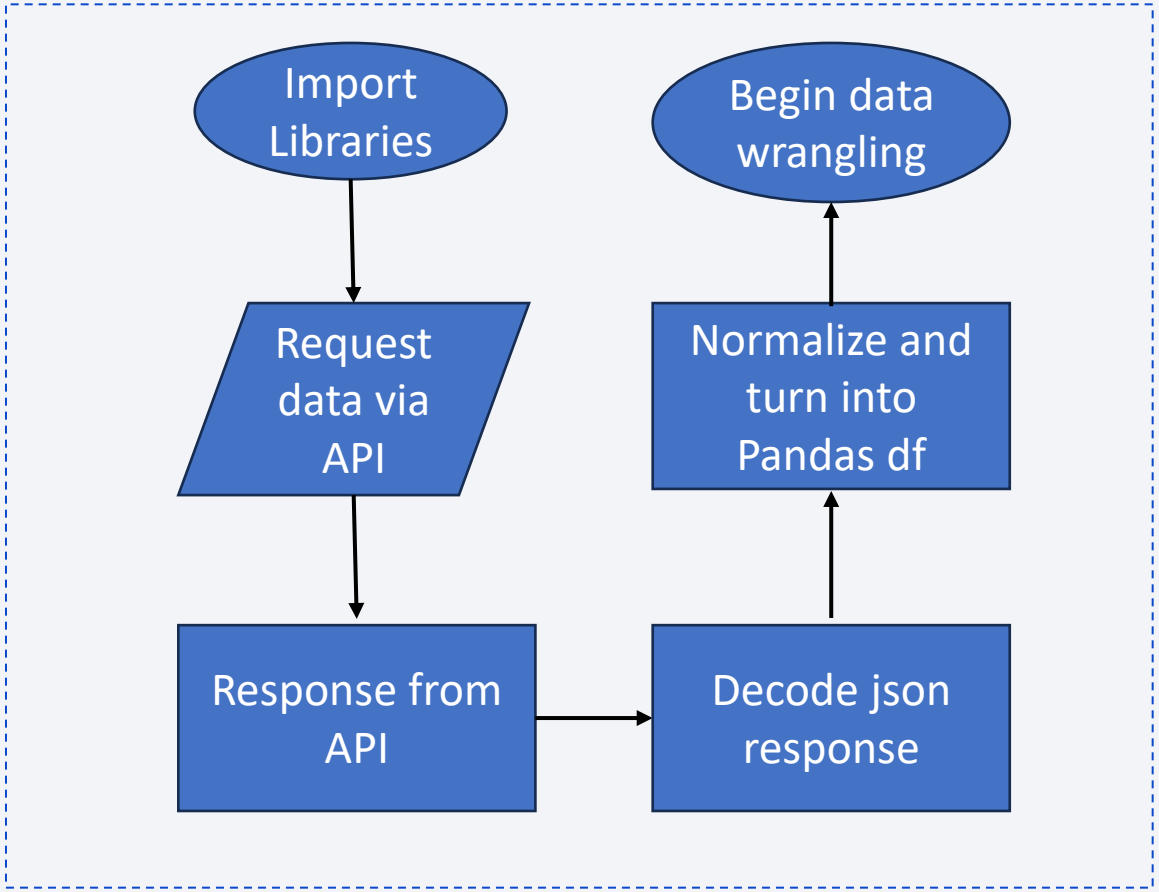
Data Collection

The datasets were collected using a Rest API for the SpaceX website, along with some web scraping. This was accomplished by importing the requests library into our notebook. The data was then requested and parsed into several json files. I used a normalize method to convert the json data into a Pandas data frame. The data frame was then filtered down into relevant columns for easier manipulation and readability. After analyzing the dataset, relevant columns with zero values were filled with an average value. The data frame was then converted to a csv for future use.

Data Collection – SpaceX API

To begin the Data Collection process, I first import the proper libraries, which were requests and pandas for this step. Using the SpaceX API URL, I make my request and get a response, in json format. I then normalize the json data so it can be put into a Pandas data frame.

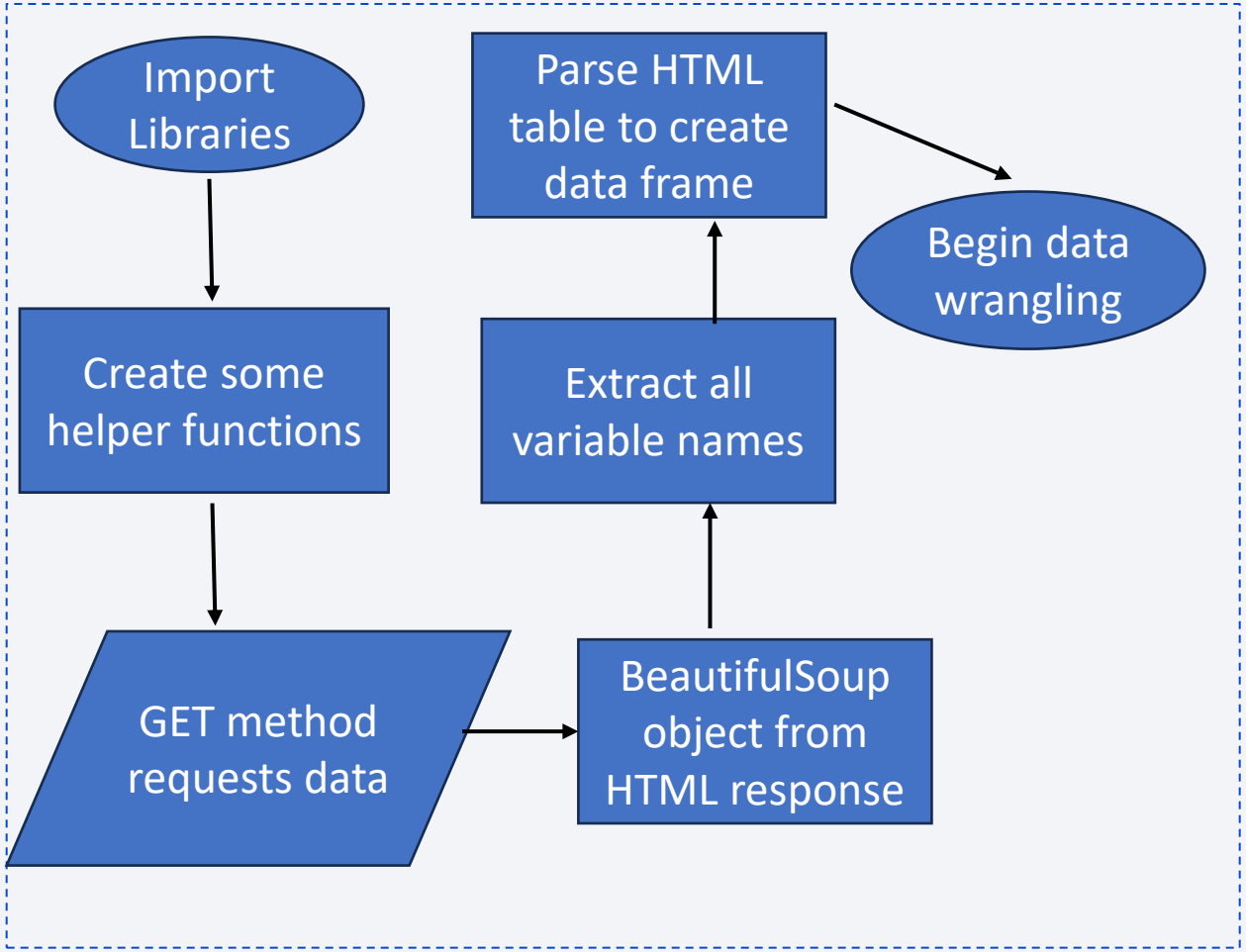
https://github.com/ShagFurnace/Coursera/blob/379ae20f75f02c411d6565c2e5894e761ab939df/jupyter-labs-spacex-data-collection-api.ipynb



Data Collection - Scraping

To begin the Data Collection process, I first import the proper libraries, which were requests and BeautifulSoup for this portion. Some helper functions are then created to assist in grabbing the pertinent data. Using a URL for SpaceX launch data and the GET method, the data is scraped from the site and parsed into a BeautifulSoup object so the data can be easily transformed. Column and variable names are extracted from the HTML table header, and then values are loaded into dictionaries of their respective column.

https://github.com/ShagFurnace/Coursera/blob/379ae20f75f02c411d6565c2e5894e761ab939df/jupyter-labs-webscraping.ipynb



Data Wrangling

To get a better sense of the data, I did some precursory analysis, checking to see which columns are of interest and which, if any, columns need to be transformed before later being used in a model; which columns contained null or blank values, and what the data types were of each column. In this case, the Outcomes column needed to be refined. It had text values that needed to be converted to a simple 0 or 1, 0 for unfavorable outcomes, and a 1 for favorable outcomes. Using the df.dtypes attribute, I was able to see the data type of each variable, noting which were numerical and which were categorical. I then calculated the number of launches at each site, the number and occurrence of each orbit, and the number/occurrence of each mission outcome.

https://github.com/ShagFurnace/Coursera/blob/379ae20f75f02c411d6565c2e5894e761ab939df/labs-jupyter-spacex-Data%20wrangling.ipynb

EDA with Data Visualization

Several scatter plots were used to help visualize the improvement and success growth of the launches as time passed, as well as where these launches took place, the payload in kgs, and the success rates related to orbit destination. In addition to the scatter plots, a bar chart and line chart are also present. The bar chart represents the success rate by orbit destination more clearly than the scatter chart does, while the line chart efficiently shows how the success rate has improved over the years.

https://github.com/ShagFurnace/Coursera/blob/2170b9c9da1e770e2eeaf5580cfd04b13d5c11d0/edadataviz.ipynb

EDA with SQL

- The first query is DROP TABLE IF EXISTS SPACEXTABLE, which empties the table
- Next, we load new data into this table, by selecting values from SPACEXTBL where the Date is not null
- The names of the unique launch sites are shown using a distinct qualifier before Landing_Outcome in our query
- Five records are shown where the launch site begins with ‘CCA’, by adding “launch_site” LIKE ‘CCA%’ limit 5 to the WHERE clause of the query
- Total payload mass carried by NASA launched boosters using SUM(payload_mass__kg_) WHERE customer LIKE ‘NASA%’
- Average payload mass carried by F9 booster using AVG(payload_mass__kg_) WHERE booster_version LIKE ‘F9 v1.1’
- First successful landing date: MIN(Date) WHERE landing_outcome LIKE ‘Success (gro%’
- Total number of specific outcomes: select mission_outcome, count(*) GROUP BY mission_outcome
- All launch sites: select Distinct launch_site from spacextable
- Additional queries that list all booster engines

https://github.com/ShagFurnace/Coursera/blob/2170b9c9da1e770e2eeaf5580cfd04b13d5c11d0/jupyter-labs-eda-sql-coursera_sqllite.ipynb

Build an Interactive Map with Folium

Using the Folium library, I created folium.Circle to add highlighted circle areas for all launch sites. I used folium.map.Marker to add text labels for each launch site circle. I created clusters of color-coded objects that denote each launch and success or failure, focused in the circle that they occurred. I calculated the distances between launch sites and geographic features. These objects help add information, clarity, and points of interest that can guide decision making on future launches.

https://github.com/ShagFurnace/Coursera/blob/e8cfa093b5b236256b1b30d47148c588c68ecfd4/lab_jupyter_launch_site_location.ipynb

Build a Dashboard with Plotly Dash

This Dashboard shows a real-time, interactive view on SpaceX launch data. There is drop-down to select a specific site, a pie chart showing successful launches by site, a scatter chart showing successful launches by payload, and a callback function that updates the pie chart and scatter chart to show successful launches based upon the site selected in the dropdown. The pie chart shows an easy-to-read visual of each sites' success rate of launches, while the scatter chart does a good representation of the success of each launch by payload.

https://github.com/ShagFurnace/Coursera/blob/627c7c4cb921ce369dbd57147ef3ee840b89b296/Plotly%20dash.PNG

https://github.com/ShagFurnace/Coursera/blob/627c7c4cb921ce369dbd57147ef3ee840b89b296/Plotly%20dash%20scatter.PNG

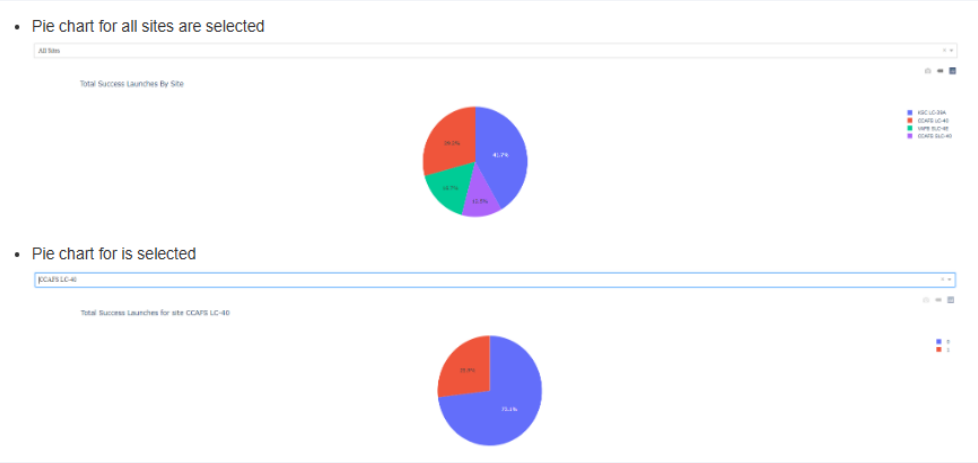
Predictive Analysis (Classification)

To begin, I created a NumPy array using to_numpy() and ensured the output was a Pandas series. I then standardized the data using .StandardScaler() before splitting the data into a train_test_split dataset. Using a given dictionary, parameters, I fit a GridSearchCV object to find the best parameters to use. The tuned (best) parameters are output using the attribute best_params_ on a logistic regression object. I also show the accuracy of this method using the attribute .best_score_. I output another accuracy metric using .score on logreg_cv object, before following that up with a confusion matrix. The confusion matrix is favorable, but has a problem with false positives. I move on to a support vector machine to see how favorable it is with the data. After the svm results, I create a DecisionTreeClassifier() object, then finally move to a k nearest neighbors object. After comparing all the accuracy scores, the best result is the tree model, with an impressive 87% accuracy.

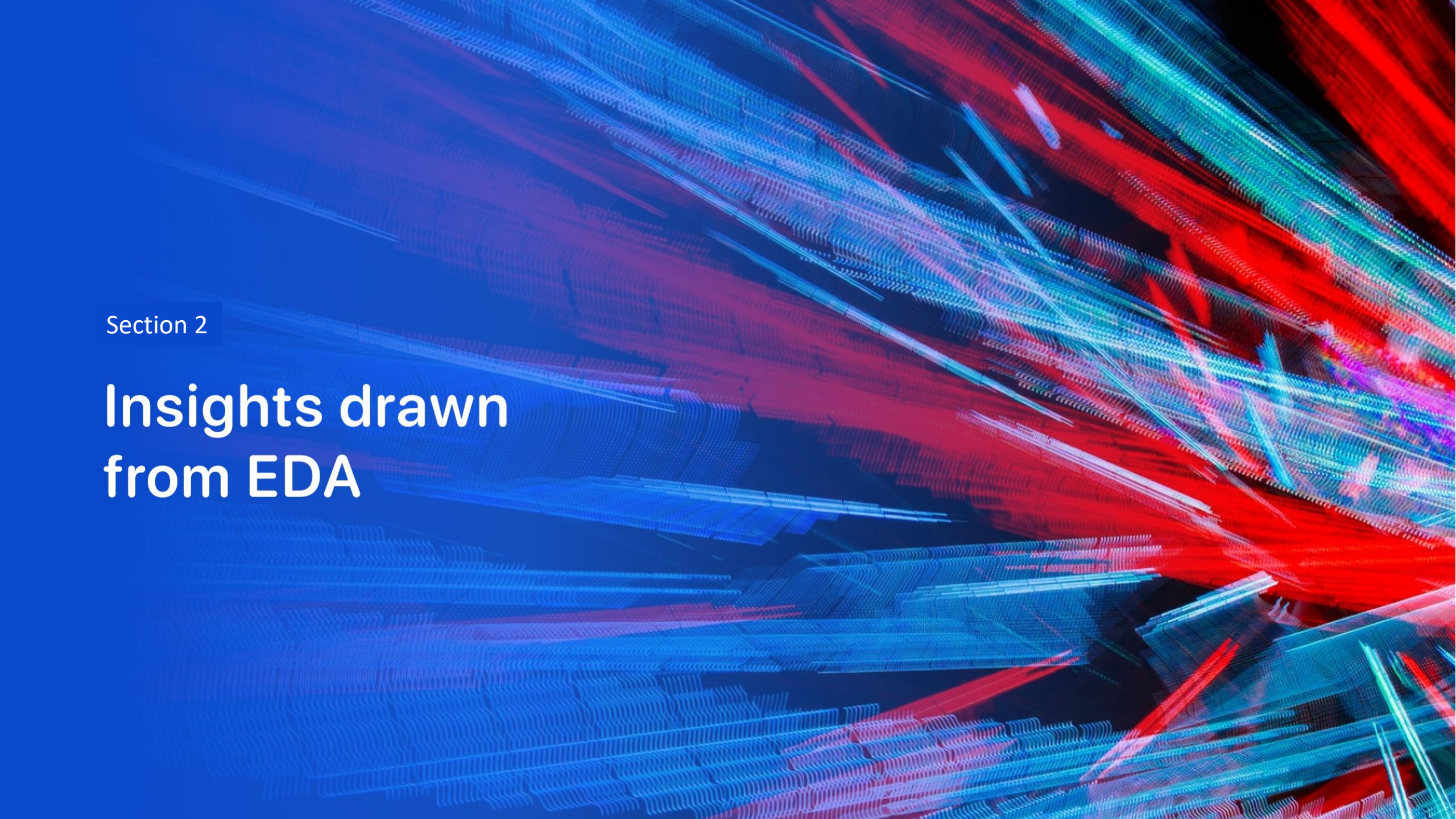
https://github.com/ShagFurnace/Coursera/blob/7a155b19f87330891dd31b77cbe904394991c3f5/SpaceX_Machine%20Learning%20Prediction_Part_5.ipynb

Results

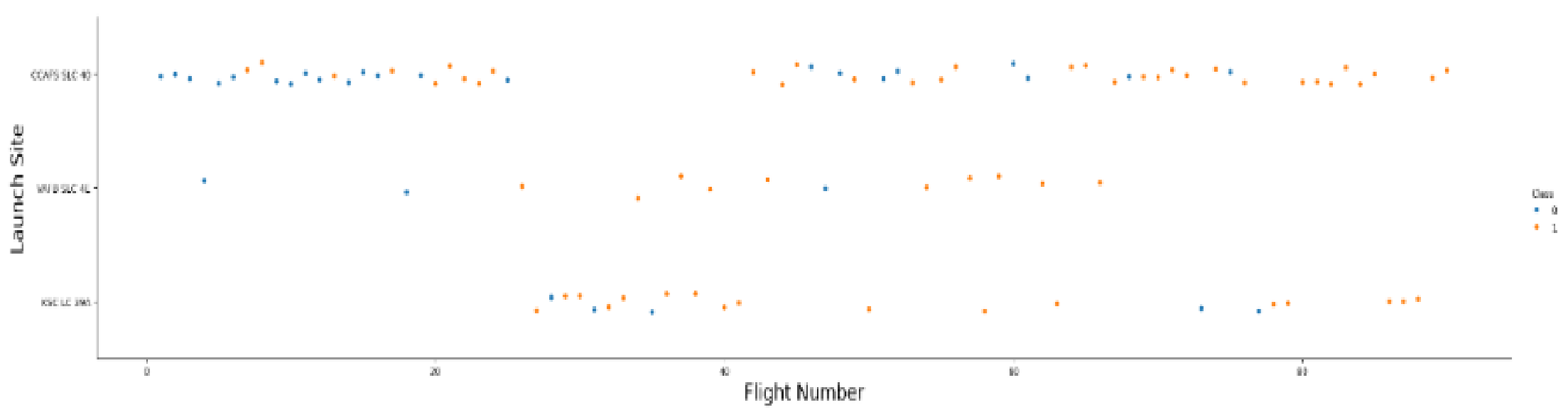
The exploratory analysis sheds light on the success rate trajectory, with various factors taken into account. Payload, orbit destination, and launch site play critical roles in the success or failure of a particular launch. Having this information in hand, we can determine the most advantageous scenario to achieve a successful launch for a given payload.



A Decision Tree Classifier was the model that performed the best on our dataset. With an 87% accuracy, it showed a significant edge over all other models tested in this project, and can be used to predict future launch outcomes.

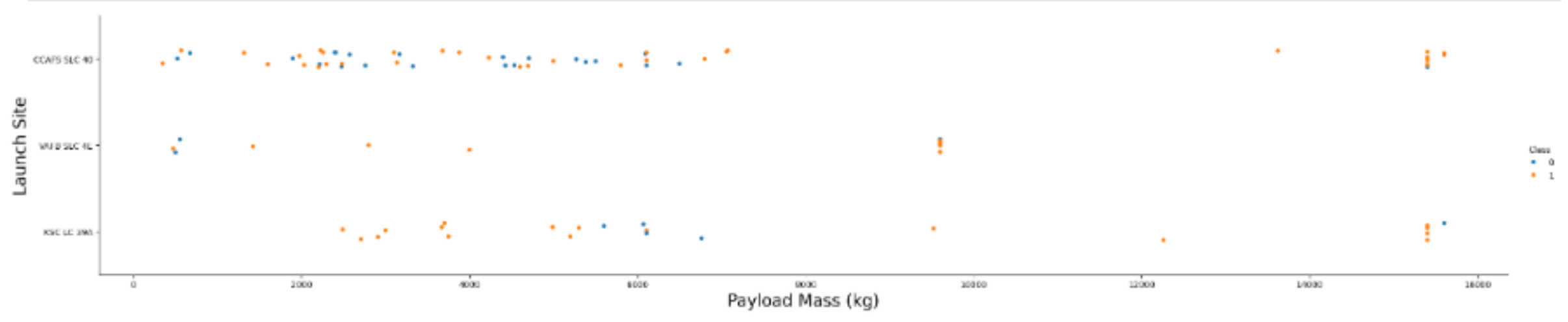

Section 2
Insights drawn from EDA

Flight Number vs. Launch Site



This scatter chart shows how the early flights had a high probability of failure, but were more successful in later flights. The top and bottom launch sites both show some persistent failures, even in later launches, while the middle launch site shows some promising results, having just the two early failures, and just one other, but the sample size is smaller than the other two.

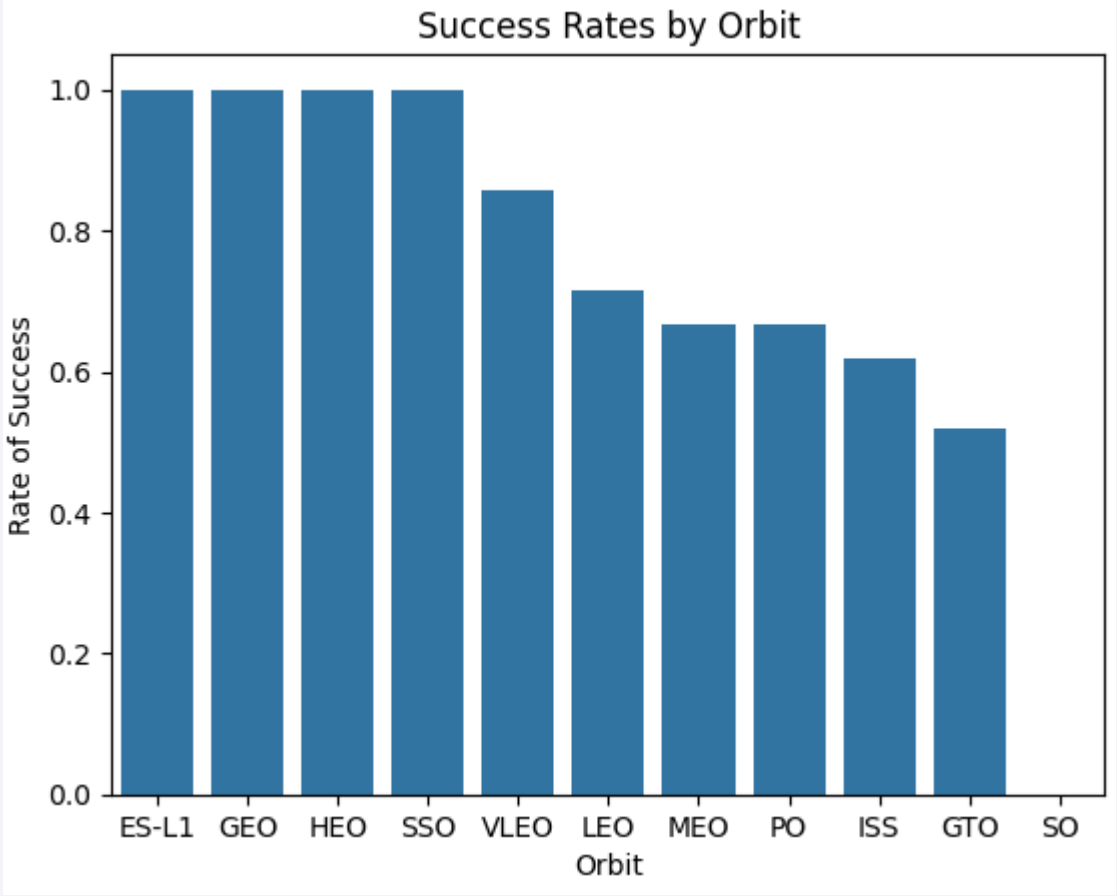
Payload vs. Launch Site



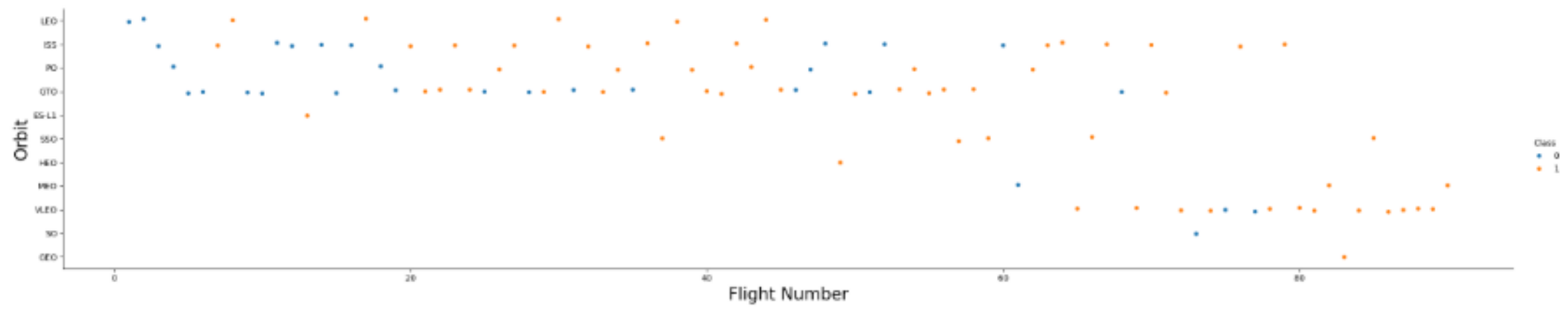
This scatter plot shows a predominantly lighter payload having more frequent launches than heavier payloads. Notice how the failures do not cluster in a single payload mass, but are instead woven throughout the spectrum of the graph. The heavier payloads do show fewer failures, but also much fewer launch attempts as well.

Success Rate vs. Orbit Type

The bar chart shown here represents how successful each launch is by the given orbit destination. Four orbits are shown as having perfect success rates, while GTO is shown as having under a 60% success rate. What contributes to these numbers? Do lighter or heavier payloads primarily go to a specific orbit? Is a particular launch site associated more strongly with a particular orbit? As will be shown in future plots, these questions are not always answered satisfactorily by the given data.

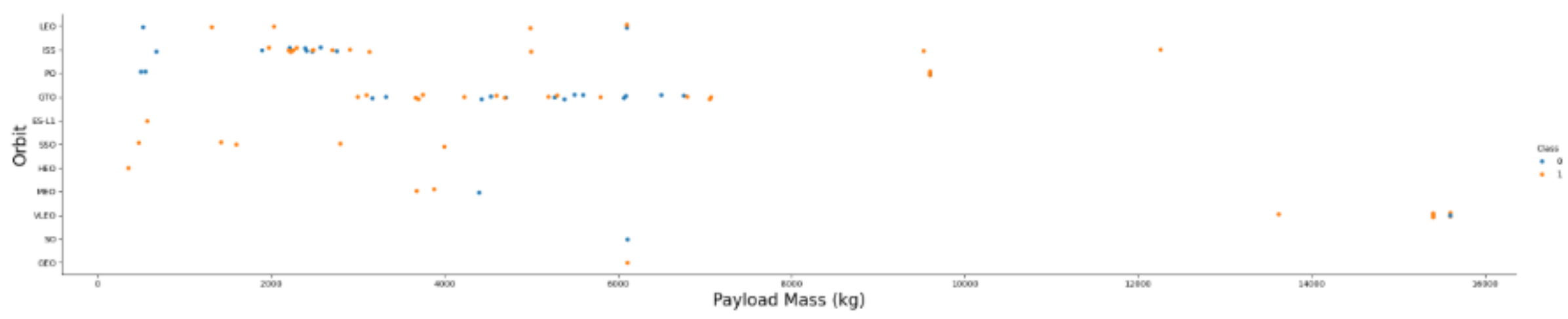


Flight Number vs. Orbit Type



The plot above displays how earlier flights tended to cluster around four of the orbit types. As flights progressed, the destination orbit began to shift away from several of the early dominant destinations, and moved more towards a seemingly favored VLEO, or Very Low Earth Orbit. The failures are also distributed fairly equally between each orbit destination, with the top five orbits each having several failures.

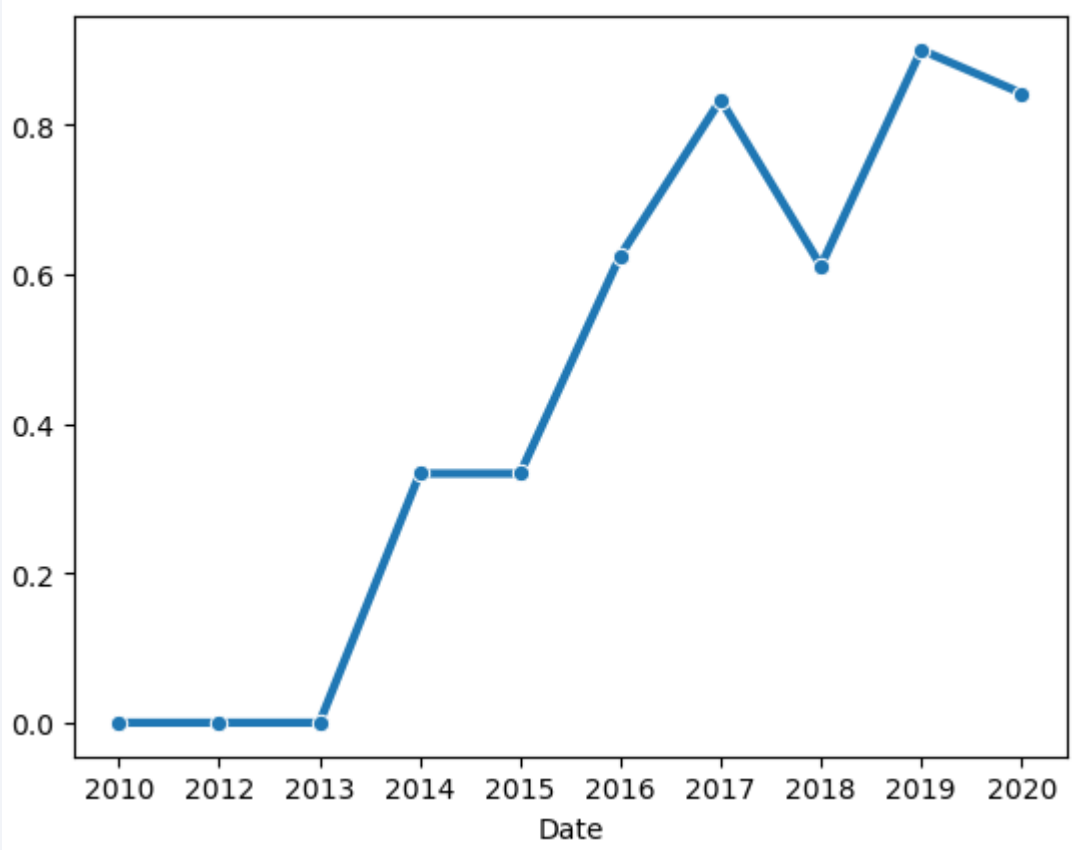
Payload vs. Orbit Type



Here, we get to see how the payload mass affects the success of launching to a particular orbit. Again, we see a scattering of failures among most of the orbits, regardless of payload mass. Some orbits do show nothing but success, but these have a small amount of datapoints and may not be the surest indicators of success.

Launch Success Yearly Trend

This line graph represents the success rate of launches since 2010. It has seen a steady increase YoY, until 2017, when there was a sharp decline, and again in 2020 with a more modest decline. Recall from the previous slides that in later years, different destination orbits began to dominate the launch landscape, so some of this success regression can possibly be attributed to that change.



All Launch Site Names

This query is quite straightforward, selecting all distinct values from the launch_site column that resides in the spacextable. It returns the four unique launch sites, listed as how they appear in the table.

```
In [50]: %sql SELECT Distinct launch_site FROM spacextable
 * sqlite:///my_data1.db
Done.
```

Out[50]:

Launch_Site
CCAFS LC-40
VAFB SLC-4E
KSC LC-39A
CCAFS SLC-40

Launch Site Names Begin with 'CCA'

This query selects all columns from spacextable where the launch_site value is like ‘CCA%’, with the % denoting that any characters may follow the CCA, but the value MUST begin that way. It is ended with limit 5, as we only want to pull in the first five records that satisfy the parameters of this search.

```
%sql SELECT * FROM spacextable WHERE "launch_site" LIKE 'CCA%' limit 5
```

```
 * sqlite:///my_data1.db
Done.
```

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing_Outcome
2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
2010-12-08	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of Brouere cheese	0	LEO (ISS)	NASA (COTS) NRO	Success	Failure (parachute)
2012-05-22	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
2012-10-08	0:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
2013-03-01	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

Total Payload Mass

```
%sql SELECT SUM(payload_mass__kg_) AS "NASA Total Payload" FROM spacextable WHERE customer LIKE 'NASA%'
```

```
 * sqlite:///my_data1.db
Done.
```

NASA Total Payload
99980

This query sums all of the values in the payload mass column from the spacextable that fit the criteria where the customer column value begins ‘NASA%’ where the % signifies that any characters may follow NASA, but must begin with ‘NASA’.

Average Payload Mass by F9 v1.1

```
%sql SELECT AVG(payload_mass__kg_) as Avg_Pyld_Mass FROM spacextable WHERE booster_version LIKE 'F9 v1.1'
```

```
 * sqlite:///my_data1.db
Done.
```

Avg_Pyld_Mass
2928.4

The query above calculates the average of the payload mass column in the spacextable, where the booster_version column is like ‘F9 v1.1’. The returning column is named Avg_Pyld_Mass using the as modifier.

First Successful Ground Landing Date

```
%sql SELECT MIN(Date) AS First_Successful_Landing FROM spacextable WHERE Landing_outcome LIKE 'Success (gro%'
```

```
 * sqlite:///my_data1.db
Done.
```

First_Successful_Landing
2015-12-22

Selecting the minimum from the date column where the landing_outcome value is like ‘Success (gro%’ will return the first successful ground landing date.

Successful Drone Ship Landing with Payload between 4000 and 6000

List the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 60

```
%sql SELECT booster_version FROM spacextable WHERE payload_mass__kg_ > 4000 AND payload_mass__kg_ < 6000
```

```
 * sqlite:///my_data1.db
Done.
```

Booster_Version
F9 v1.1
F9 v1.1 B1011
F9 v1.1 B1014
F9 v1.1 B1016
F9 FT B1020
F9 FT B1022

This query results in a long list, which I have truncated above as they will not all fit, that selects all booster_version values where the payload mass is between 4000 and 6000 kg.

Total Number of Successful and Failure Mission Outcomes

```
%sql SELECT mission_outcome, count(*) FROM spacextable GROUP BY mission_outcome
```

```
 * sqlite:///my_data1.db
Done.
```

Mission_Outcome	count(*)
Failure (in flight)	1
Success	98
Success	1
Success (payload status unclear)	1

The table above shows the count of mission outcomes. Notice the three rows of success and their respective counts. While it is obvious the bottom value text differed from the two above it, the two Success outcomes that look identical likely had a leading or trailing whitespace character that could be read as being a distinct value.

Boosters Carried Maximum Payload

```
%sql SELECT distinct booster_version FROM spacextable WHERE (SELECT MAX(payload_mass__kg_) FROM spacextable
```

```
 * sqlite:///my_data1.db
Done.
```

Booster_Version
F9 v1.0 B0003
F9 v1.0 B0004
F9 v1.0 B0005
F9 v1.0 B0006
F9 v1.0 B0007
F9 v1.1 B1003

This is another long results list, which I truncated. I am selecting distinct values from the booster_version column where the max payload mass is present in the payload_mass column.

2015 Launch Records

%sql SELECT substr(Date,6,2) AS Month, substr(Date,0,5) AS Year, landing_outcome, booster_version, launch_site FROM spacextable WHERE Year = '2015' AND landing_outcome LIKE 'Failure (dro%)'

Month	Year	Landing_Outcome	Booster_Version	Launch_Site
01	2015	Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40
04	2015	Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40

This query is longer than the others, and has more substance. Using the substr function to grab the year from the Date column, along with using it to grab the numeric month from the Date column, all Failure outcomes that occurres on drone ships are retrieved where the year was 2015.

Rank Landing Outcomes Between 2010-06-04 and 2017-03-20

%sql SELECT landing_outcome, count(*) AS Total FROM spacextable WHERE Date between '2010-06-04' AND '2017-03-20' GROUP BY landing_outcome ORDER BY Total DESC

The results to the right show the landing outcomes present in the table. No attempt seems like a fancy way to say ‘we meant for that to blow up’.

Landing_Outcome	Total
No attempt	10
Success (drone ship)	5
Failure (drone ship)	5
Success (ground pad)	3
Controlled (ocean)	3
Uncontrolled (ocean)	2
Failure (parachute)	2
Precluded (drone ship)	1

Section 3

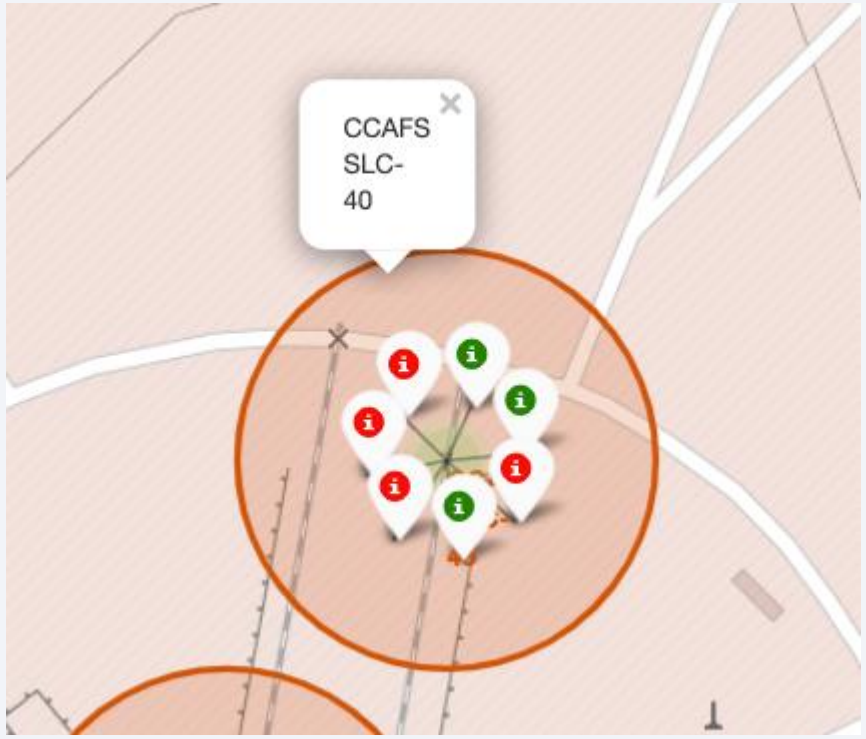
Launch Sites Proximities Analysis

SpaceX Launch Sites

The map screenshot shows one launch site in California, with the others in Florida. The names of the ones in Florida overlap, showing that they are clustered relatively close together.

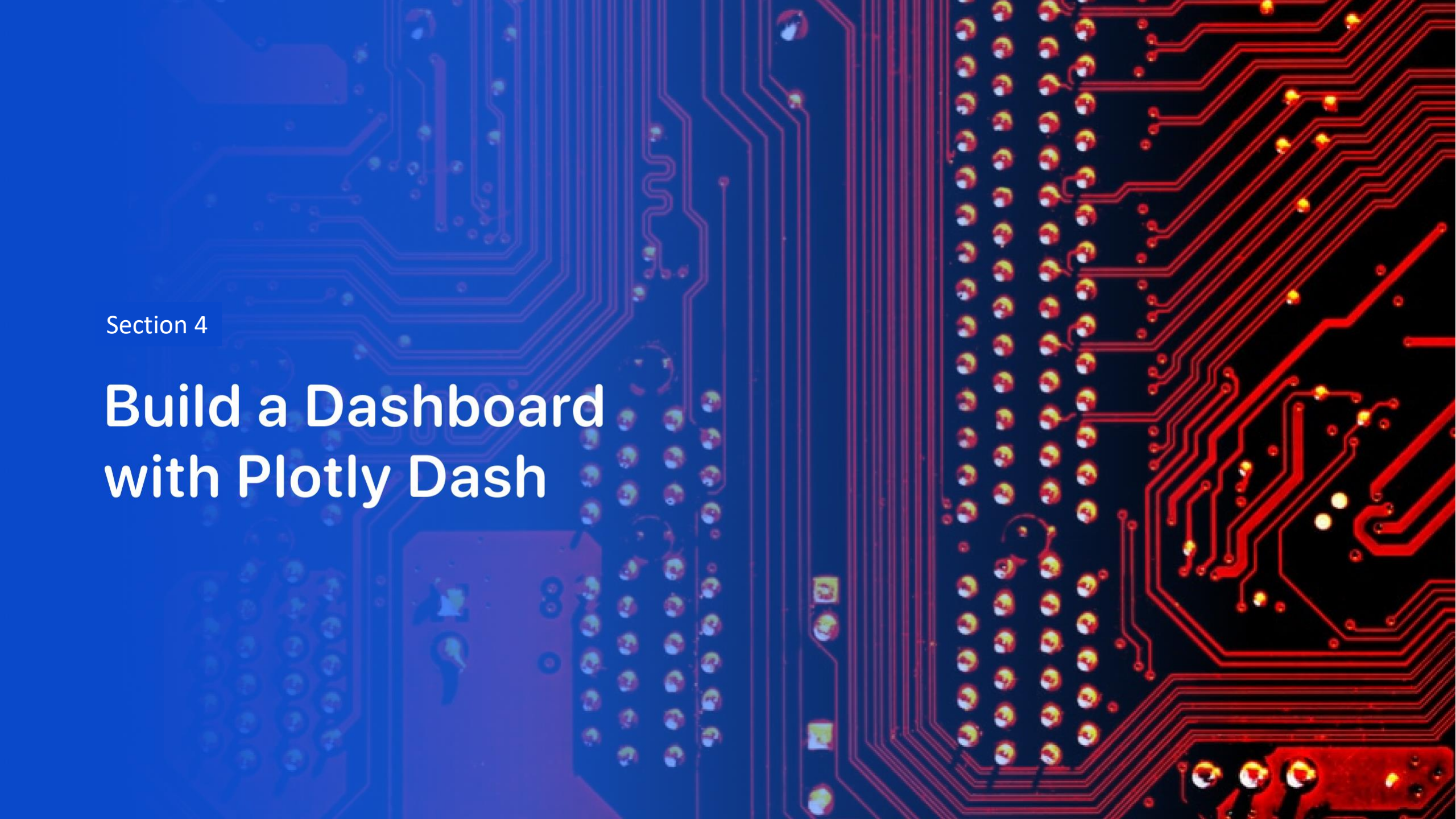
SpaceX Launch Success Rate Map of CCAFS SLC-40

The text identifies the name of the launch site, while the bubble indicators show successful launches in green, with failures in red. Roads and railways are visible in the map details.



SpaceX CCAFS SLS-40 Launch Site Distance to Coast

The above screenshot shows the distance from launch site CCAFS SLC-40 from the Eastern coast of Florida. The distance is displayed as 0.90 KM, which is slightly above half a mile.

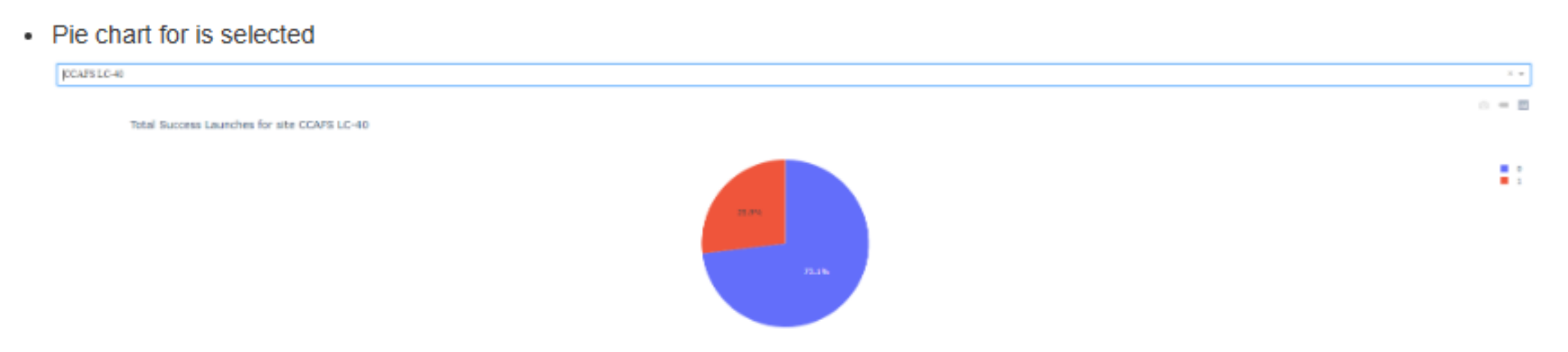

Section 4
Build a Dashboard with Plotly Dash

SpaceX Launch Success Rate All Sites

- Pie chart for all sites are selected

The pie chart shows the four launch sites with their respective success rates. The KSC LC-39A location has the highest success percentage.

SpaceX Most Successful Launch Site



The above pie chart shows the success rate of the selected launch site, KSC LC-39A, and the portion of successful and failed launches.

SpaceX Success Rate By Payload and Launch Site

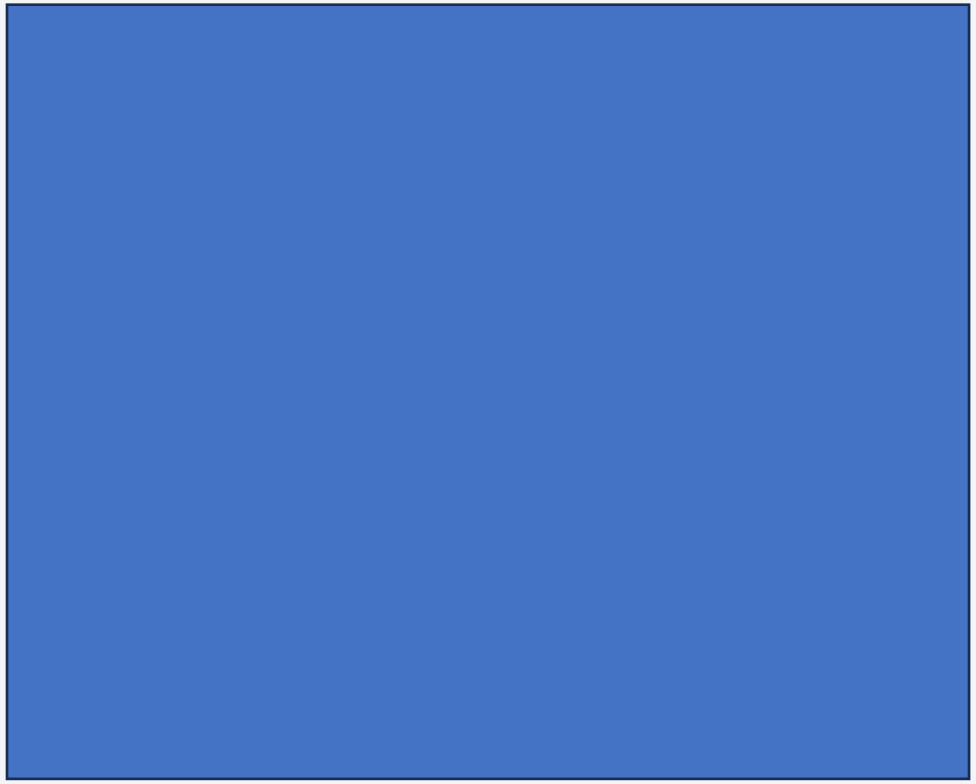
The scatter chart shows the correlation between payload and success by site, with the FT booster version showing the highest success rate.

Section 5

Predictive Analysis (Classification)

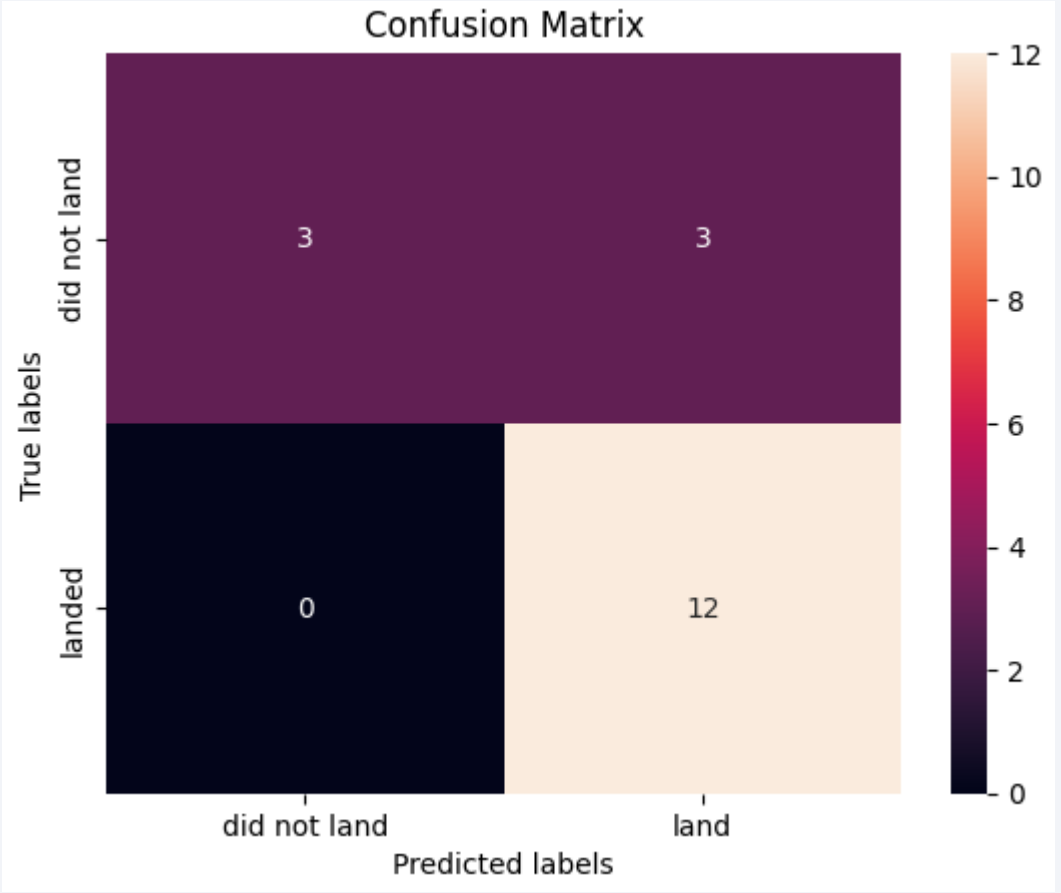
Classification Accuracy

The Decision Tree Classifier gave the highest accuracy scores. With this information in hand, it will be used for future modeling.



Confusion Matrix

This is the confusion matrix for the Decision Tree Model. It shows 12 true positives, 3 true negatives, and 3 false positives.



Conclusions

- The data is not as robust as would normally be preferred to have for these modeling techniques.
- The data does seem to show some trends as far as launch site, payload, and orbit destination is concerned.
- Comparing the five different ML models gave me the best model for the current dataset. Expanding the size of the dataset could alter this.
- Decision Tree Classifier model gave the highest accuracy score in predicting the outcome of a given launch.

Appendix

- Include any relevant assets like Python code snippets, SQL queries, charts, Notebook outputs, or data sets that you may have created during this project

Thank you!