Winning Space Race with Data Science

Rithvik Vinekar
22-Aug-2023

Outline


Executive Summary
Introduction
Methodology
Exploratory Data Analysis
Dashboard
Predictive Analysis
Conclusion
Appendix

Executive Summary

Methodology

Data from SpaceX launch and return was analyzed to build a model. The model can predict successful recovery of launch module after successful launch

Results

The results show that we have a model which shows an accuracy score of 88.75% in predicting the recovery of the launch module

Introduction

- Most space launch vehicles which put satellites into orbit around the world are one-time use vehicles.
- Due to this the cost of most launches are around $165 million or higher
- However, SpaceX has pioneered reusable launchers. The first stage of these launchers return to earth after use and can be reused after landing successfully
- Reuse of launcher vehicles brings down the cost of satellite launches. SpaceX can thus advertise a cost as low as $62 million per launch
- However, not all launch vehicles successfully result in return of the first stage. Multiple factors complicate the successful landing and return of the launcher's 1st stage.
- We can analyse these factors and build a predictive model which will predict if a launch has the factors necessary to be reusable
- This model can later be used to predict a reasonable cost for each type of satellite launch depending on their success rate

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Data was collected using SpaceX APIs and Data scraping from Wikipedia
- Perform data wrangling
  - The status string was classified as Success or Fail (1 or 0)
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - Grid Search Cross Validation was performed for 4 different classification models to estimate the best parameters

Data Collection

Data was collected using two sources:

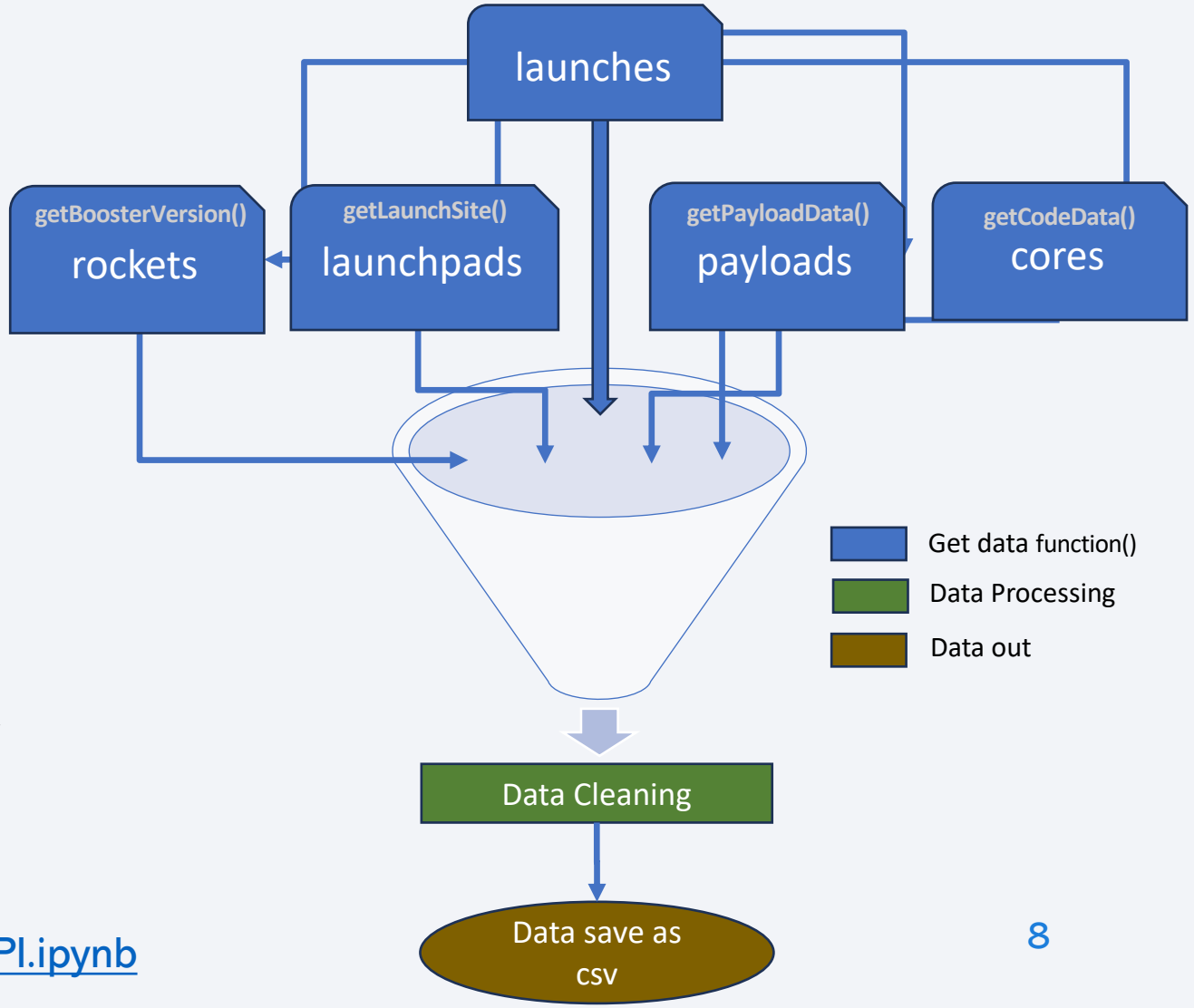
1. SpaceX API. Using the REST protocol, a JSON string was obtained containing requested data
2. Data Scraping : Wikipedia page on the launch status of SpaceX launches was scraped for information

The two data sources can be either combined or one of them used, as there is duplicated information. The REST method was preferred

Data Collection – SpaceX API

- Spacex launch data is obtained from https://api.spacexdata.com/v4/launches/past as JSON data which is then converted to a dataframe
- This dataframe has codes for rockets, launchpads, payloads and cores
- The codes are used to obtain specific data, such as rockets from https://api.spacexdata.com/v4/rockets
- This data from codes and launch data is combined, retaining only relevant fields
- The data is cleaned for missing values and NULLs
  - For example, payload data which is missing is replaced by average payload
- The notebook is available on github here:

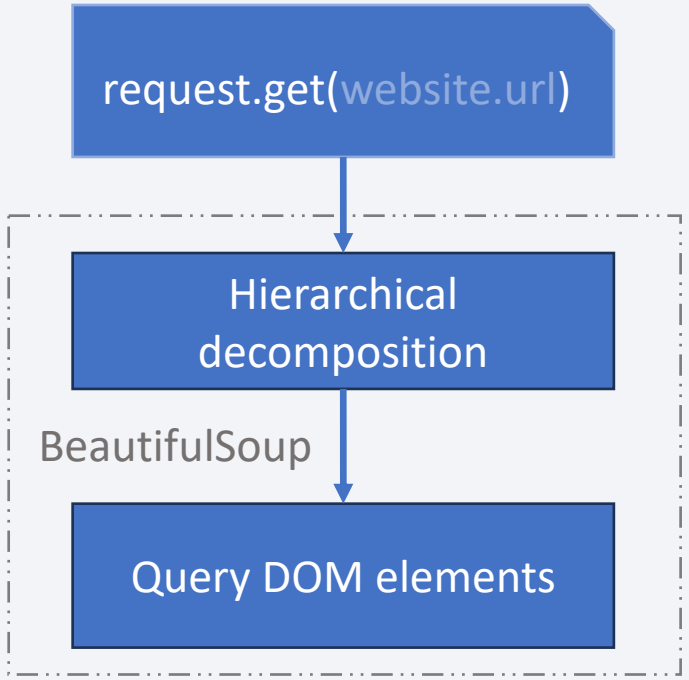
  https://github.com/rsvinekar/Capstone-IBMDS/blob/main/01%20Data%20Collection%20API.ipynb



Data Collection - Scraping

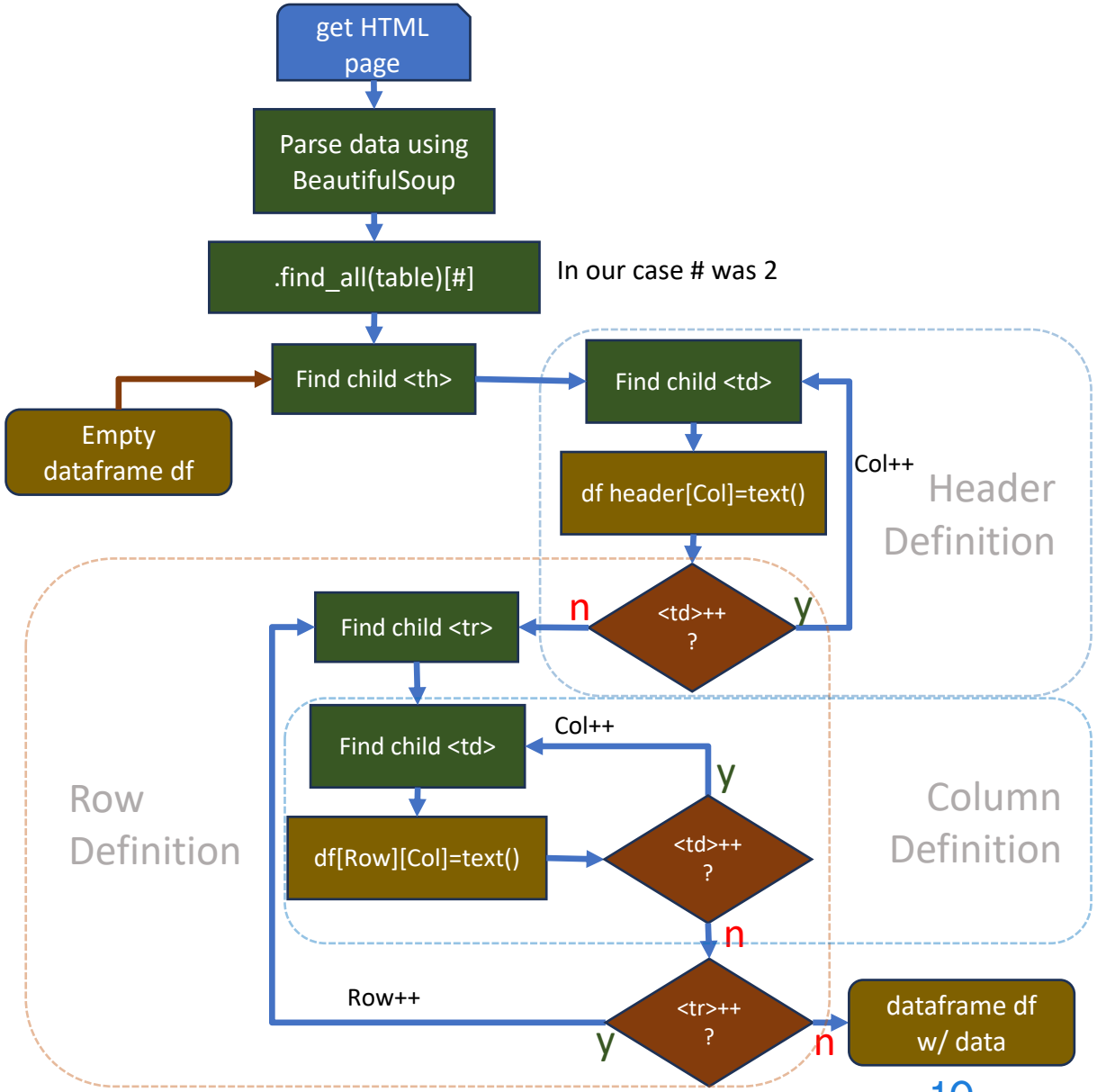
- Website data can be parsed
  - BeautifulSoup (BS4) library in python can be used
- The html website is broken up into hierarchical elements by BS4
- These are accessed in accordance with their level of nesting
- Contents such as text within the html tags can be accessed as data
- GitHub URL:

https://github.com/rsvinekar/Capstone-IBMDS/blob/main/02%20Data%20Collection%20webscraping.ipynb



Data Scraping Logic

- BeautifulSoup breaks down the html data into tag elements
- Tables will be in <table> element that can be located by `.find_all('table')` to get an array of table(s)
- If there are more than one table, they can be accessed as index to array, like `.find_all('table')[2]` to get the 2nd element
- Header row is the <th> element and header columns are under the <th> element as <td> elements
- All other rows are under the <tr> element and <td> columns.
- Nested loops for outer row<tr> and inner column<td>



Data Wrangling

1. The data has NULL values
   - NULL values in Landing pad are removed
2. Orbital types
3. Calculate Number of launches on each site
4. Consolidate the mission outcome
5. Classify the mission outcome as a success (1) or failure(0)

1
```
df.isnull().sum()/df.shape[0]*100
```
```
LandingPad        28.888889
Block              0.000000
```

2
```
df['LaunchSite'].value_counts()
```
```
CCAFS SLC 40    55
KSC LC 39A      22
VAFB SLC 4E     13
Name: LaunchSite, dtype: int64
```

3
```
df['Orbit'].value_counts()
```
```
GTO      27
ISS      21
VLEO     14
PO        9
LEO       7
SSO       5
MEO       3
ES-L1     1
HEO       1
SO        1
GEO       1
Name: Orbit, dtype: int64
```

4
```
for i,outcome in enumerate(landing_outcomes.keys()):
    print(i,outcome)
```
```
0 True ASDS
1 None None
2 True RTLS
3 False ASDS
4 True Ocean
5 False Ocean
6 None ASDS
7 False RTLS
```

5
```
bad_outcomes=set(landing_outcomes.keys()[[1,3,5,6,7]])
landing_class = ~df['Outcome'].isin(bad_outcomes)
landing_class = landing_class.astype(int)
```

EDA with Data Visualization

Exploratory data analysis was done to get trends or correlations:

1. Flight Number vs Launch Site
2. Payload vs Launch Site
3. Success Rate vs Orbit type
4. Flight Number vs Orbit type
5. Payload vs Orbit type
6. Launch Success Yearly trend

https://github.com/rsvinekar/Capstone-IBMDS/blob/main/05%20Exploratory%20Analysis.ipynb

EDA with SQL

Using bullet point format, summarize the SQL queries you performed

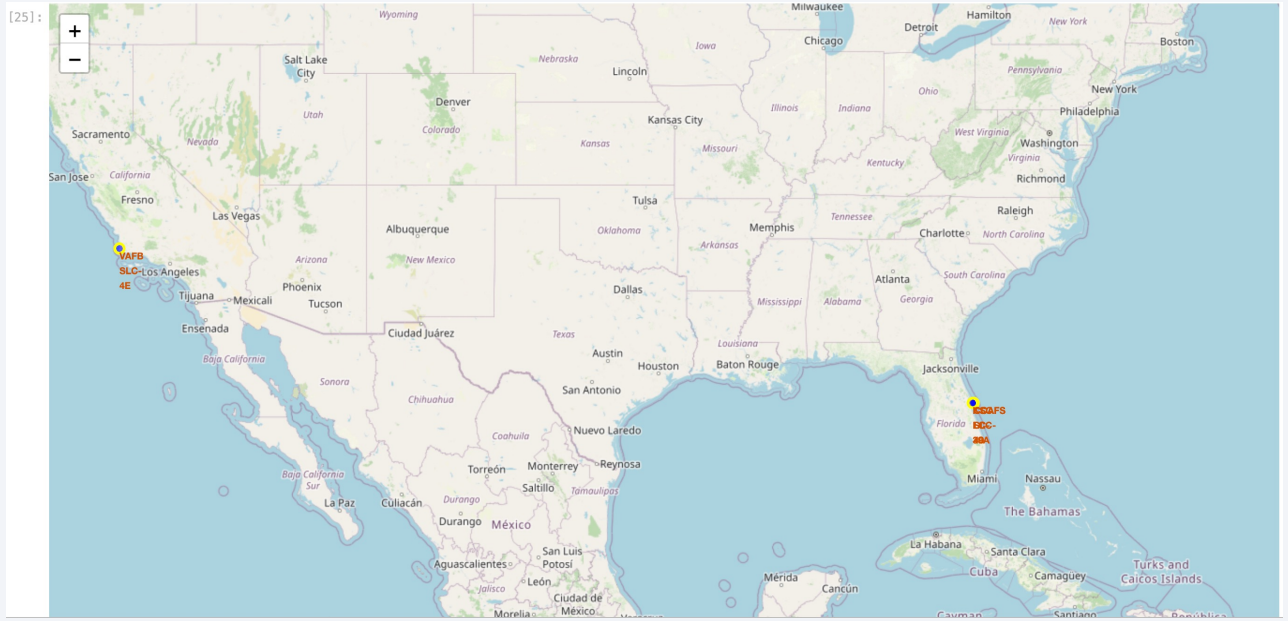
Add the GitHub URL of your completed EDA with SQL notebook, as an external reference and peer-review purpose

https://github.com/rsvinekar/Capstone-IBMDS/blob/main/03%20Database%20querying.ipynb

An Interactive Map with Folium

A Map is made using Folium shows the following points:

1. Icon added, which shows name of the Launch site
2. CircleMarker for each Launch site
3. MarkerCluster to add number for each launch site
4. Colored Icons with success or failure status
5. Polyline to mark nearest coast line, road, rail and major city
6. Distance measures added as icon for these points



https://github.com/rsvinekar/Capstone-IBMDS/blob/main/06%20Maps%20with%20Folium.ipynb

Dashboard with Plotly Dash

The Dashboard is provided which runs with python and dash

The dashboard has two charts

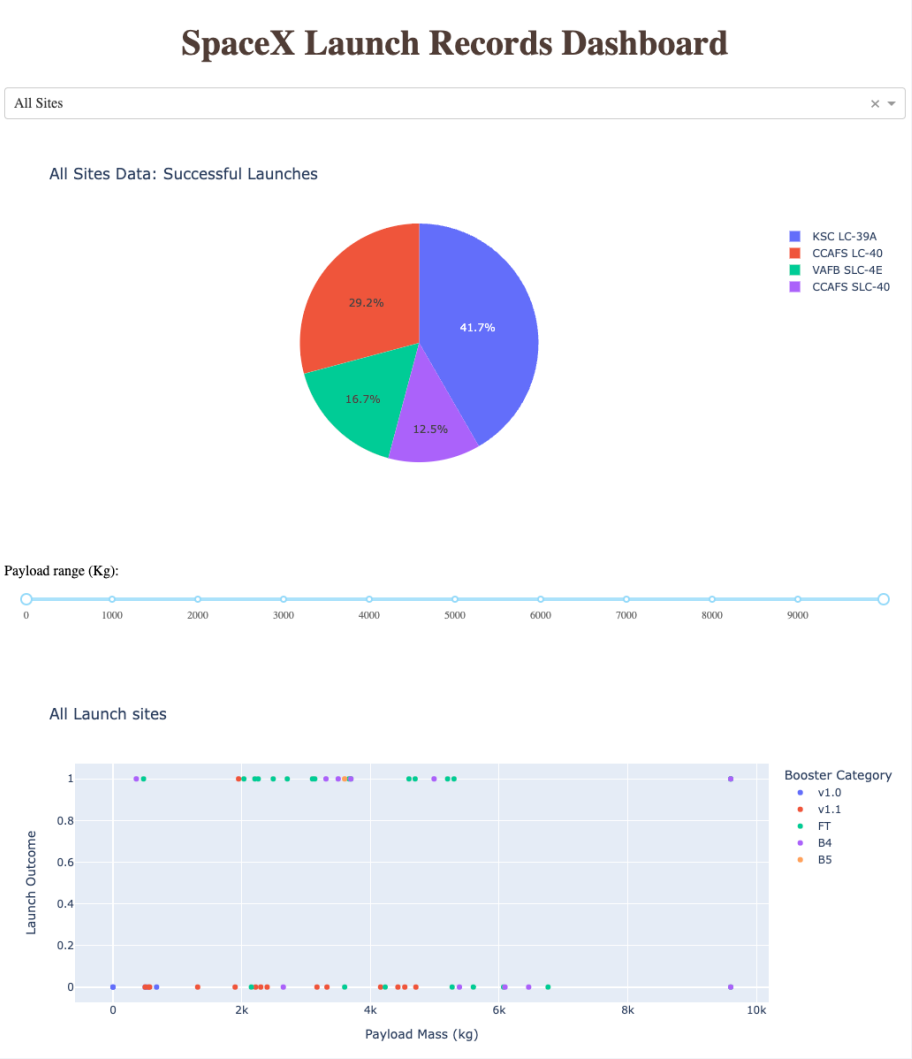
1. A pie chart. A dropdown to choose the Launch site is provided. For Option “All Sites”, it provides a summary of success rates of all sites. Specific sites give success vs failure rate.

2. A scatter plot of Success rate vs Failure of landing vs payload. A Range slider for payload selection is provided

The app link is provided below:

https://github.com/rsvinekar/Capstone-IBMDS/blob/main/spacex_dash_app_dash1.py

Also, please check the Appendix for more details



Predictive Analysis (Classification)

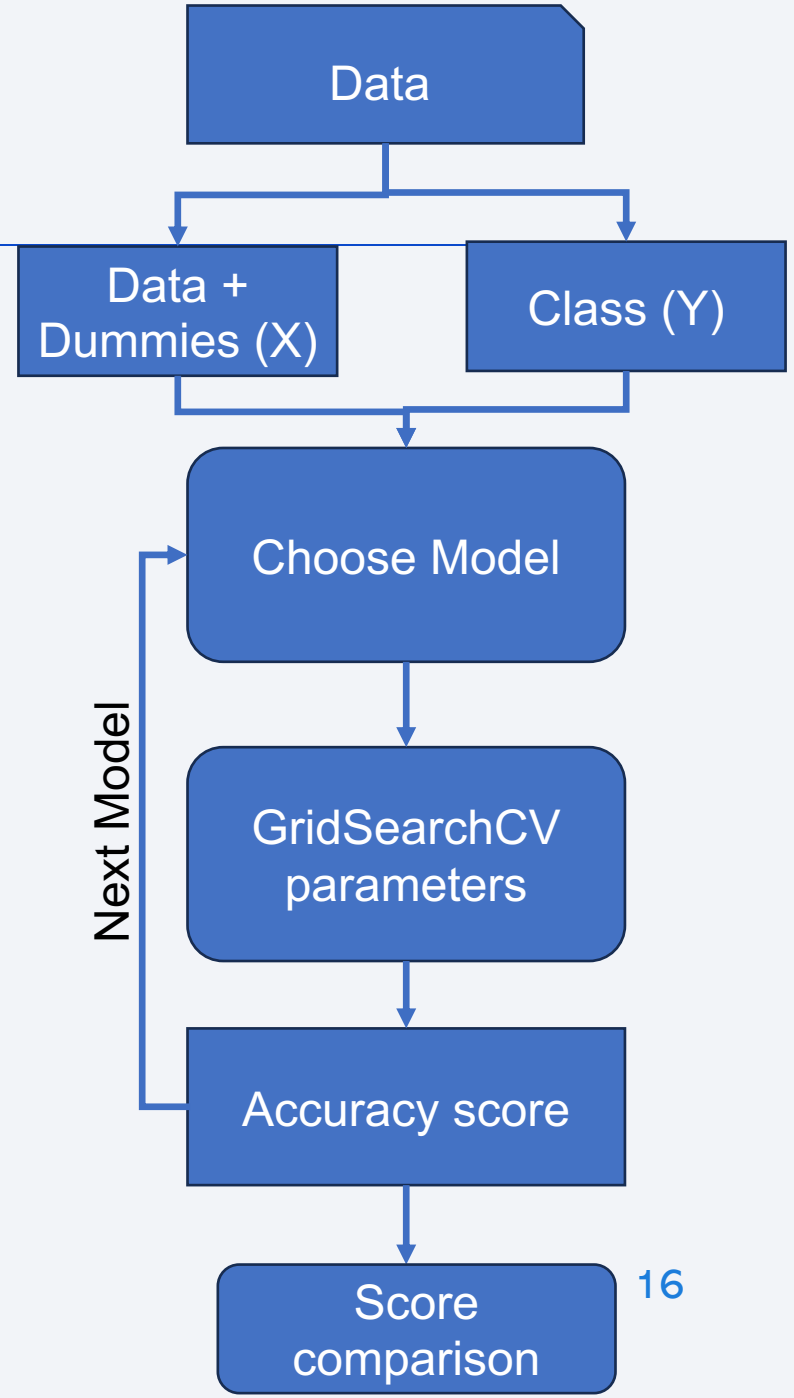
Landing status was interpreted into a class variable (0 or 1 for failure or success) and the data was converted into dummy variables for machine learning

Four different models for machine learning were used to predict the class variable from the other variables available to us. The best parameters were chosen using GridSearch cross validation for each model

Four models were tried:

- logistic regression (logreg)
- Support vector machine (svm)
- Decision trees (tree)
- K-nearest neighbours (knn)

The optimal parameters were determined using Grid Search Cross Validation (GridSearchCV)



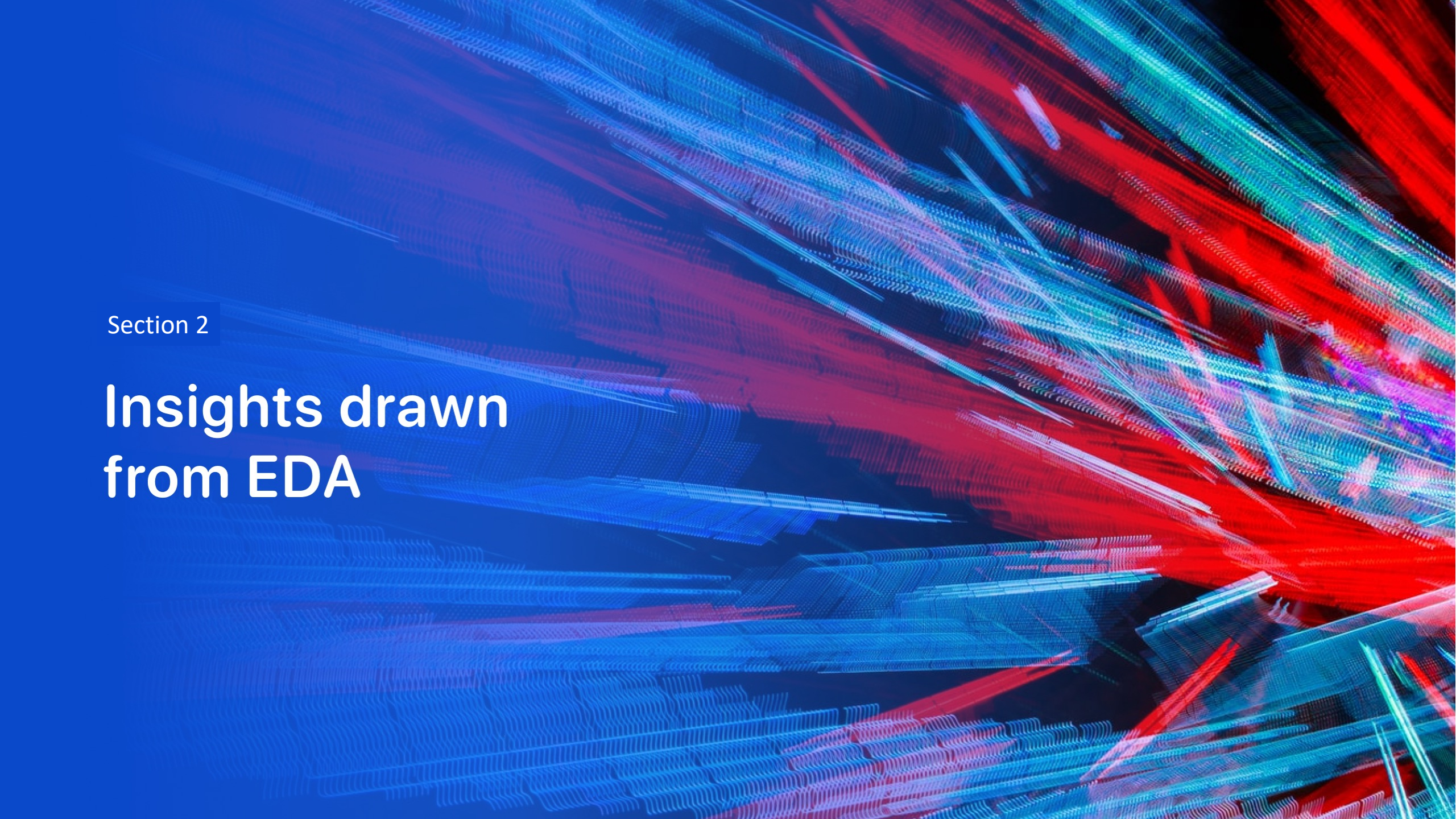
Results

Insights from Exploratory Data Analysis (EDA)

Dashboard Interactive analysis using Plotly and Dash

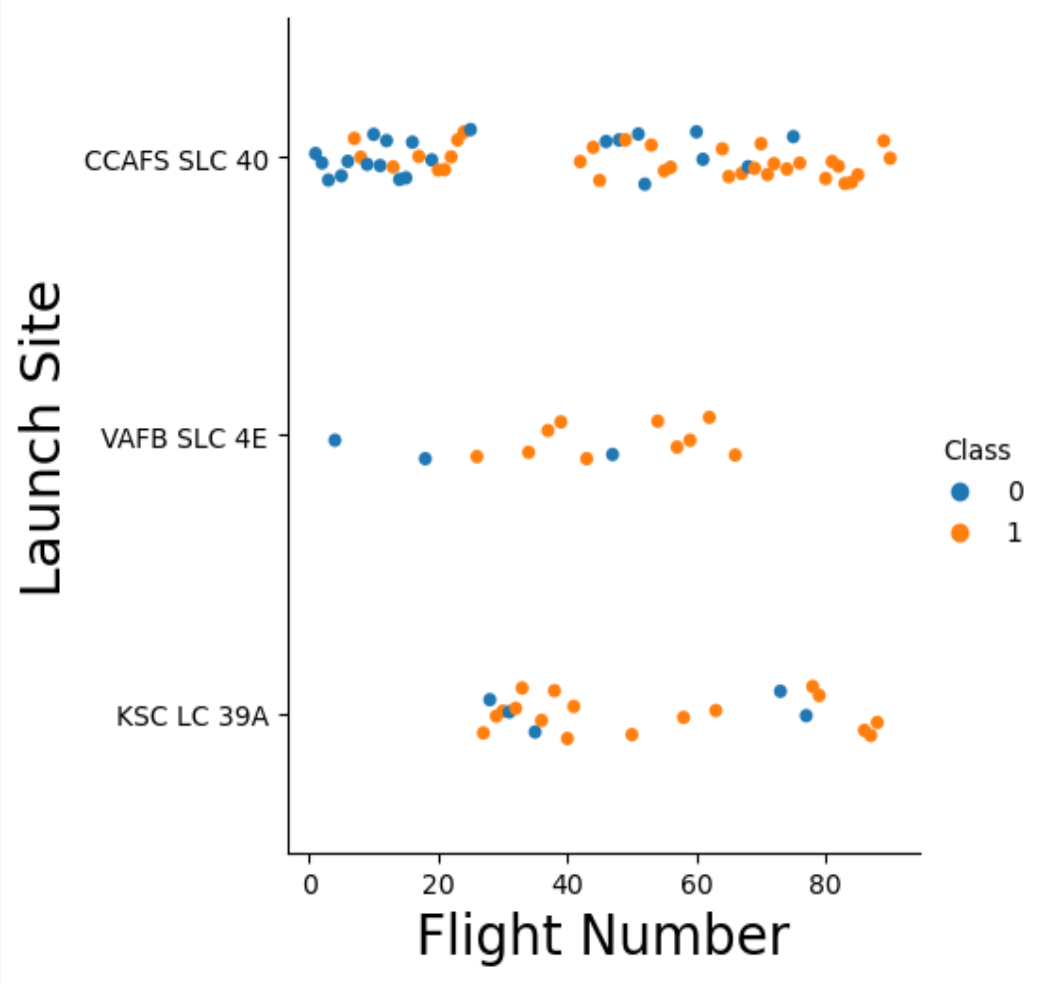
Launch site proximity analysis

Predictive Analysis using Machine Learning


Section 2
Insights drawn from EDA

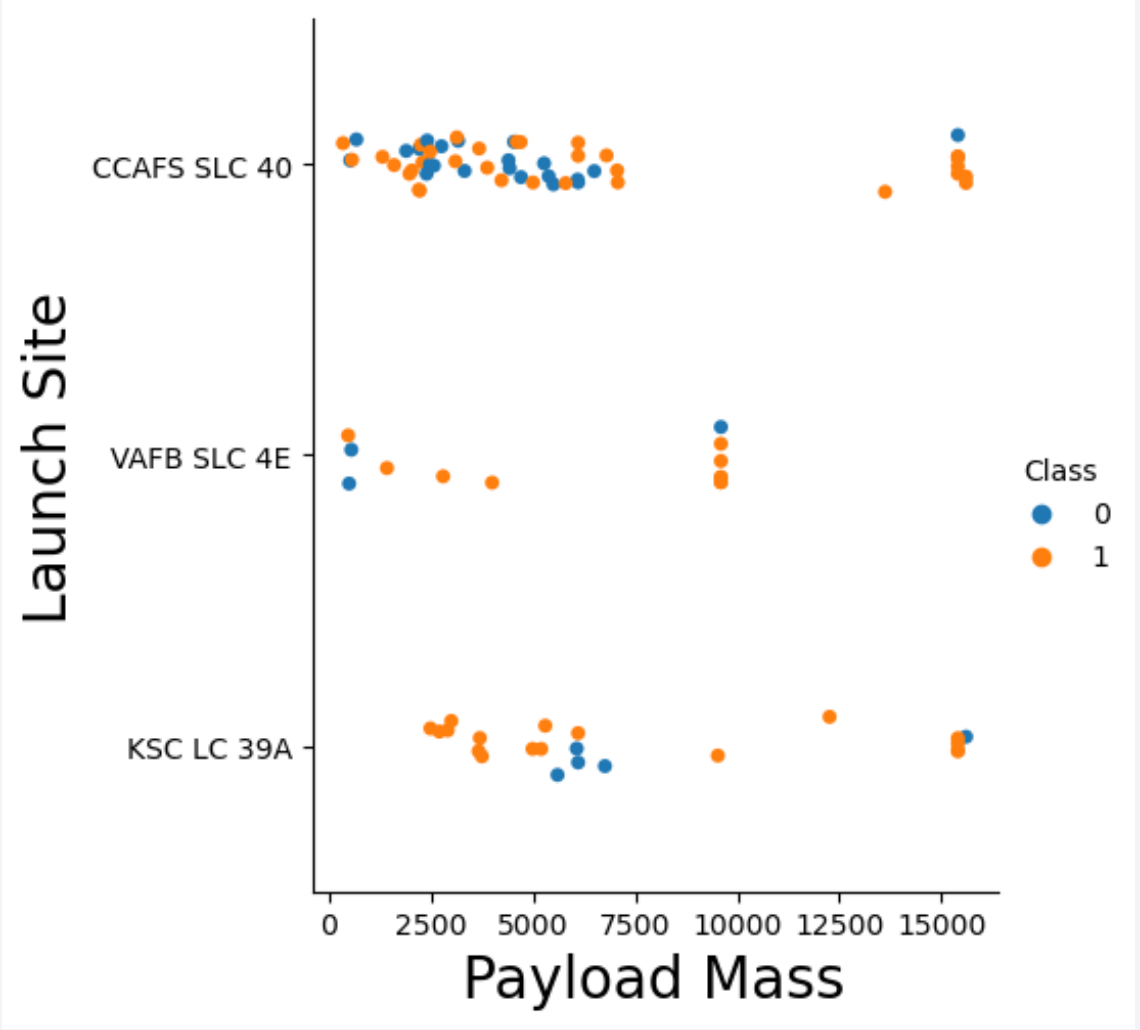
Flight Number vs. Launch Site

- This graph shows the launch sites and the flight numbers, which are roughly correlated with time
- The color is according to landing outcome
  - Class 0 is failure, class 1 is success for landing outcome.
  - In some cases, there was no landing attempted. Here the class is 0 but the mission was successful
- We can see the site CCAFS SLC 40 having the highest number of flights
- Note that in these graphs CCAFS LC 40 and CCAFS SLC 40 refer to the same launch site labeled CCAFS SLC 40, as they are geographically at nearly the same location
- KSC LC 39A site has flight numbers from ~30 onwards



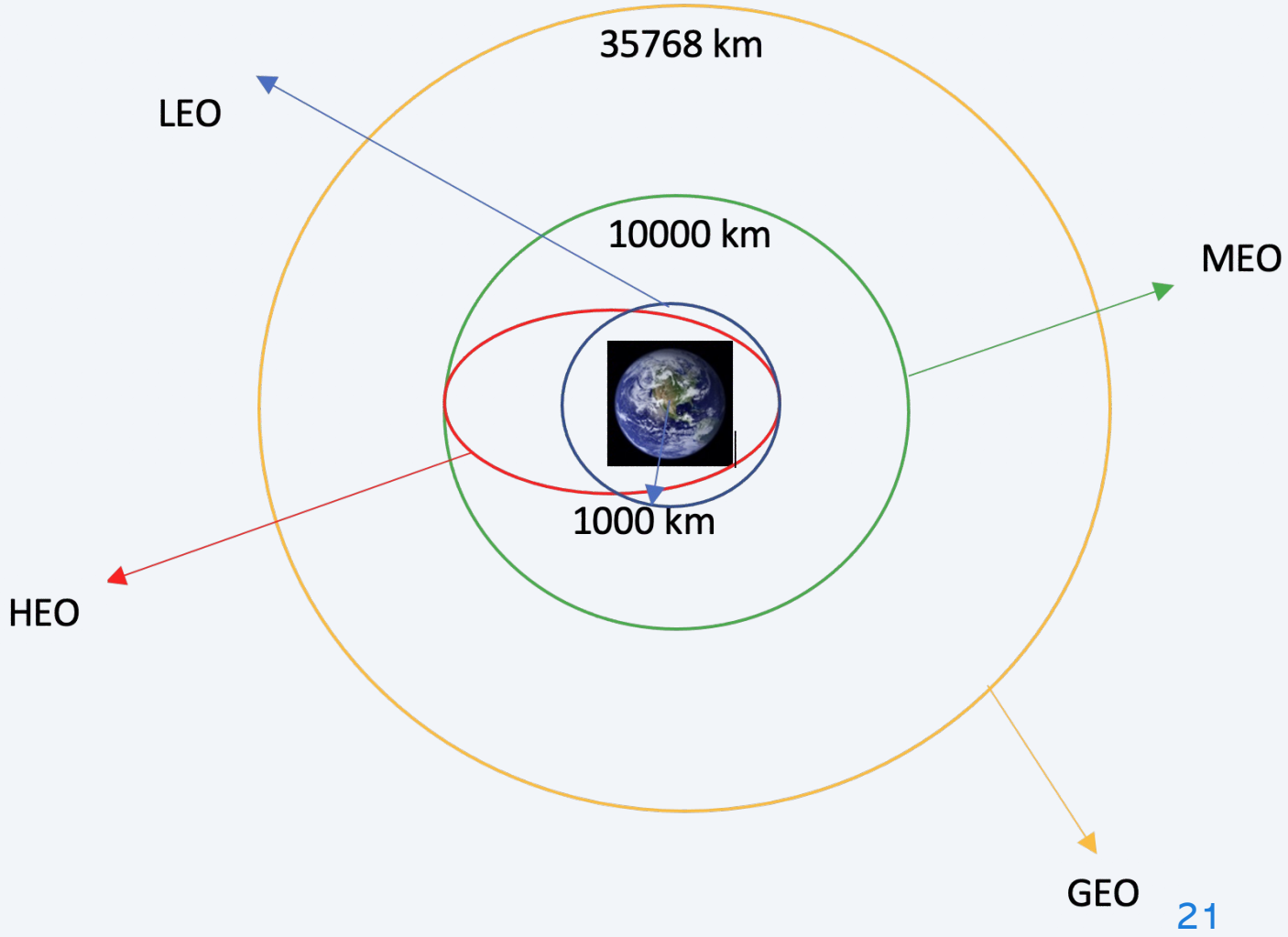
Payload vs. Launch Site

- Most trials are for low payload mass
- VAFB SLC 4E is not used to launch high payload missions



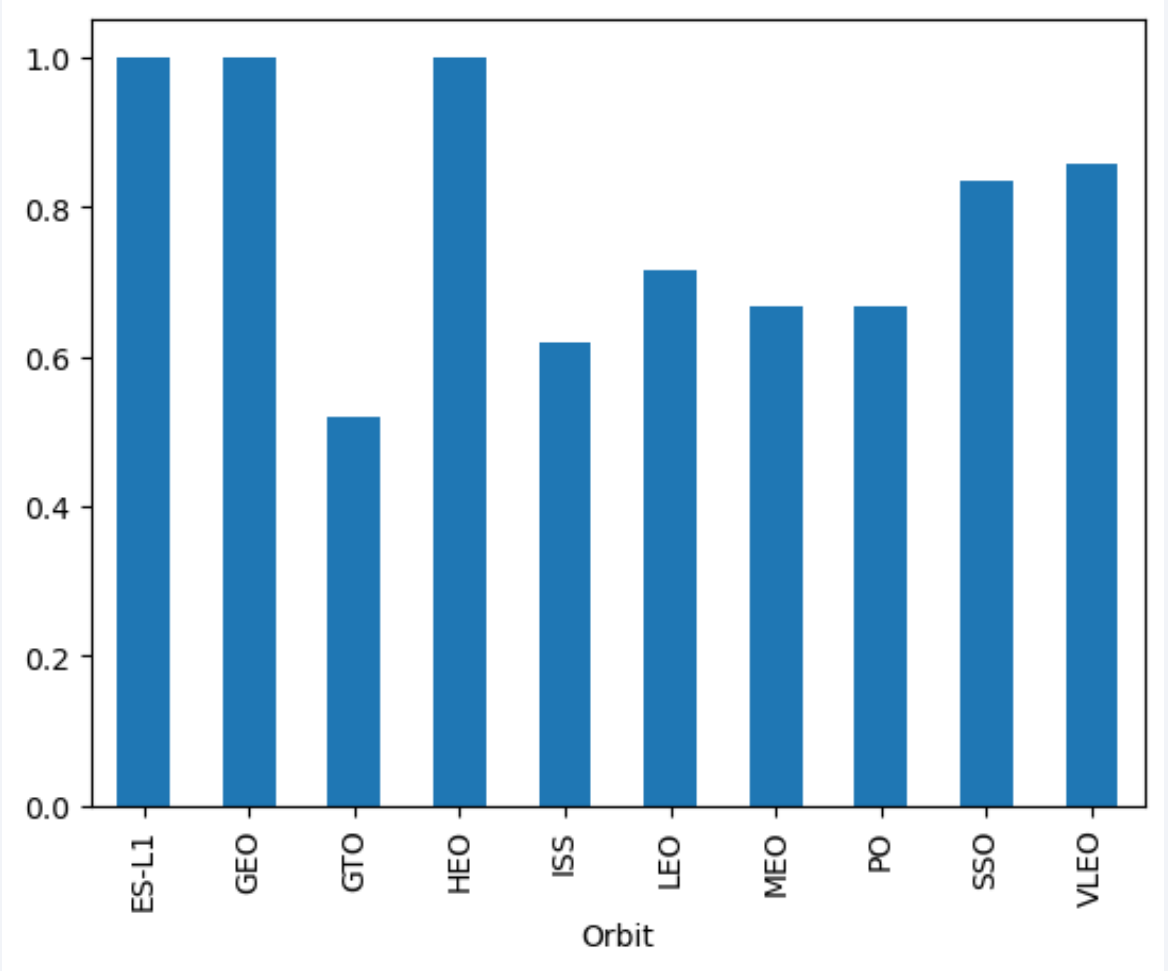
Orbit Type

- **LEO**: Low Earth orbit
- **VLEO**: Very Low Earth Orbits
- **GTO** Geosynchronous orbit
- **SSO (or SO)**: Sun-synchronous
- **ES-L1** : L1, the first Lagrangian Point
- **HEO** A highly elliptical orbit
- **ISS -** International Space station
- **MEO** Geocentric Medium earth orbits
- **HEO** Geocentric orbits High Earth orbit
- **GEO** Circular geosynchronous orbit
- **PO** Polar orbit



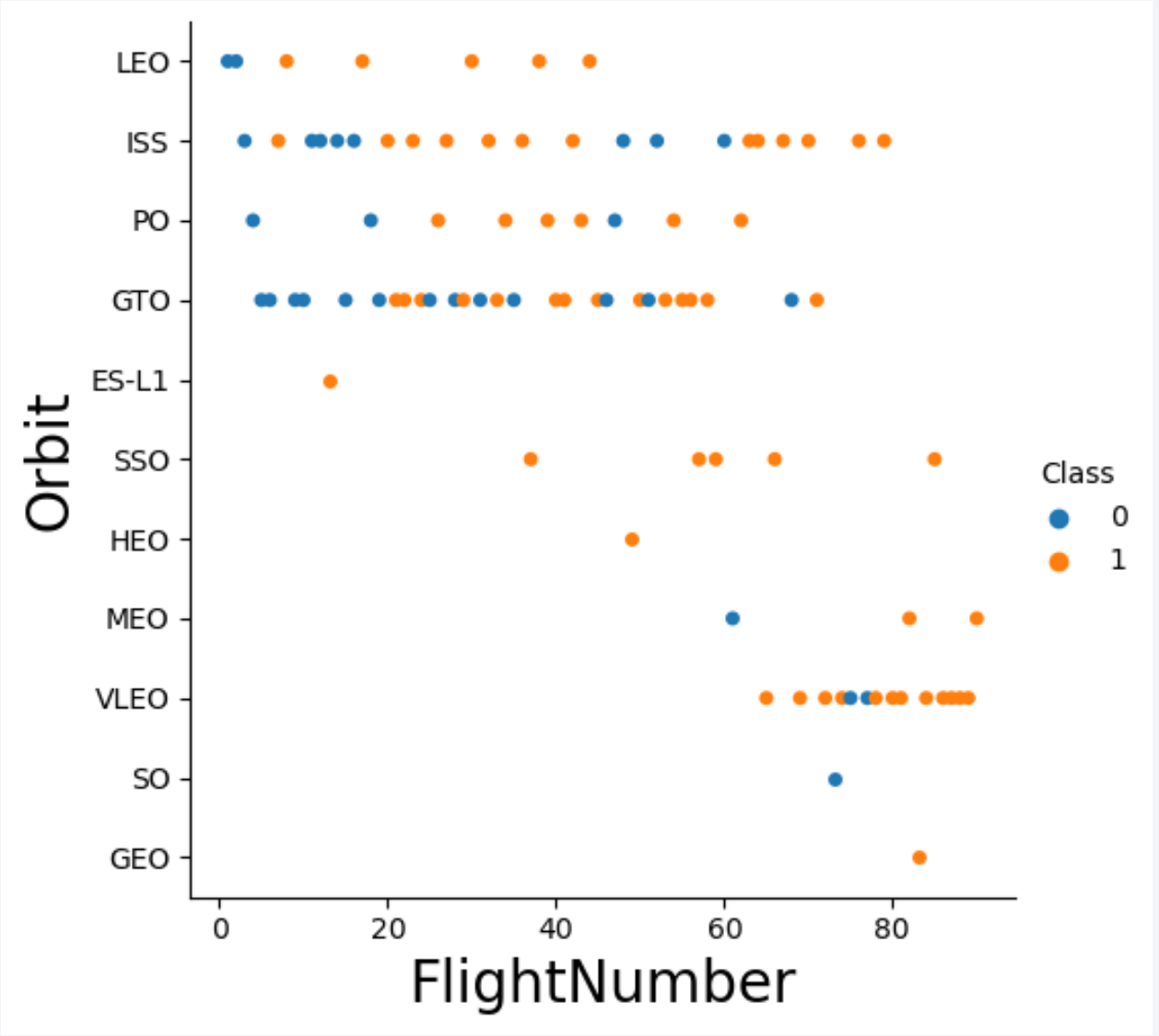
Success Rate vs. Orbit Type

- There is a very high chance of success for geosynchronous orbits
- There was only one attempt at an SO orbit, which failed
- Launch missions labelled SSO were all successful, and there were 5 such attempts
- Thus, with SO and SSO combined, the success rate is 0.83 and the SO term is removed



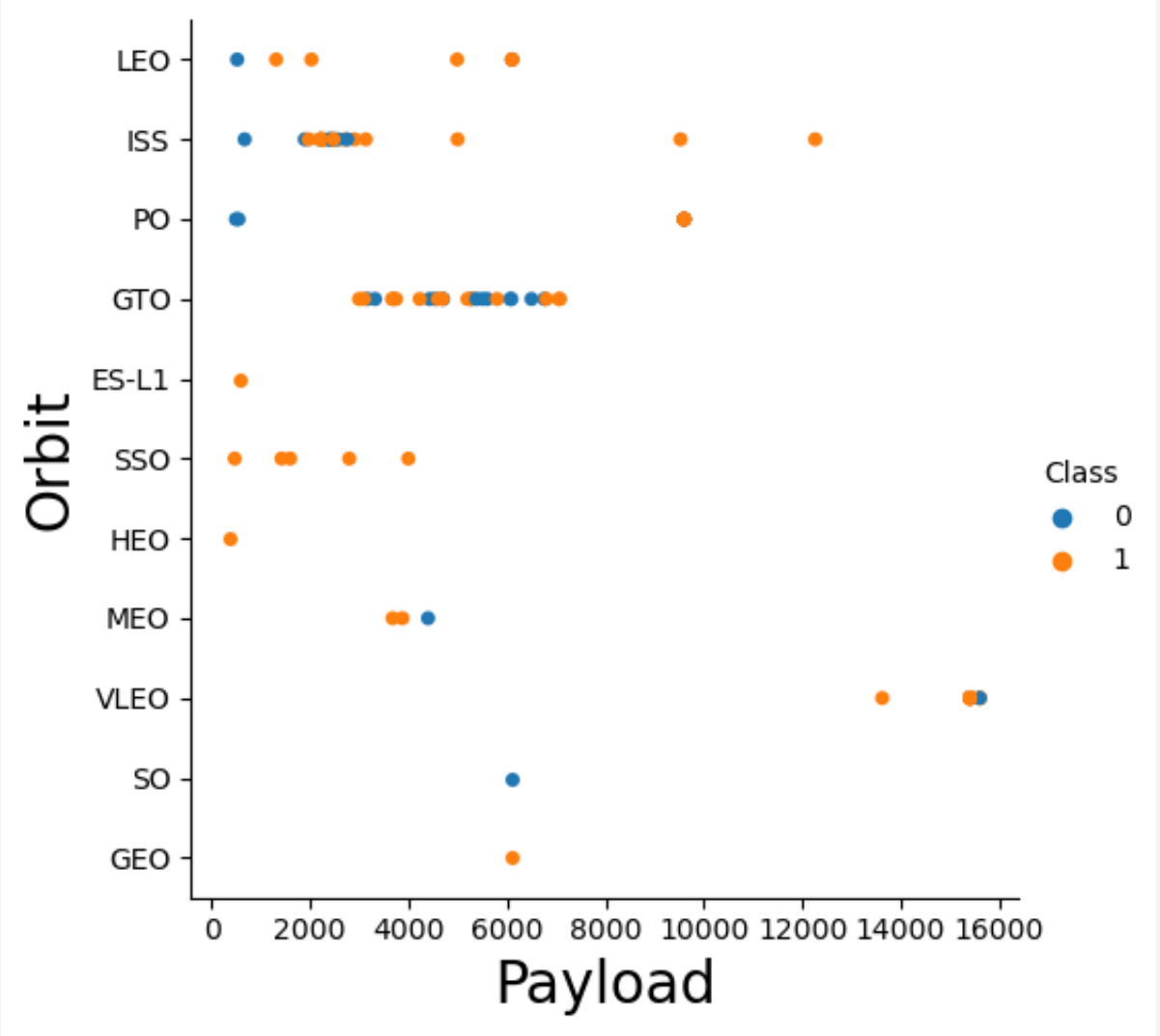
Flight Number vs. Orbit Type

- This graph shows the Launch attempts over time by flight number
- A number of launches were made for LEO, ISS, PO and GTO in the early phases
- SSO and SO are taken as the same, so there is one failure reported for SO and 5 successes for SSO
- The success rate is progressively improving as after flightnumber 75, there are no failures reported



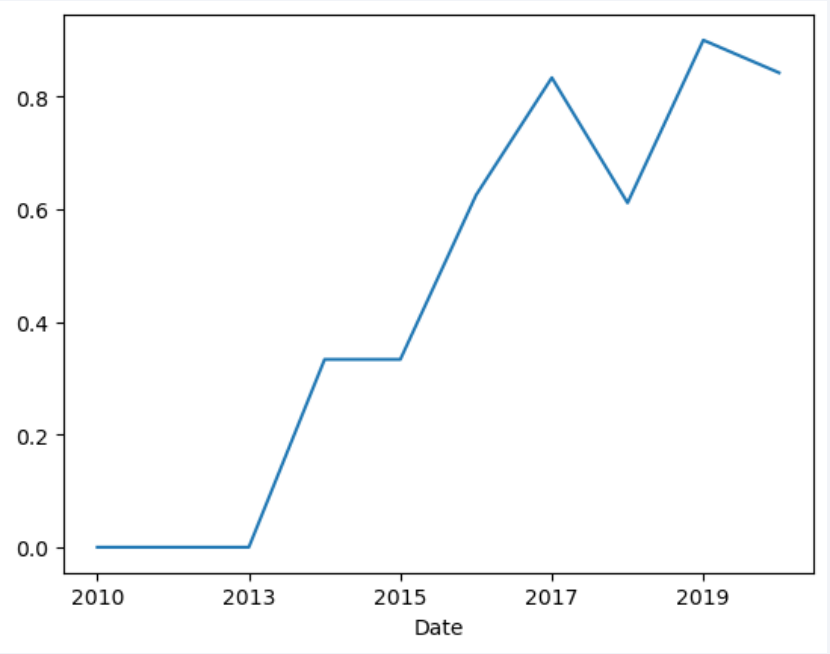
Payload vs. Orbit Type

- The graph shows payload vs orbit type
- Landing attempts are successful across different payload values



Launch Success Yearly Trend

- Show a line chart of yearly average success rate
- The Yearly Average success rate increased steadily from 2013 to 2017 as experience with launches increased
- Dips in success rate may be attributed to testing of newer next gen technology



All Launch Site Names

The Launch sites have the following names:

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

We can do this with SQL Queries.

```
[11]: %sql SELECT DISTINCT "Launch_site" FROM SPACEXTABLE
 * sqlite:///my_data1.db
Done.
```

[11]:

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

The dataframe is loaded into an SQLite database and queries are run. In most graphic visualizations above, CCAFS LC-40 and CCAFS SLC-40 have been combined, as their geographic location is within the same complex

Launch Site Names Begin with 'CCA'

There are two launch sites beginning with CCA. We can look at a sample of the data from these launch sites. We can see the customer being either SpaceX or NASA

Display 5 records where launch sites begin with the string 'CCA'

[12]:
```
%sql SELECT * FROM SPACEXTABLE WHERE "Launch_Site" LIKE 'CCA%' LIMIT 5
```

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

[12]:

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing_Outcome
2010-04-06	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
2010-08-12	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	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
2012-08-10	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
2013-01-03	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

Total Payload Mass

The total payload carried by boosters for NASA is 99980 kg

```
[13]: %sql SELECT SUM("PAYLOAD_MASS__KG_") AS 'Total Payload' FROM SPACEXTABLE WHERE "Customer" like 'NASA%'
 * sqlite:///my_data1.db
Done.
[13]:
```

Total Payload
99980

Average Payload Mass by F9 v1.1

The average payload mass carried by booster version F9 v1.1

Display average payload mass carried by booster version F9 v1.1

```
[14]: %sql SELECT AVG("PAYLOAD_MASS__KG_") AS 'Average Payload' FROM SPACEXTABLE WHERE "BOOSTER_VERSION" like 'F9 v1.1%'
 * sqlite:///my_data1.db
Done.
```

[14]:

Average Payload
2534.6666666666665

The average is 2534.66 kg

First Successful Ground Landing Date

The first successful Ground landing was done on 12-12-2015

```
[15]: %sql select MIN("date") from SPACEXTABLE WHERE "LANDING_OUTCOME" like 'Success%'
```

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

[15]:

MIN("date")
2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

This provides the names of boosters which have successfully landed on drone ship and had payload mass greater than 4000 but less than 6000

```
[16]: %sql select "booster_Version" from SPACEXTABLE WHERE "LANDING_OUTCOME" like 'Success%drone%' AND ("Payload_mass__kg_" > 4000 AND "Payload_mass__kg_" < 6000)
 * sqlite:///my_data1.db
Done.
```

[16]:

Booster_Version
F9 FT B1022
F9 FT B1026
F9 FT B1021.2
F9 FT B1031.2

The WHERE clause uses two Boolean parts, LANDING_OUTCOME starting with Success and a Boolean operation with Payload_mass_kg < 4000 and <6000. The BETWEEN _ AND _ clause can also be used

Total Number of Successful and Failure Mission Outcomes

The total number of Failure and Success of mission is given :-

Success = 98+1+1 = 100

Failure = 1

```
[17]: %sql select "Mission_Outcome", count(*) from SPACEXTABLE GROUP BY "Mission_Outcome"
```

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

[17]:

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

Boosters Carried Maximum Payload

These are the boosters which carry the maximum payload. All are variants of Falcon 9 heavy

```
[18]: %sql select "booster_version" from SPACEXTABLE where "payload_mass__kg_" = ( select max("payload_mass__kg_") from SPACEXTABLE)
 * sqlite:///my_data1.db
Done.
```

[18]:

Booster_Version
F9 B5 B1048.4
F9 B5 B1049.4
F9 B5 B1051.3
F9 B5 B1056.4
F9 B5 B1048.5
F9 B5 B1051.4
F9 B5 B1049.5
F9 B5 B1060.2
F9 B5 B1058.3
F9 B5 B1051.6
F9 B5 B1060.3
F9 B5 B1049.7

2015 Launch Records

We can list the failed landing outcomes, their booster versions and launch site names for a specific year, like 2015

Limitations of SQLite force us to use some string-based techniques to get the month name. A 3-letter month is obtained

[16]:
```
%sql select substr("JanFebMarAprMayJunJulAugSepOctNovDec", strftime("%m", "Date")*3 - 2, 3) as 'Month',\
"landing_outcome","Booster_version","Launch_Site" from SPACEXTABLE \
where substr(Date,0,5)='2015' and "Landing_Outcome" like 'Failure%drone%'
```

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

[16]:

Month	Landing_Outcome	Booster_Version	Launch_Site
Oct	Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40
Apr	Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40

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

We can check the ranking of Landing Outcomes between 2010-06-04 and 2017-03-20, in descending order

We can see that no landing was attempted for 10 launching missions. 6 Failures have been noted, and 16 successful attempts

```
[17]: %sql select "Landing_outcome",count("Landing_outcome") from SPACEXTABLE WHERE date < '2017-03-20' and date > '2010-06-04' group by "Landing_outcome"\
order by count("Landing_outcome") desc
```

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

[17]:

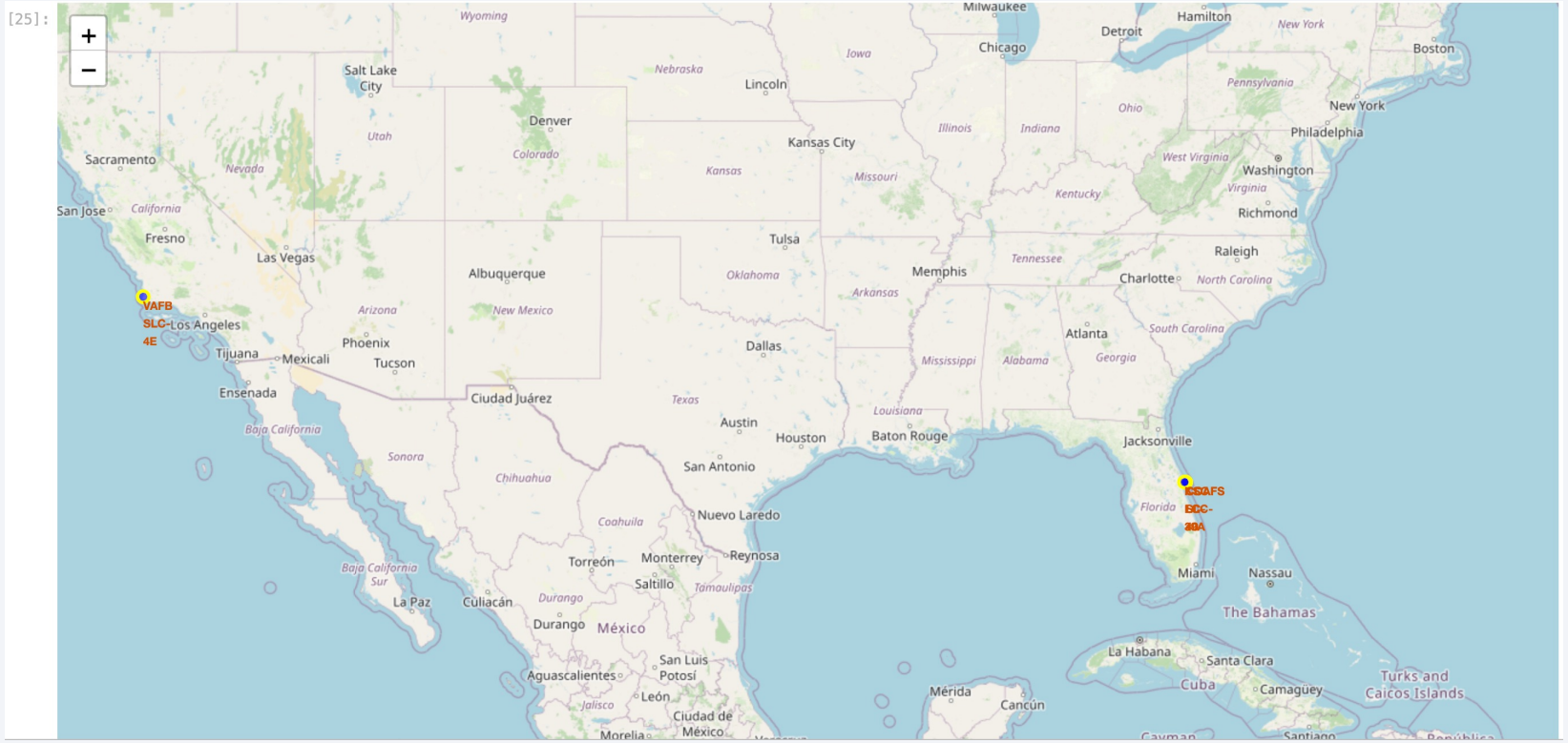
Landing_Outcome	count("Landing_outcome")
No attempt	10
Success (ground pad)	5
Success (drone ship)	5
Failure (drone ship)	5
Controlled (ocean)	3
Uncontrolled (ocean)	2
Precluded (drone ship)	1
Failure (parachute)	1

Section 3

Launch Sites Proximities Analysis

Launch Sites

- This map shows the position of the launch sites.
- One site is on the west coast, while other three are on the east coast in Florida
- The sites are close to the ocean

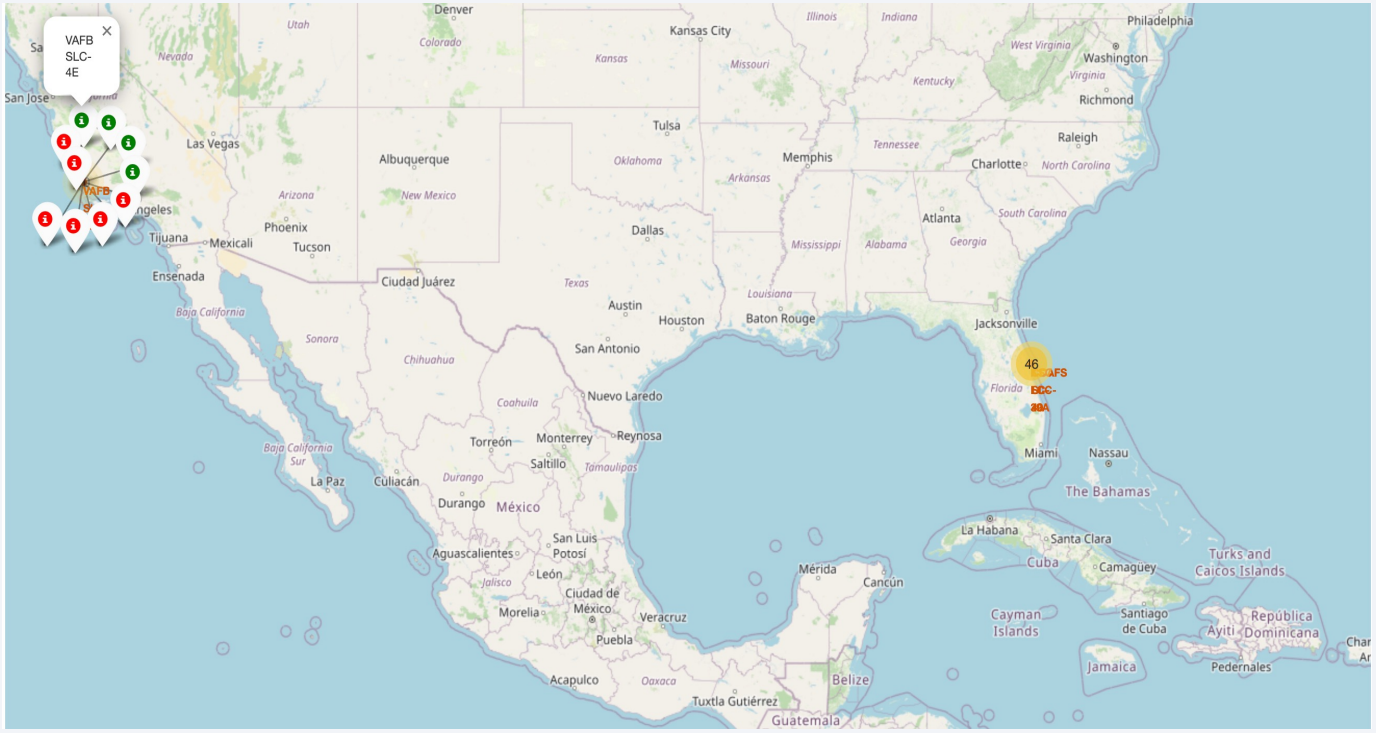


https://github.com/rsvinekar/Capstone-IBMDS/blob/main/folium_t1.html

Launch Outcome

The map interactively shows positive launch outcomes with green markers and failed missions with red markers

The total number of missions launched are clustered near the site



https://github.com/rsvinekar/Capstone-IBMDS/blob/main/folium_t3.html

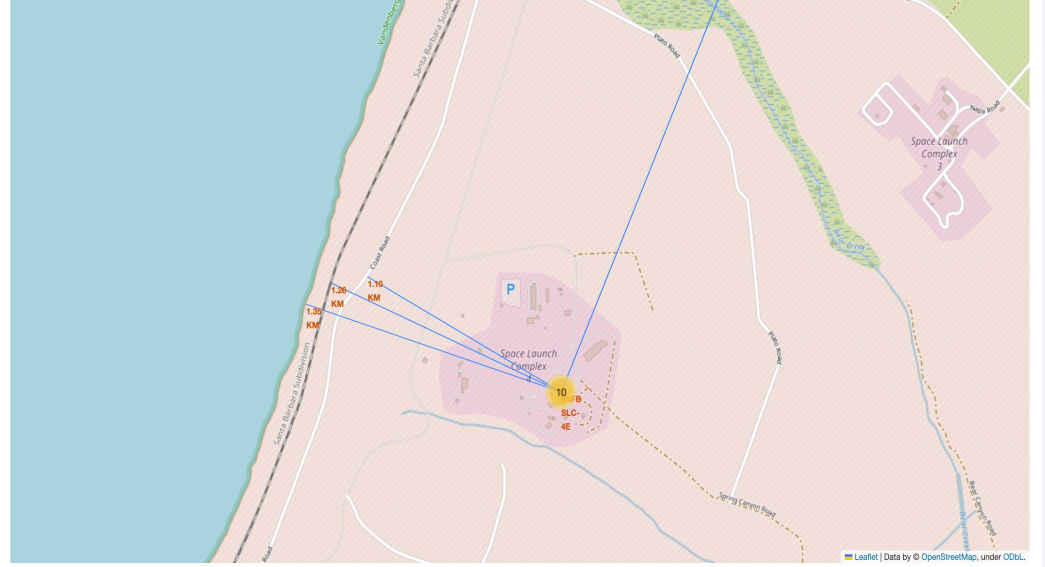
Launch Site proximity

The Launch sites are all close to the shoreline, and there are railway and road connections in its proximity

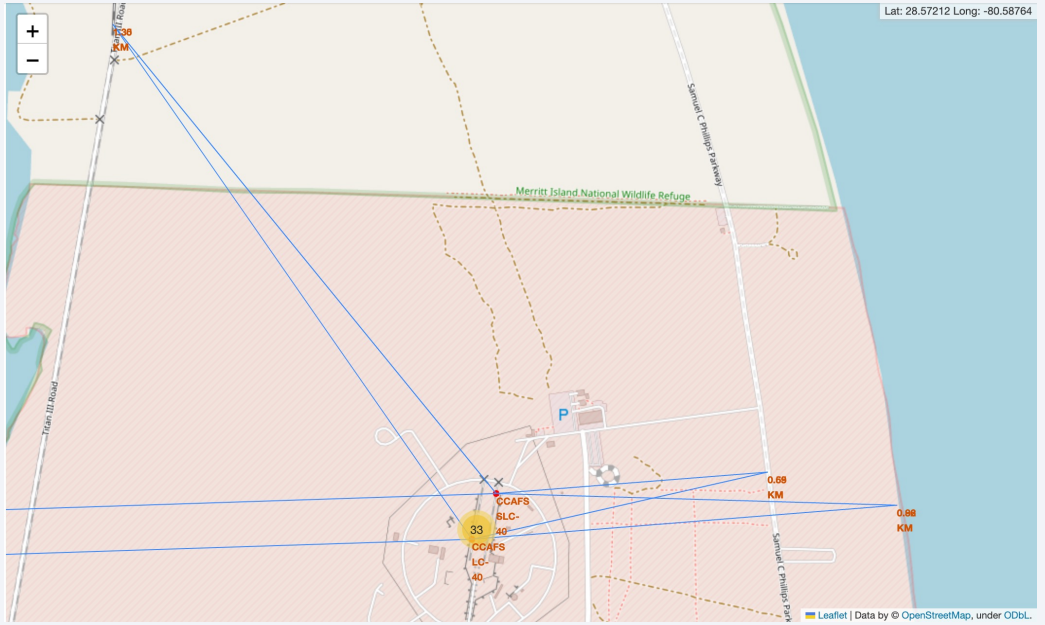
Large cities like Santa Maria on west coast (top) or Orlando in Florida (bottom) are much further away

These can be explained as follows:-

- Launches go over the Ocean. Any failure can avoid populated areas
- Rails and Roadways are close by, as logistics is required
- Populated cities are far off, to minimize any casualties in case of failures

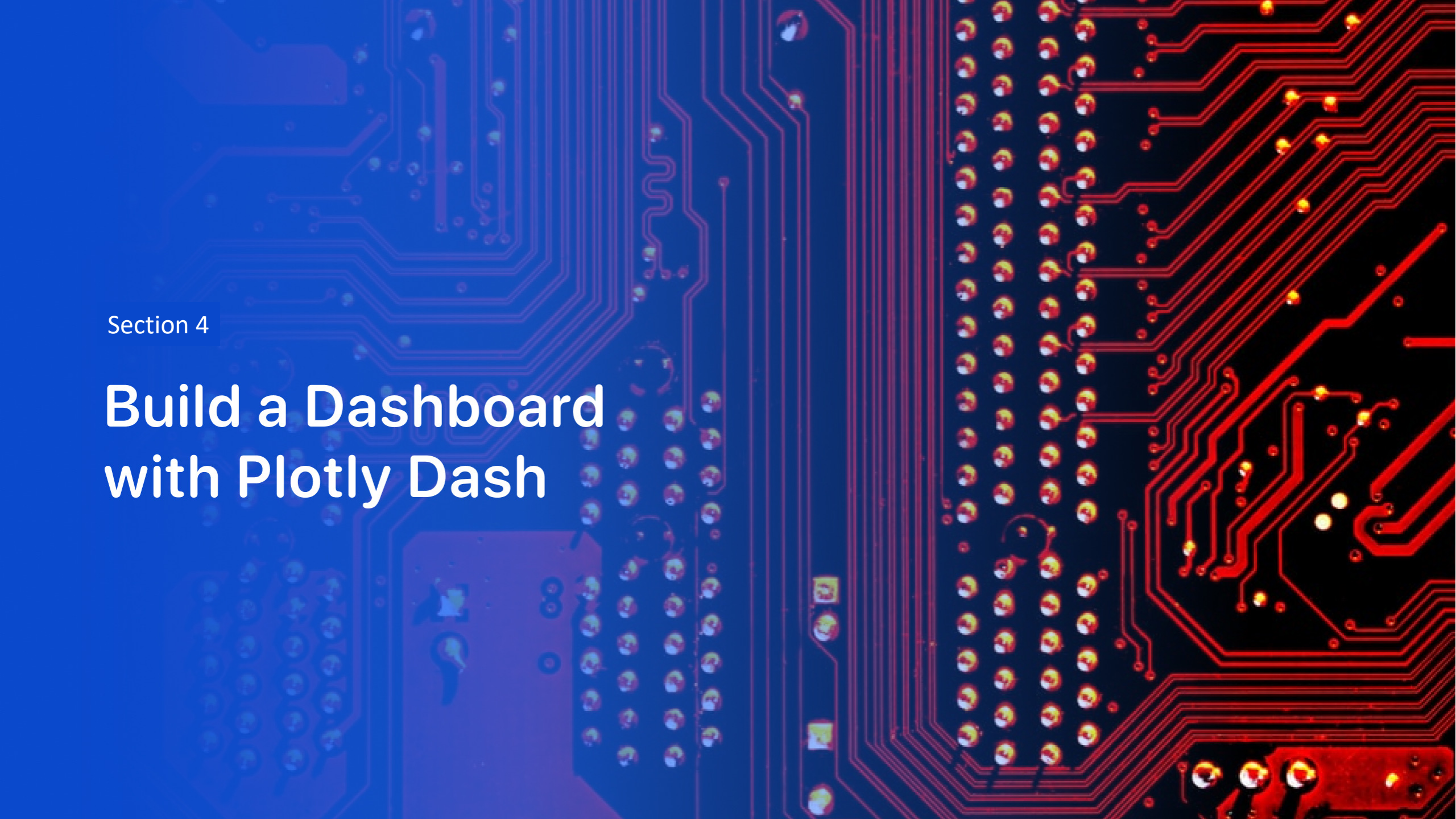


VAFB-SLC 4E is on the West Coast



CCAFS-LC 40 and CCAFS-SLC 40 are in close proximity on east coast. KSC LC-39A is to the west of these (not seen here)

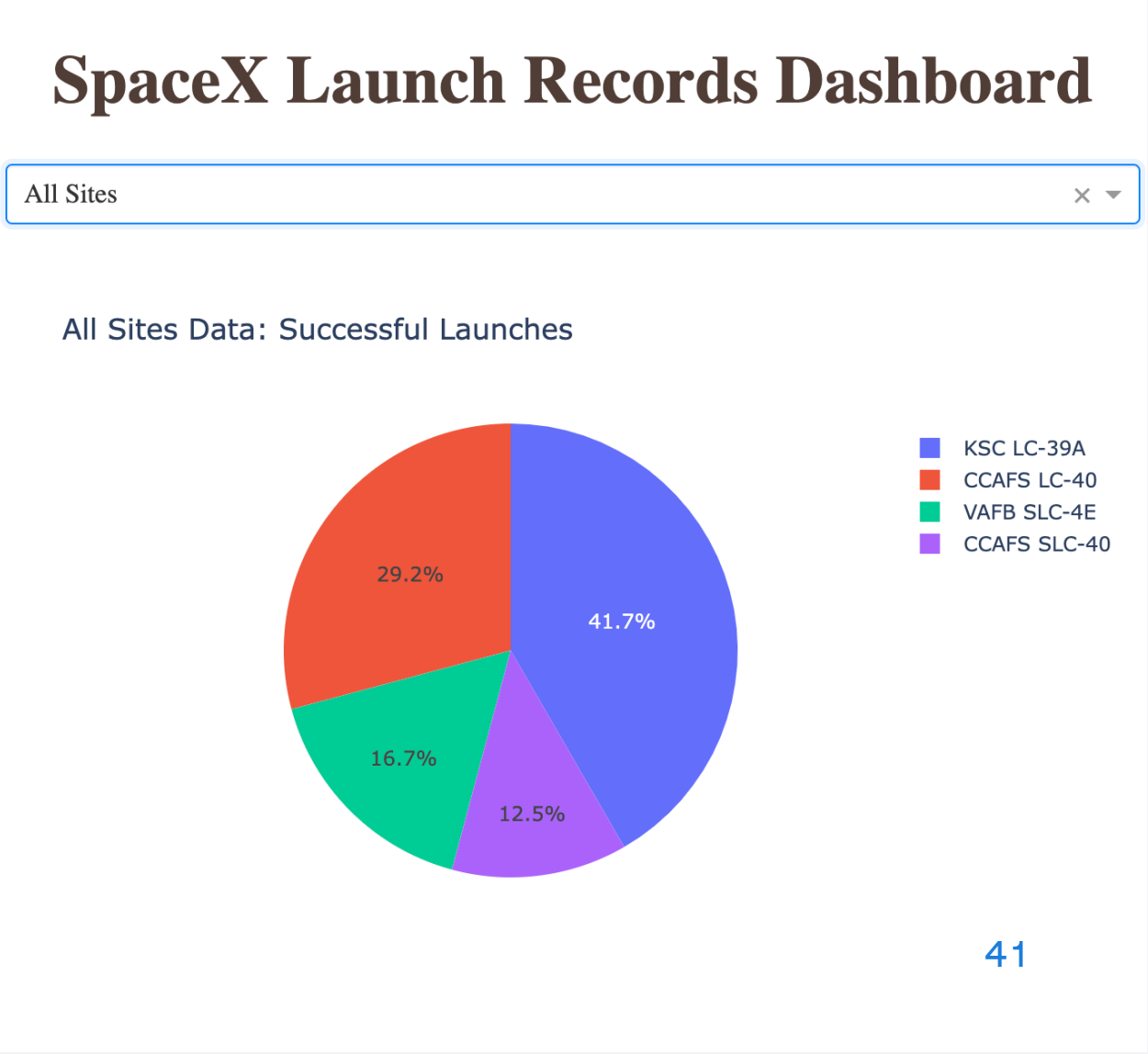
https://github.com/rsvinekar/Capstone-IBMDS/blob/main/folium_t4.html


Section 4
Build a Dashboard with Plotly Dash

Successful Launches per site

The dashboard provides a dropdown which can choose “All Sites”

The Piechart shows KSC LC-39A having the highest success rate of 41.7%



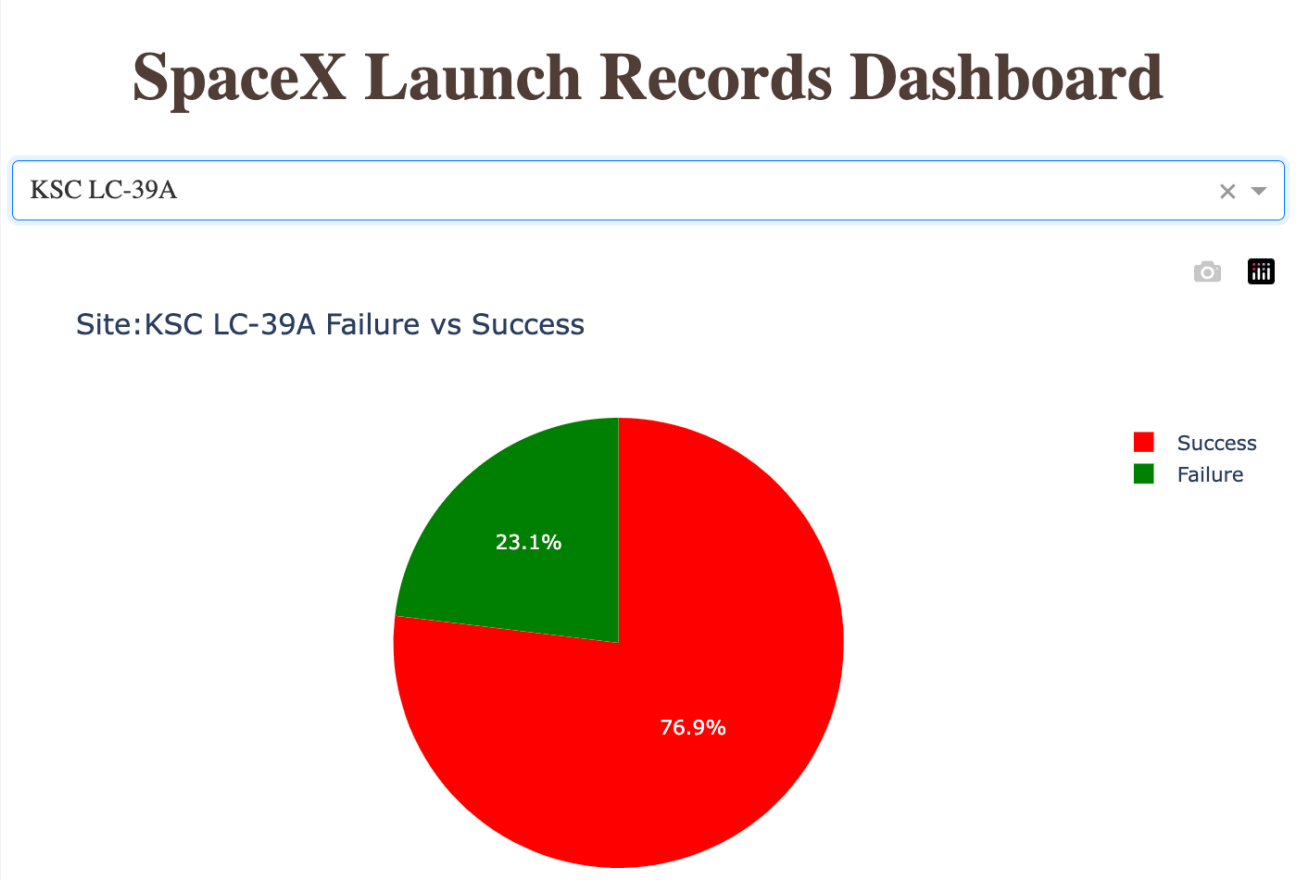
Success rate of the most productive launch site

The KSC LC-39A site shows the highest success rate compared to other sites

However, its actual success vs failure percentage is low : 23.1%

The CCAFS SLC-40 has a higher success rate of 42.9%

This is because KSC LC-39A has more launch attempts than CCAFS SLC-40



Dashboard Scatter plot

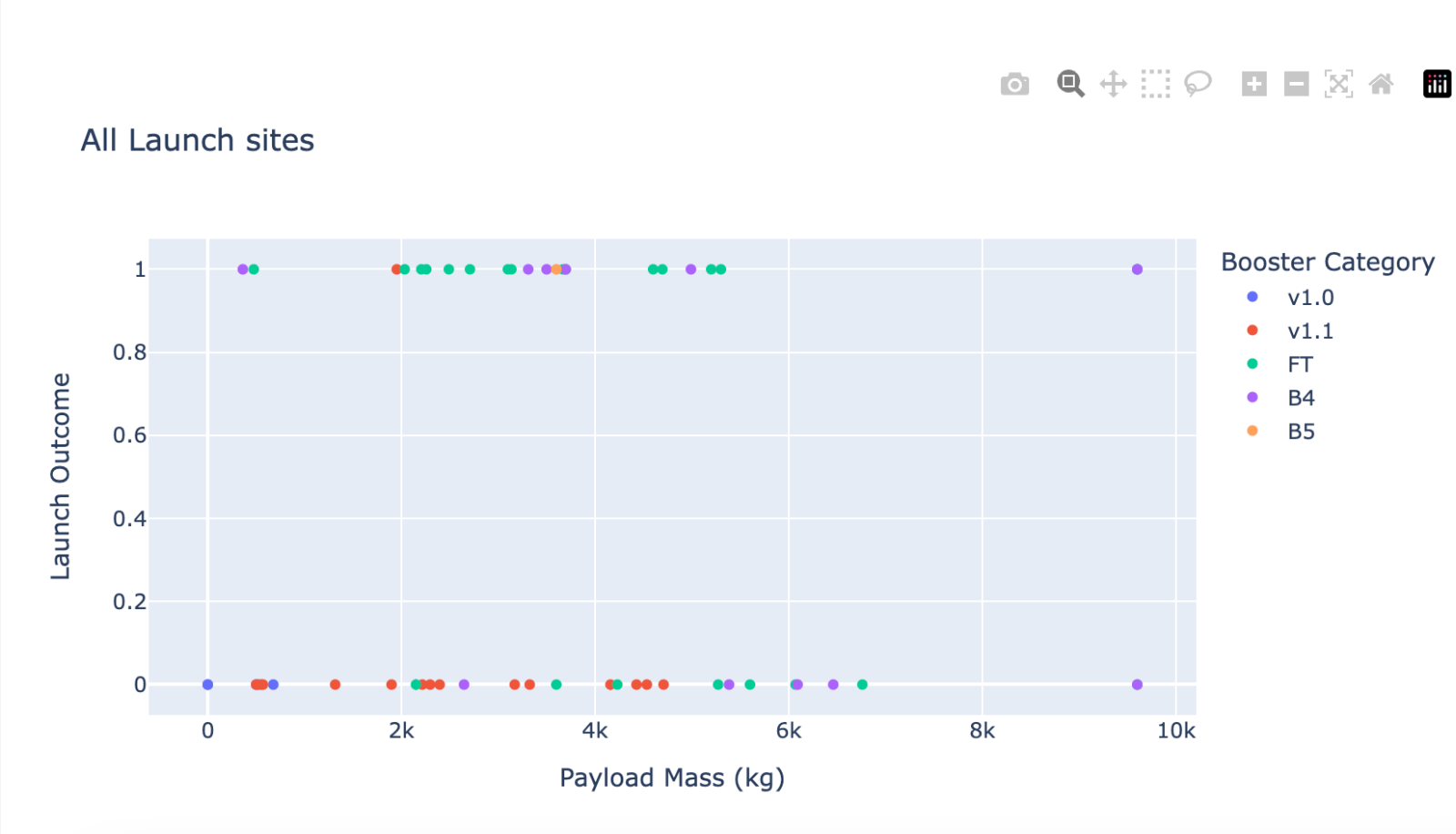
The Launch outcome is a binary quantity, 1 for success and 0 for failure

In the dashboard, we can get a scatter plot of Launch Outcome vs payload mass, colored by booster category

The plot shows that the FT Booster category has the higher success rate for payloads less than 6000 kg

The B4 Booster category works better at higher payloads

The booster payloads from 2000 kg and 6000 kg is most successful



Section 5

Predictive Analysis (Classification)

Classification Accuracy

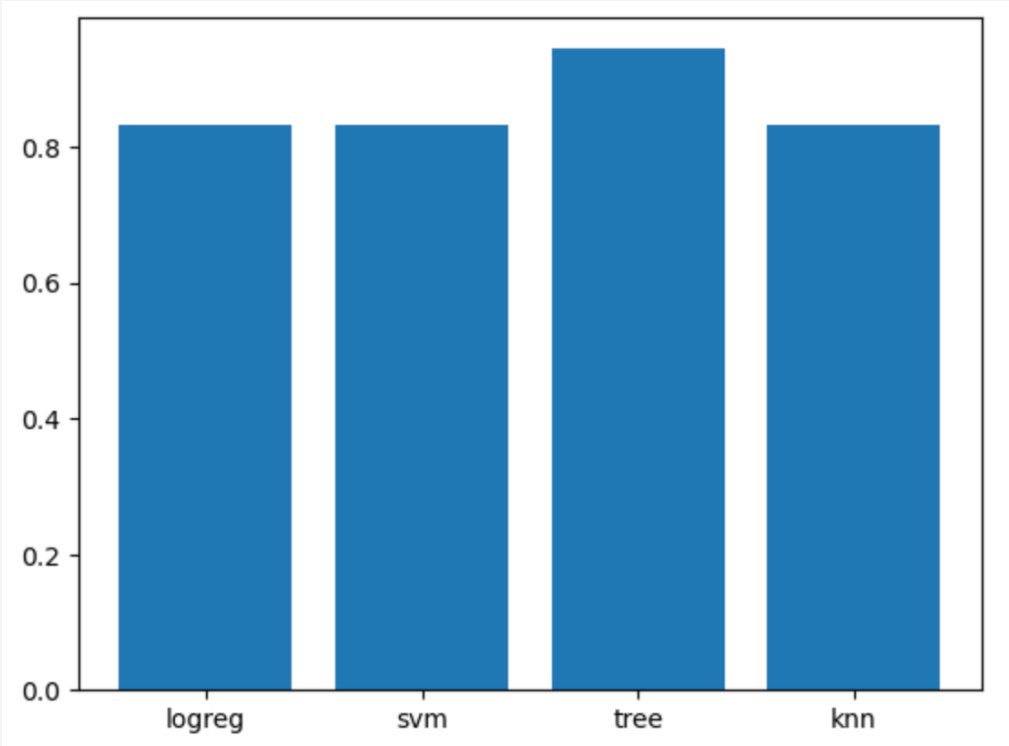
A model has been built which can predict the landing outcome

Four models were tried:

- logistic regression (logreg)
- support vector machine (svm)
- decision trees (tree)
- K-nearest neighbours (knn)

The optimal parameters were determined using Grid Search Cross Validation (GridSearchCV)

The results show that the decision trees show the greatest accuracy score



Confusion Matrix

The Decision tree model shows good predictions with the confusion matrix showing:-

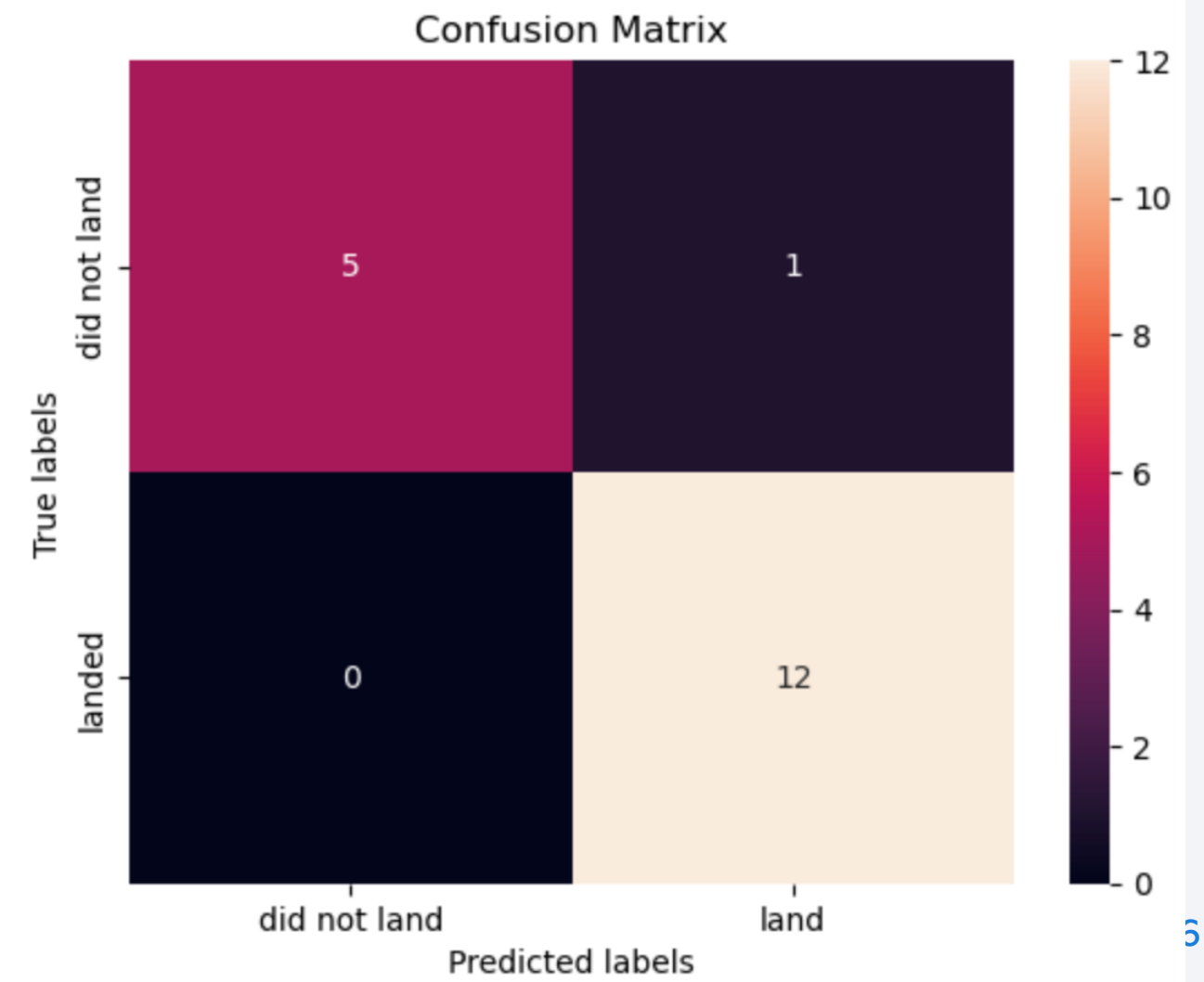
❌ **1 False positive**

✅ **5 True negatives**

✅ **12 True positives**

Thus, there is 1 undesirable result out of 18

This is a fairly good model, although it may do better with more data points



Conclusions

Data was obtained by REST API and web scraping on the launch status of SpaceX launches and specifications of the launchers

Exploratory data analysis gives us insights into the data, and maps can show the position of the launch sites and their proximity to coast, logistics infrastructure and distance from populated cities

Four different models for machine learning were used to predict the class variable from the other variables available to us. The best parameters were chosen using GridSearch cross validation for each

The decision tree model gave the best accuracy score of 0.86 and the confusion map seems acceptable. This ML model may be used for predictions on new data

Appendix

The GitHub repository for this work is : https://github.com/rsvinekar/Capstone-IBMDS

Two versions of the plotly dash application are provided, both give the same output

- Dash1: https://github.com/rsvinekar/Capstone-IBMDS/blob/main/spacex_dash_app_dash1.py
- Dash2: https://github.com/rsvinekar/Capstone-IBMDS/blob/main/spacex_dash_app_dash2.py

First is for dash 1.x as used in the Skills Network online. Second is for Dash 2.x installed on the latest Anaconda desktop. The latter is provided by default on Anaconda. To run the app, make sure dash is installed and then run

```
python spacex_dash_app_dash2.py
```

The difference is mainly in the import statements and slight changes elsewhere.

Thank you!