IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

<Natalie>
<20230914>

Outline

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

Executive Summary

Space X advertises Falcon 9 rocket launches on its website with a cost of 62 million dollars; other providers cost upward of 165 million dollars each, much of the savings is because Space X can reuse the first stage. Therefore if we can determine if the first stage will land, we can determine the cost of a launch. This information can be used if an alternate company wants to bid against space X for a rocket launch. In this lab, you will create a machine learning pipeline to predict if the first stage will land given the data from the preceding labs.

Introduction

Perform exploratory Data Analysis and determine Training Labels
•create a column for the class
•Standardize the data
•Split into training data and test data
-Find best Hyperparameter for SVM, Classification Trees and Logistic Regression
•Find the method performs best using test data

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Describe how data was collected
- Perform data wrangling
  - Describe how data was processed
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - How to build, tune, evaluate classification models

Data Collection

•In this capstone, we will predict if the Falcon 9 first stage will land successfully. SpaceX advertises Falcon 9 rocket launches on its website with a cost of 62 million dollars; other providers cost upward of 165 million dollars each, much of the savings is because SpaceX can reuse the first stage. Therefore if we can determine if the first stage will land, we can determine the cost of a launch. This information can be used if an alternate company wants to bid against SpaceX for a rocket launch. In this lab, you will collect and make sure the data is in the correct format from an API. The following is an example of a successful and launch.

In this lab, we will make a get request to the SpaceX API. We will also do some basic data wrangling and formating.

•Request to the SpaceX API

•Clean the requested data

Data Collection – SpaceX API

- Request to the SpaceX API
- Clean the requested data
- https://github.com/icylizzie/MyProject/blob/main/jupyter-labs-spacex-data-collection-api.ipynb

Task 2: Filter the dataframe to only include `Falcon 9` launches

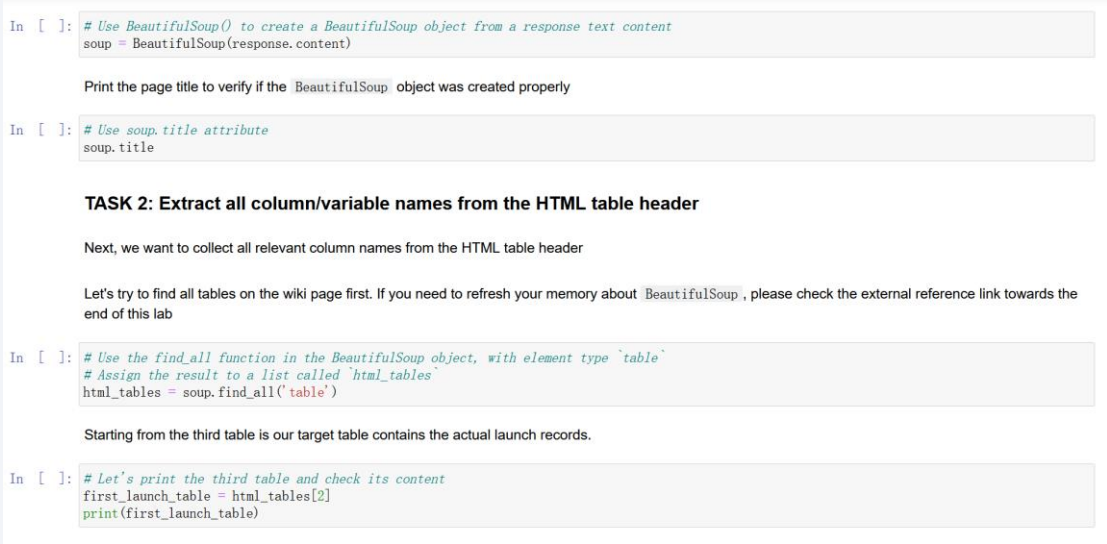
Finally we will remove the Falcon 1 launches keeping only the Falcon 9 launches. Filter the data dataframe using the `BoosterVersion` column to only keep the Falcon 9 launches. Save the filtered data to a new dataframe called `data_falcon9`.

```
# Hint data['BoosterVersion']!='Falcon 1'
data_falcon9=df_launch.loc[df_launch['BoosterVersion']!='Falcon 1']
data_falcon9.head()
```

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block	ReusedCount	Serial
4	6	2010-06-04	Falcon 9	NaN	LEO	None None	1	False	False	False	None	1.0	0	B0003
5	8	2012-05-22	Falcon 9	525.0	LEO	None None	1	False	False	False	None	1.0	0	B0005
6	10	2013-03-01	Falcon 9	677.0	ISS	None None	1	False	False	False	None	1.0	0	B0007
7	11	2013-09-29	Falcon 9	500.0	PO	False Ocean	1	False	False	False	None	1.0	0	B1003
8	12	2013-12-03	Falcon 9	3170.0	GTO	None None	1	False	False	False	None	1.0	0	B1004

Data Collection - Scraping

- Extract a Falcon 9 launch records HTML table from Wikipedia
- Parse the table and convert it into a Pandas data frame
- https://github.com/icylizzie/MyProject/blob/main/jupyter-labs-webscraping.ipynb



In [ ]:
```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(response.content)
```

Print the page title to verify if the `BeautifulSoup` object was created properly

In [ ]:
```
# Use soup.title attribute
soup.title
```

TASK 2: Extract all column/variable names from the HTML table header

Next, we want to collect all relevant column names from the HTML table header

Let's try to find all tables on the wiki page first. If you need to refresh your memory about `BeautifulSoup`, please check the external reference link towards the end of this lab

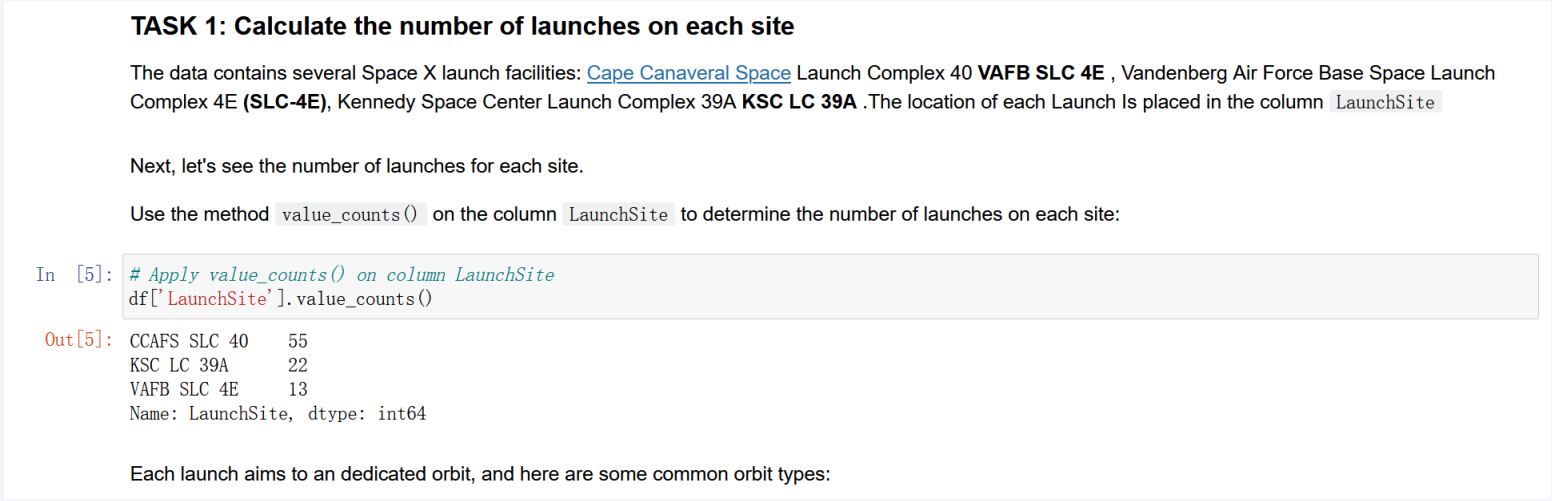
In [ ]:
```
# Use the find_all function in the BeautifulSoup object, with element type `table`
# Assign the result to a list called `html_tables`
html_tables = soup.find_all('table')
```

Starting from the third table is our target table contains the actual launch records.

In [ ]:
```
# Let's print the third table and check its content
first_launch_table = html_tables[2]
print(first_launch_table)
```

Data Wrangling

- Exploratory Data Analysis
- Determine Training Labels



TASK 1: Calculate the number of launches on each site

The data contains several Space X launch facilities: Cape Canaveral Space Launch Complex 40 **VAFB SLC 4E** , Vandenberg Air Force Base Space Launch Complex 4E **(SLC-4E)**, Kennedy Space Center Launch Complex 39A **KSC LC 39A** .The location of each Launch Is placed in the column `LaunchSite`

Next, let's see the number of launches for each site.

Use the method `value_counts()` on the column `LaunchSite` to determine the number of launches on each site:

In [5]:
```
# Apply value_counts() on column LaunchSite
df['LaunchSite'].value_counts()
```

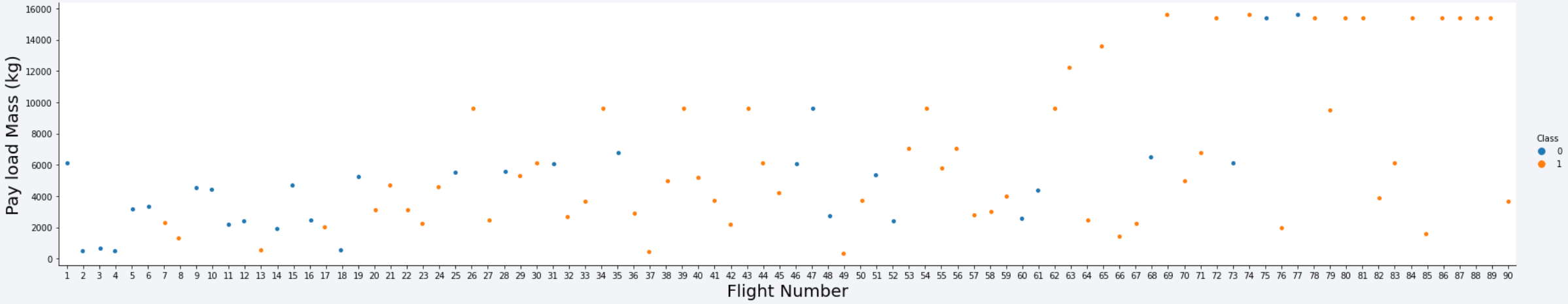
Out[5]:
```
CCAFS SLC 40    55
KSC LC 39A      22
VAFB SLC 4E     13
Name: LaunchSite, dtype: int64
```

Each launch aims to an dedicated orbit, and here are some common orbit types:

- https://github.com/icylizzie/MyProject/blob/main/labs-jupyter-spacex-Data%20wrangling.ipynb

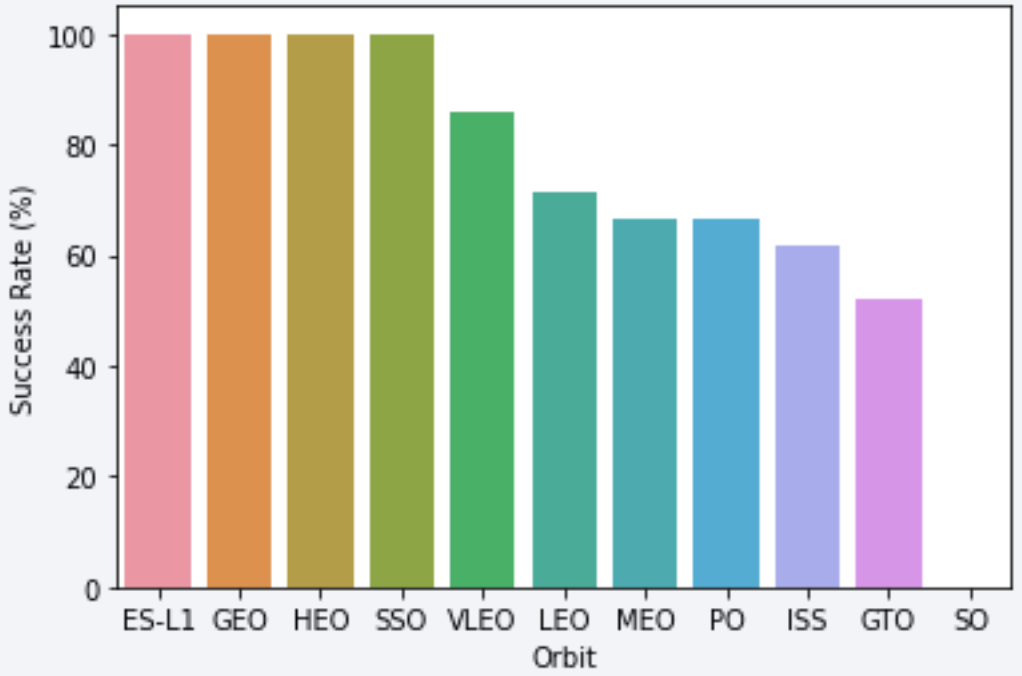
EDA with Data Visualization （1）

- This stage involved Data Analysis and Feature Engineering using Pandas and Matplotlib.
- Scatter plots were used to visualize the relationship between Flight number and Payload mass, Payload and Launch site among others.
- https://github.com/icylizzie/MyProject/blob/main/jupyter-labs-eda-dataviz.ipynb



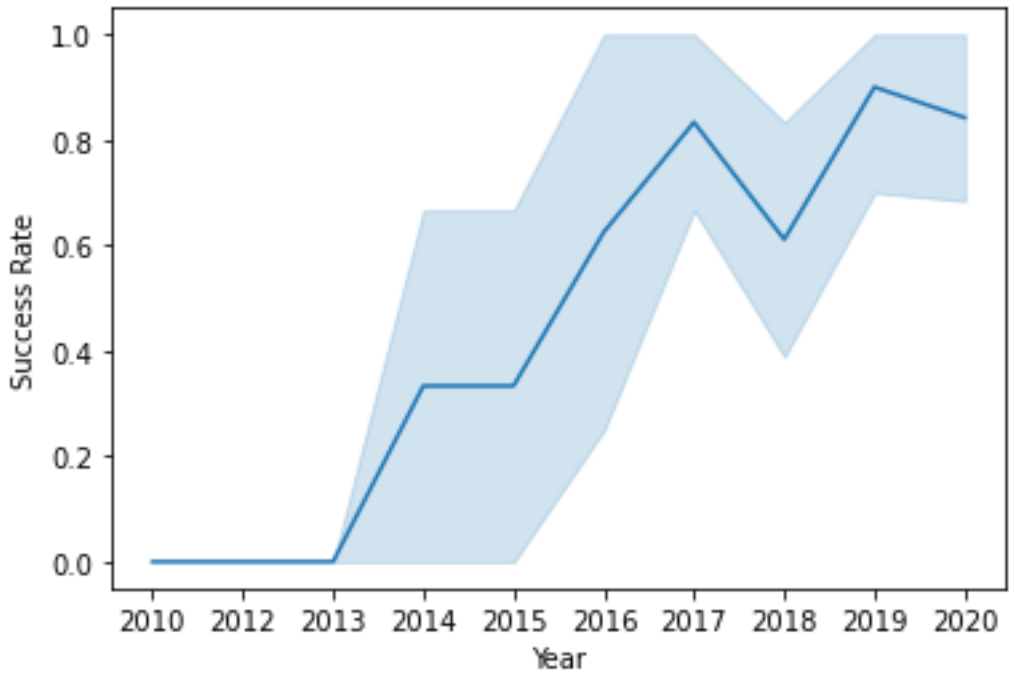
EDA with Data Visualization （2）

- A bar chart was used to visualize the relationship between success rate of each orbit type.
- https://github.com/icylizzie/MyProject/blob/main/jupyter-labs-eda-dataviz.ipynb



EDA with Data Visualization （3）

- A line chart was used to visualize the extracted year and the success rate.
- https://github.com/icylizzie/MyProject/blob/main/jupyter-labs-eda-dataviz.ipynb



EDA with SQL （1）

- https://github.com/icylizzie/MyProject/blob/main/jupyter-labs-eda-sql-coursera_sqllite.ipynb

1.Display the names of the unique launch sites in the space mission
select DISTINCT Launch_Site from SPACEXTBL

2.Display 5 records where launch sites begin with the string 'CCA’
SELECT * FROM SPACEXTABLE WHERE Launch_Site LIKE 'CCA%

3. Display the total payload mass carried by boosters launched by NASA (CRS)
SELECT SUM(PAYLOAD_MASS__KG_) FROM SPACEXTABLE WHERE Customer = 'NASA (CRS)’

4.Display average payload mass carried by booster version F9 v1.1
SELECT AVG(PAYLOAD_MASS__KG_) FROM SPACEXTABLE WHERE Booster_Version = 'F9 v1.1’

5.List the date when the first succesful landing outcome in ground pad was acheived.
SELECT MIN(Date) FROM SPACEXTABLE WHERE Landing_Outcome = 'Success (ground pad)'

EDA with SQL （2）

- https://github.com/icylizzie/MyProject/blob/main/jupyter-labs-eda-sql-coursera_sqllite.ipynb

6.List the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000

```
SELECT Booster_Version FROM SPACEXTABLE WHERE PAYLOAD_MASS__KG_ > 4000 AND PAYLOAD_MASS__KG_ < 6000 AND Landing_Outcome = 'Success (drone ship)’
```

7.List the total number of successful and failure mission outcomes

```
SELECT 'Successful',COUNT(*) FROM SPACEXTABLE WHERE Mission_Outcome LIKE 'Success%' UNION SELECT 'Unsuccessful',COUNT(*) FROM SPACEXTABLE WHERE Mission_Outcome LIKE 'Failure%’
```

8. List the names of the booster_versions which have carried the maximum payload mass. Use a subquery

```
SELECT DISTINCT Booster_Version FROM SPACEXTABLE WHERE PAYLOAD_MASS__KG_ IN (SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTABLE)
```

EDA with SQL （3）

- https://github.com/icylizzie/MyProject/blob/main/jupyter-labs-eda-sql-coursera_sqllite.ipynb

9. List the records which will display the month names, failure landing_outcomes in drone ship ,booster versions, launch_site for the months in year 2015.

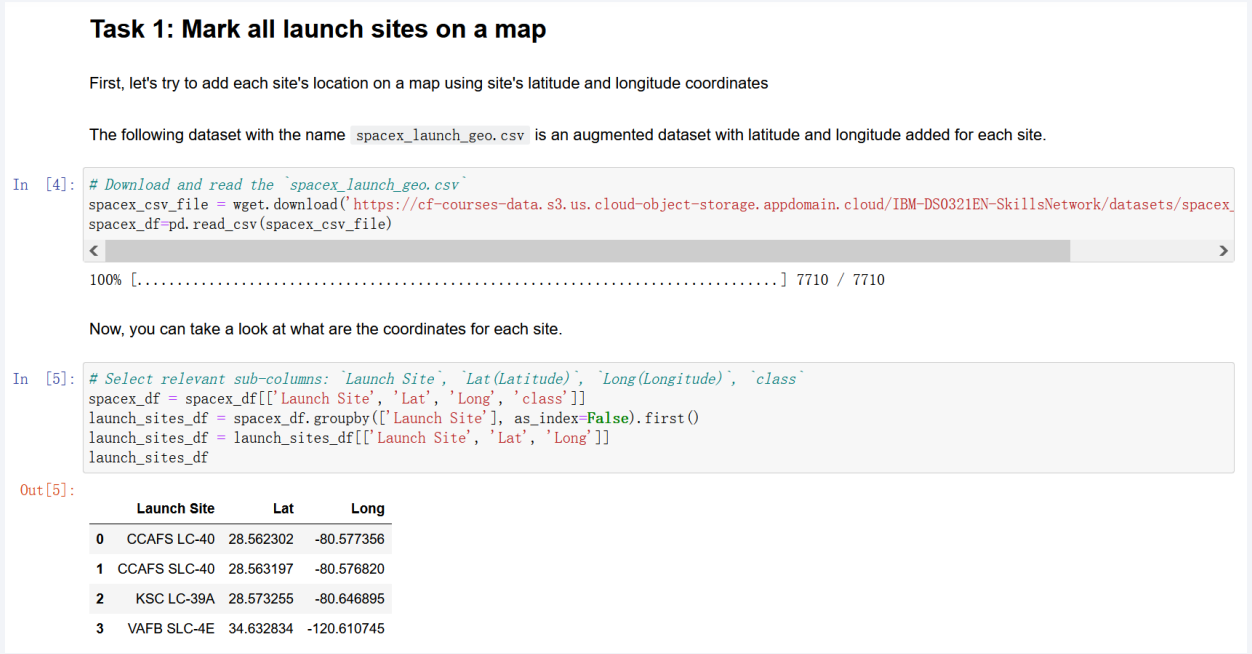
```
SELECT CASE WHEN substr(Date,6,2) = '01' THEN 'January' WHEN substr(Date,6,2) = '02' THEN 'February' WHEN substr(Date,6,2) = '03' THEN 'March' WHEN substr(Date,6,2) = '04' THEN 'April' WHEN substr(Date,6,2) = '05' THEN 'May' WHEN substr(Date,6,2) = '06' THEN 'June' WHEN substr(Date,6,2) = '07' THEN 'July' WHEN substr(Date,6,2) = '08' THEN 'August' WHEN substr(Date,6,2) = '09' THEN 'September' WHEN substr(Date,6,2) = '10' THEN 'October' WHEN substr(Date,6,2) = '11' THEN 'November'  WHEN substr(Date,6,2) = '12' THEN 'December' END as Month,Landing_Outcome,Booster_Version,Launch_Site FROM SPACEXTABLE WHERE Landing_Outcome = 'Failure (drone ship)' AND substr(Date,1,4)
```

10. Rank the count of landing outcomes (such as Failure (drone ship) or Success (ground pad)) between the date 2010-06-04 and 2017-03-20, in descending order.

```
SELECT Landing_Outcome, COUNT(*) FROM SPACEXTABLE WHERE Date > '2010-06-04' AND Date < '2017-03-20'GROUP BY Landing_Outcome ORDER BY COUNT(*) DESC
```

Build an Interactive Map with Folium

- Mark all launch sites on a map
- Mark the success/failed launches for each site on the map
- Calculate the distances between a launch site to its proximities
- https://github.com/icylizzie/MyProject/blob/main/lab_jupyter_launch_site_location.ipynb



Task 1: Mark all launch sites on a map

First, let's try to add each site's location on a map using site's latitude and longitude coordinates

The following dataset with the name `spacex_launch_geo.csv` is an augmented dataset with latitude and longitude added for each site.

In [4]:
```
# Download and read the `spacex_launch_geo.csv`
spacex_csv_file = wget.download('https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/spacex_
spacex_df=pd.read_csv(spacex_csv_file)
```

100% [..........] 7710 / 7710

Now, you can take a look at what are the coordinates for each site.

In [5]:
```
# Select relevant sub-columns: `Launch Site`, `Lat(Latitude)`, `Long(Longitude)`, `class`
spacex_df = spacex_df[['Launch Site', 'Lat', 'Long', 'class']]
launch_sites_df = spacex_df.groupby(['Launch Site'], as_index=False).first()
launch_sites_df = launch_sites_df[['Launch Site', 'Lat', 'Long']]
launch_sites_df
```

Out[5]:

	Launch Site	Lat	Long
0	CCAFS LC-40	28.562302	-80.577356
1	CCAFS SLC-40	28.563197	-80.576820
2	KSC LC-39A	28.573255	-80.646895
3	VAFB SLC-4E	34.632834	-120.610745

Build a Dashboard with Plotly Dash

- 1. Added dropdown control for Launch site
- 2. Added a callback function to render a success-pie-chart based on the selected launch
- site from the dropdown control
- 3. Added a range slider for Payload selection
- 4. Added a callback function to render the success-payload-scatter-chart plot

- https://github.com/icylizzie/MyProject/blob/main/spacex_dash_app.py

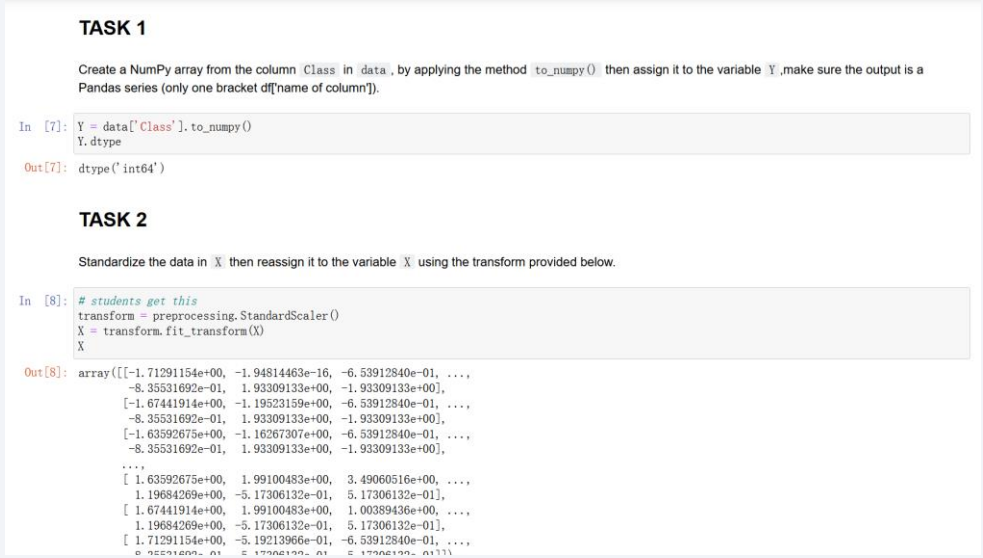
Predictive Analysis (Classification)

Perform exploratory Data Analysis and determine Training Labels

- create a column for the class
- Standardize the data
- Split into training data and test data

Find best Hyperparameter for SVM, Classification Trees and Logistic Regression

- Find the method performs best using test data



TASK 1

Create a NumPy array from the column `Class` in `data`, by applying the method `to_numpy()` then assign it to the variable `Y`,make sure the output is a Pandas series (only one bracket df['name of column']).

```
In [7]: Y = data['Class'].to_numpy()
        Y.dtype
Out[7]: dtype('int64')
```

TASK 2

Standardize the data in `X` then reassign it to the variable `X` using the transform provided below.

```
In [8]: # students get this
        transform = preprocessing.StandardScaler()
        X = transform.fit_transform(X)
        X
Out[8]: array([[-1.71291154e+00, -1.94814463e-16, -6.53912840e-01, ...,
                -8.35531692e-01,  1.93309133e+00, -1.93309133e+00],
               [-1.67441914e+00, -1.19523159e+00, -6.53912840e-01, ...,
                -8.35531692e-01,  1.93309133e+00, -1.93309133e+00],
               [-1.63592675e+00, -1.16267307e+00, -6.53912840e-01, ...,
                -8.35531692e-01,  1.93309133e+00, -1.93309133e+00],
               ...,
               [ 1.63592675e+00,  1.99100483e+00,  3.49060516e+00, ...,
                 1.19684269e+00, -5.17306132e-01,  5.17306132e-01],
               [ 1.67441914e+00,  1.99100483e+00,  1.00389436e+00, ...,
                 1.19684269e+00, -5.17306132e-01,  5.17306132e-01],
               [ 1.71291154e+00, -5.19213966e-01, -6.53912840e-01, ...,
```

- https://github.com/icylizzie/MyProject/blob/main/SpaceX_Machine_Learning_Prediction_Part_5.jupyterlite.ipynb

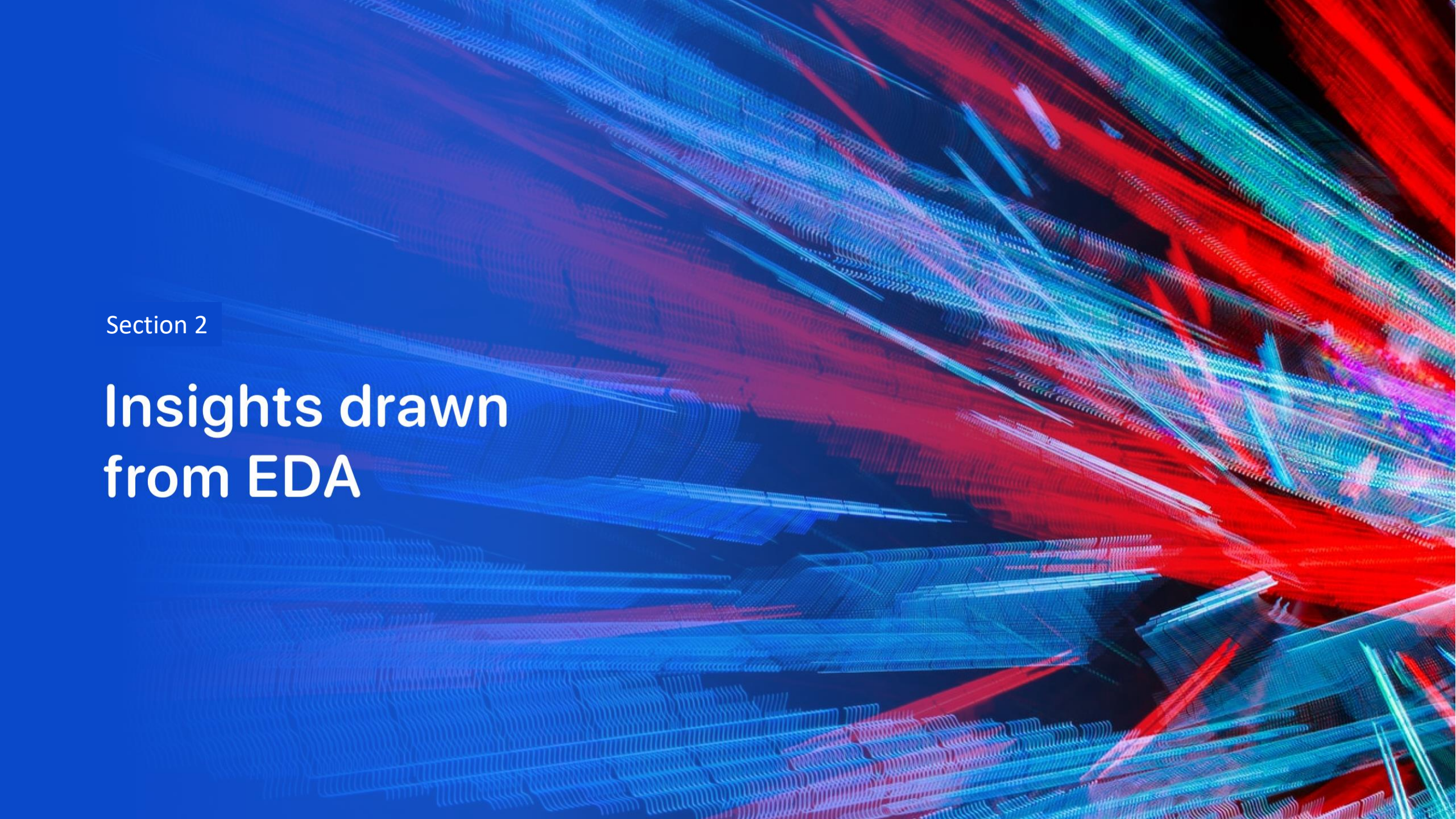
Results

- Predictive analysis results

Find the method performs best:

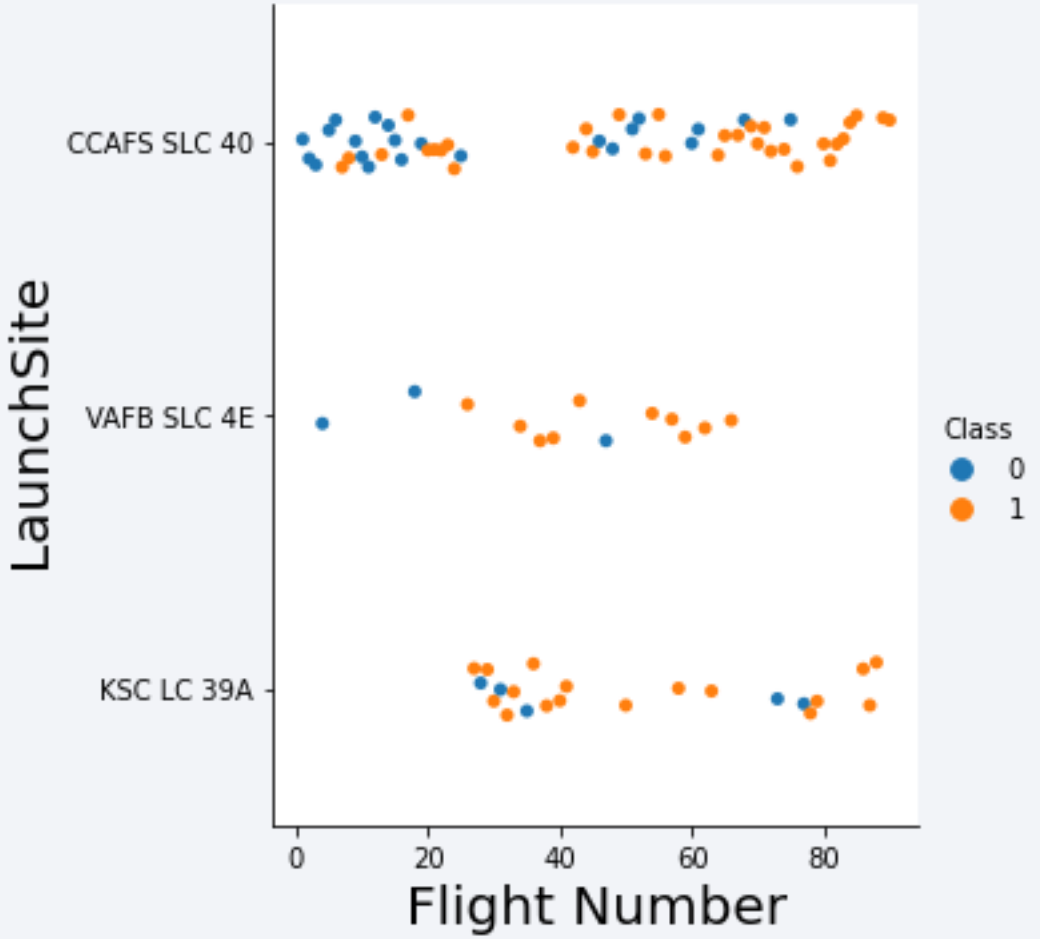
```
Report = pd.DataFrame({'Method' : ['Test Data Accuracy']})
knn_accuracy=knn_cv.score(X_test, Y_test)
Decision_tree_accuracy=tree_cv.score(X_test, Y_test)
SVM_accuracy=svm_cv.score(X_test, Y_test)
Logistic_Regression=logreg_cv.score(X_test, Y_test)

Report['Logistic_Reg'] = [Logistic_Regression]
Report['SVM'] = [SVM_accuracy]
Report['Decision Tree'] = [Decision_tree_accuracy]
Report['KNN'] = [knn_accuracy]

Report.transpose()
```

	0
Method	Test Data Accuracy
Logistic_Reg	0.833333
SVM	0.833333
Decision Tree	0.666667
KNN	0.833333


Section 2
Insights drawn from EDA

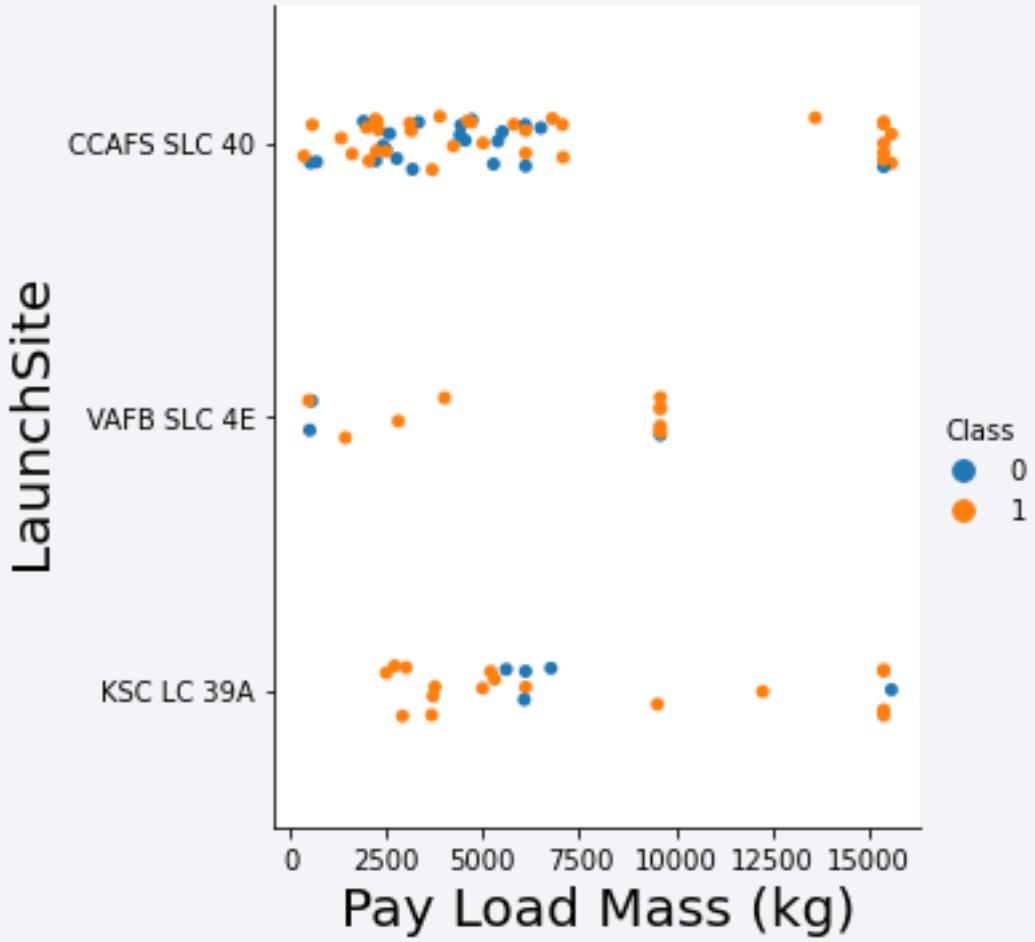
Flight Number vs. Launch Site

- It can be deduced that, as flight number increases, the success rate also increased.



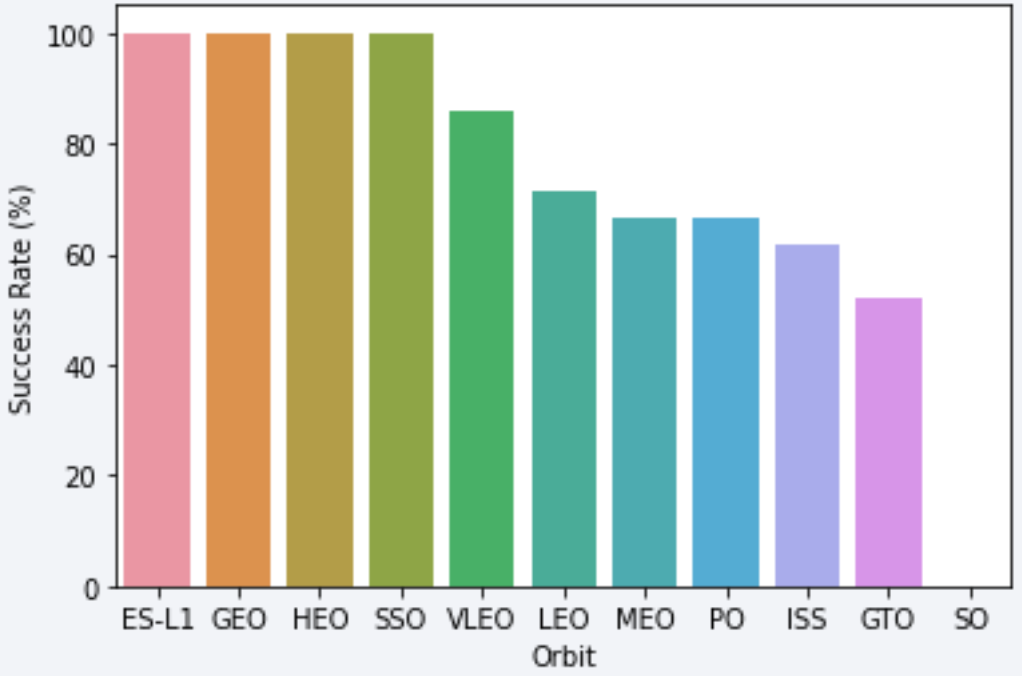
Payload vs. Launch Site

- If you observe the Payload Vs. Launch Site scatter point chart, you will find for the VAFB-SLC launch site there are not rockets launched for heavypayload mass(greater than 10000)



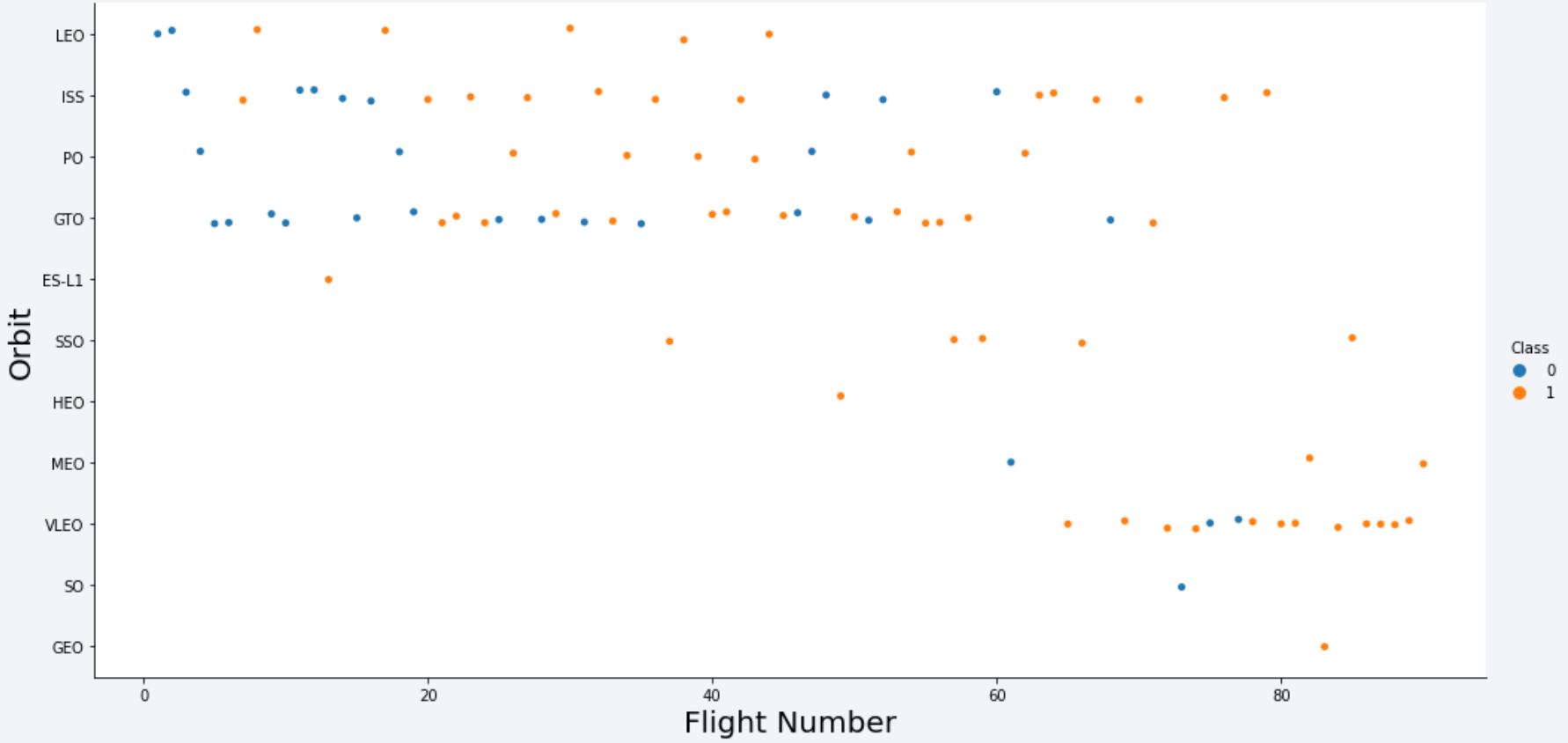
Success Rate vs. Orbit Type

- Orbits ES-L1, GEO, HEO and SSO have success rates of100%. Orbit SO has the lowest success rate.



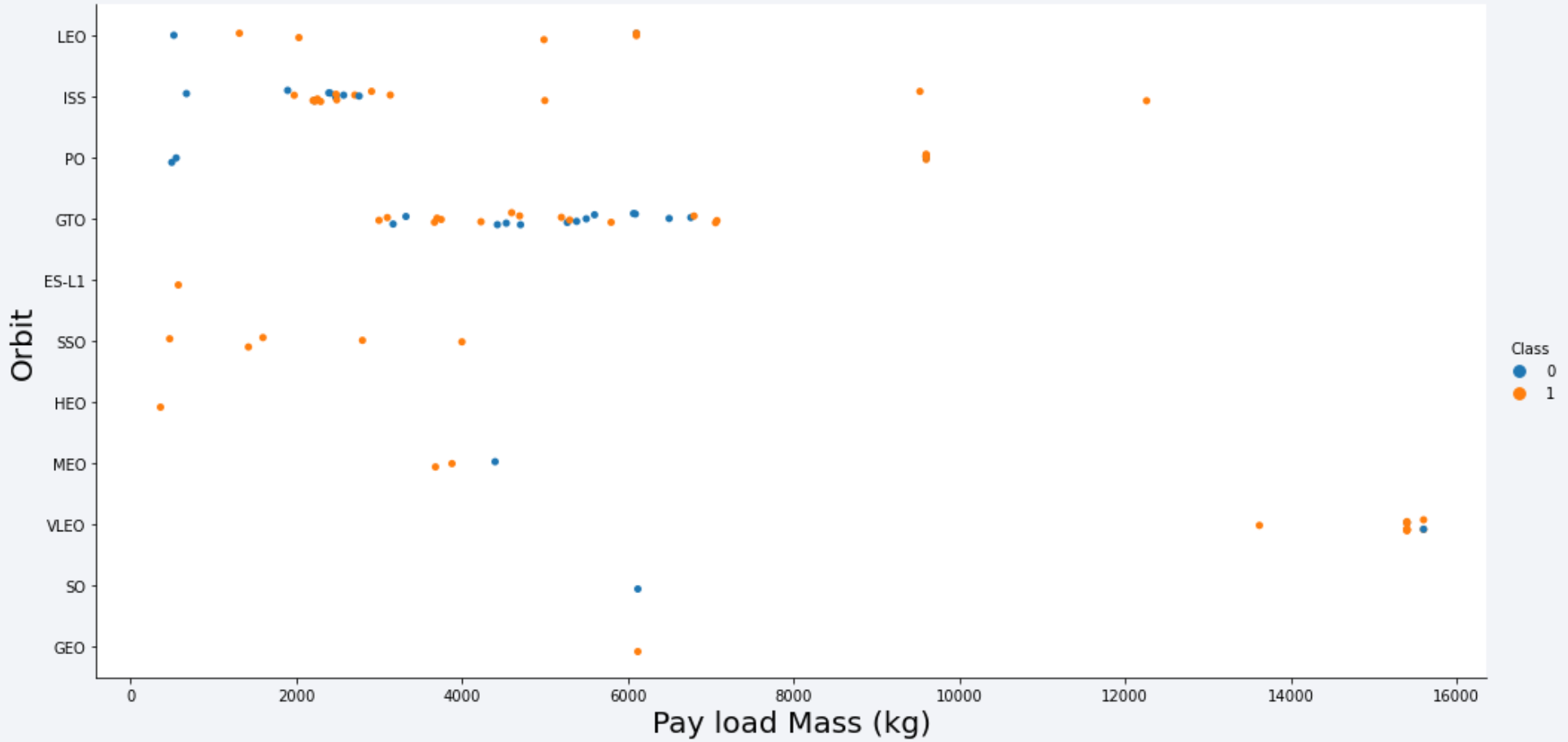
Flight Number vs. Orbit Type

- In the LEO orbit the success appears related to the number of flights; on the other hand, there seems to be no relationship between flight number when in GTO orbit



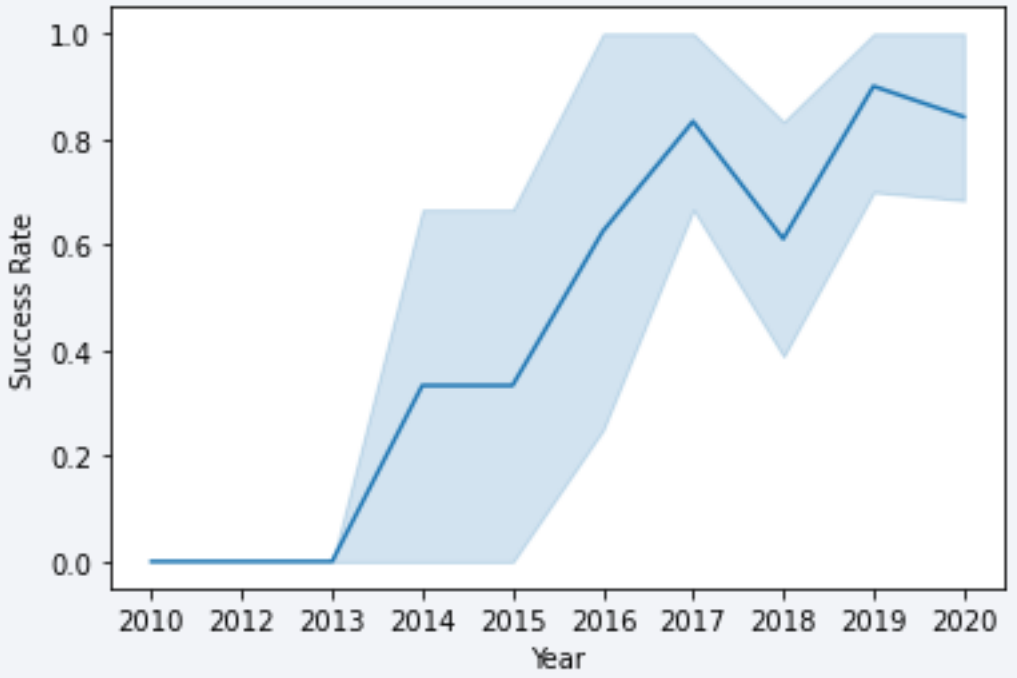
Payload vs. Orbit Type

- With heavy payloads the successful landing or positive landing rate are more for Polar,LEO and ISS.



Launch Success Yearly Trend

- Success rate has been rising since 2013



All Launch Site Names

- Find the names of the unique launch sites

Task 1

Display the names of the unique launch sites in the space mission

In [10]:
```
cur.execute("select DISTINCT Launch_Site from SPACEXTBL")
cur.fetchall()
```

Out[10]:
```
[('CCAFS LC-40',), ('VAFB SLC-4E',), ('KSC LC-39A',), ('CCAFS SLC-40',)]
```

Launch Site Names Begin with 'CCA'

- Find 5 records where launch sites begin with `CCA`

Task 2

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

In [3]:
```
cur.execute("SELECT * FROM SPACEXTABLE WHERE Launch_Site LIKE 'CCA%'Limit 5")
cur.fetchall()
```

Out[3]:
```
[('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

- Calculate the total payload carried by boosters from NAS

Task 3

Display the total payload mass carried by boosters launched by NASA (CRS)

In [17]:
```
cur.execute("SELECT SUM(PAYLOAD_MASS__KG_) FROM SPACEXTABLE WHERE Customer = 'NASA (CRS)'")
cur.fetchone()
```

Out[17]: (45596,)

Average Payload Mass by F9 v1.1

- Calculate the average payload mass carried by booster version F9 v1.1

Task 4

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

In [18]:
```
cur.execute("SELECT AVG(PAYLOAD_MASS__KG_) FROM SPACEXTABLE WHERE Booster_Version = 'F9 v1.1'")
cur.fetchone()
```

Out[18]: (2928.4,)

First Successful Ground Landing Date

- Find the dates of the first successful landing outcome on ground pad

Task 5

List the date when the first succesful landing outcome in ground pad was acheived.

Hint:Use min function

In [19]:
```
cur.execute("SELECT MIN(Date) FROM SPACEXTABLE WHERE Landing_Outcome = 'Success (ground pad)'")
cur.fetchone()
```

Out[19]:
```
('2015-12-22',)
```

Successful Drone Ship Landing with Payload between 4000 and 6000

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

Task 6

List the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000

In [21]:
```
cur.execute("SELECT Booster_Version FROM SPACEXTABLE WHERE PAYLOAD_MASS__KG_ > 4000 AND PAYLOAD_MASS__KG_ < 6000 AND Landing_Outcome = 'Success
cur.fetchall()
```

Out[21]:
```
[('F9 FT B1022',), ('F9 FT B1026',), ('F9 FT  B1021.2',), ('F9 FT  B1031.2',)]
```

Total Number of Successful and Failure Mission Outcomes

- Calculate the total number of successful and failure mission outcomes

Task 7

List the total number of successful and failure mission outcomes

In [23]:
```
cur.execute("SELECT 'Successful',COUNT(*) FROM SPACEXTABLE WHERE Mission_Outcome LIKE 'Success%' UNION SELECT 'Unsuccessful',COUNT(*) FROM SPAC
cur.fetchall()
```

Out[23]:
```
[('Successful', 100), ('Unsuccessful', 1)]
```

Boosters Carried Maximum Payload

- List the names of the booster which have carried the maximum payload mass

Task 8

List the names of the booster_versions which have carried the maximum payload mass. Use a subquery

In [26]:
```
cur.execute("SELECT DISTINCT Booster_Version FROM SPACEXTABLE WHERE PAYLOAD_MASS__KG_ IN (SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTABLE)")
cur.fetchall()
```

Out[26]:
```
[('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

- List the failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015

Task 9

List the records which will display the month names, failure landing_outcomes in drone ship ,booster versions, launch_site for the months in year 2015. ¶

Note: SQLLite does not support monthnames. So you need to use substr(Date, 4, 2) as month to get the months and substr(Date,7,4)='2015' for year.

In [29]:
```
cur.execute("SELECT CASE WHEN substr(Date,6,2) = '01' THEN 'January' WHEN substr(Date,6,2) = '02' THEN 'February' WHEN substr(Date,6,2) = '03'
cur.fetchall()
```

Out[29]:
```
[('October', 'Failure (drone ship)', 'F9 v1.1 B1012', 'CCAFS LC-40'),
 ('April', 'Failure (drone ship)', 'F9 v1.1 B1015', 'CCAFS LC-40'),
 ('January', 'Failure (drone ship)', 'F9 v1.1 B1017', 'VAFB SLC-4E'),
 ('April', 'Failure (drone ship)', 'F9 FT B1020', 'CCAFS LC-40'),
 ('June', 'Failure (drone ship)', 'F9 FT B1024', 'CCAFS LC-40')]
```

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

- Rank the count of landing outcomes (such as Failure (drone ship) or Success (ground pad)) between the date 2010-06-04 and 2017-03-20, in descending order

Task 10

Rank the count of landing outcomes (such as Failure (drone ship) or Success (ground pad)) between the date 2010-06-04 and 2017-03-20, in descending order.

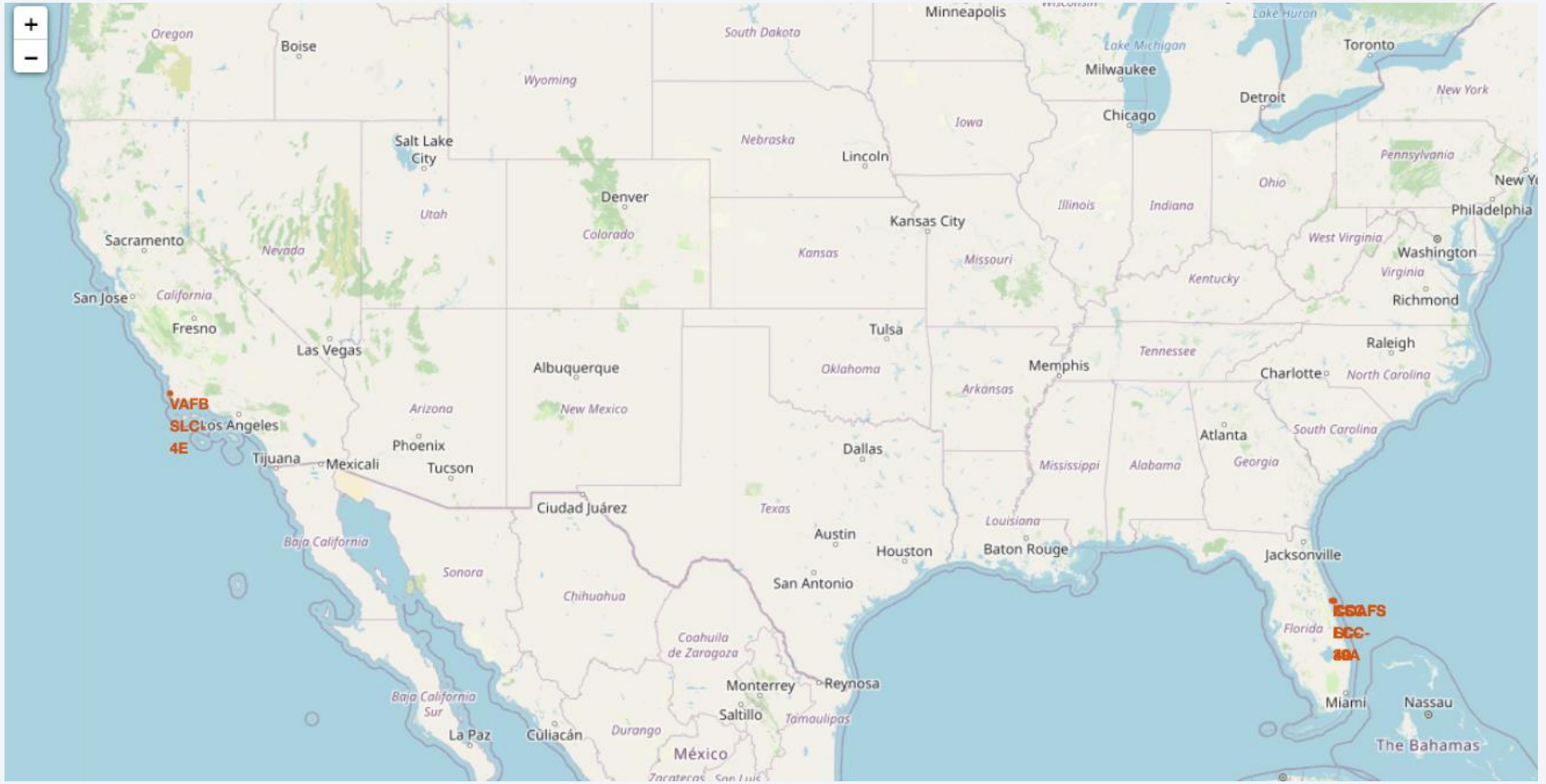
In [4]:
```
cur.execute("SELECT Landing_Outcome, COUNT(*) FROM SPACEXTABLE WHERE Date > '2010-06-04' AND Date < '2017-03-20'GROUP BY Landing_Outcome ORDER
cur.fetchall()
```

Out[4]:
```
[('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

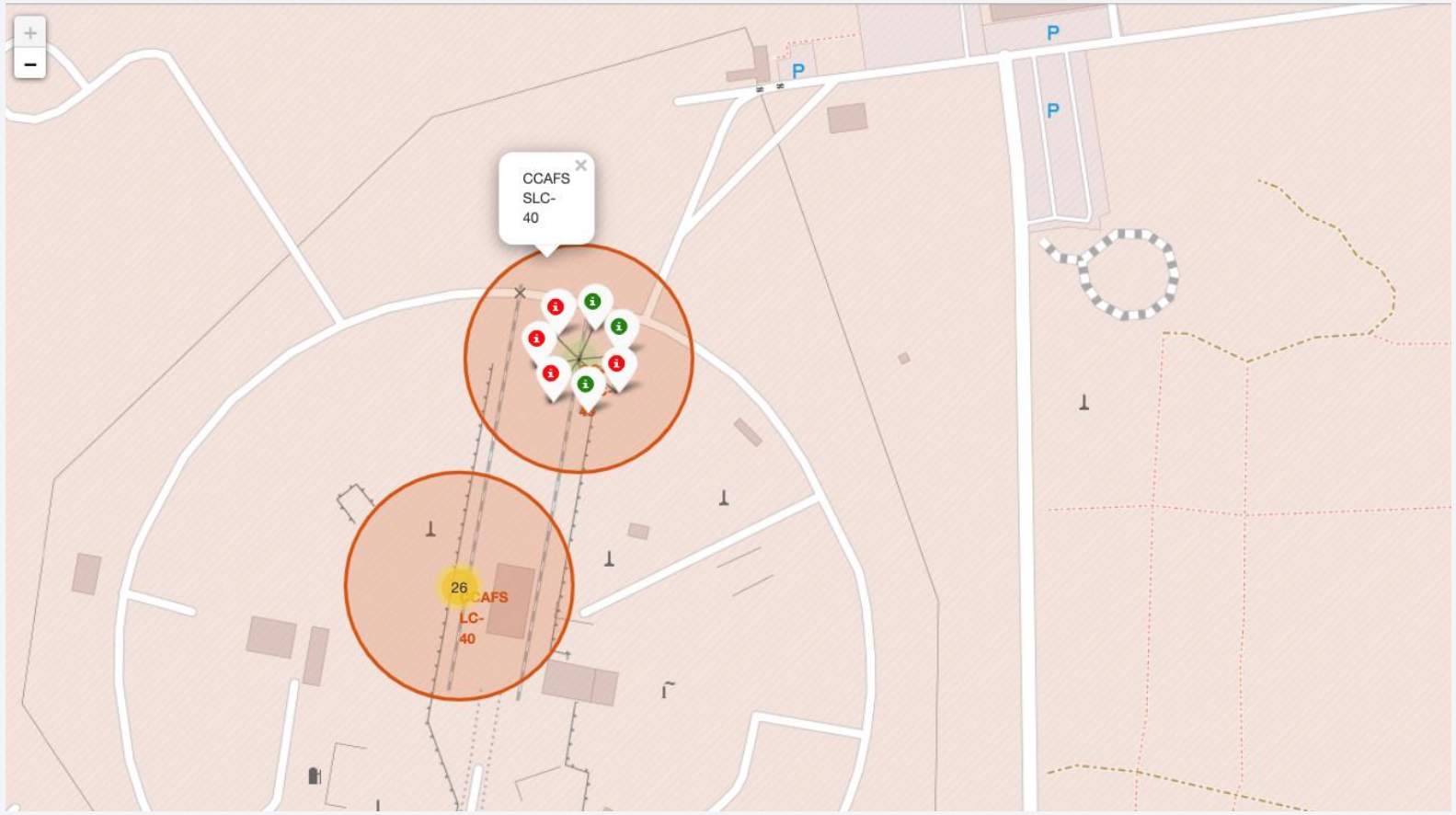
SpaceX Launch Sites



The launch sites are close to the coastal lines.

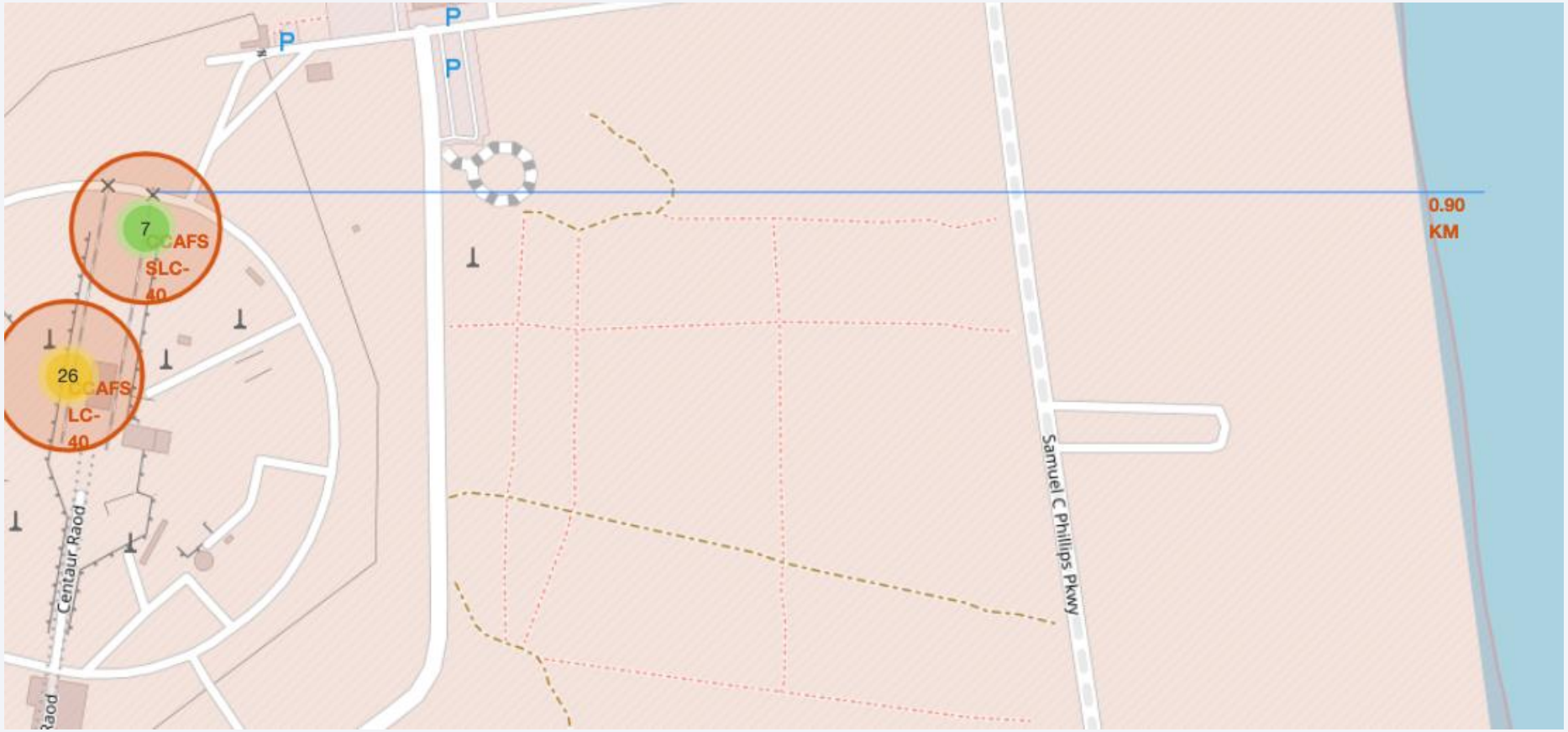
Launch outcomes for each site

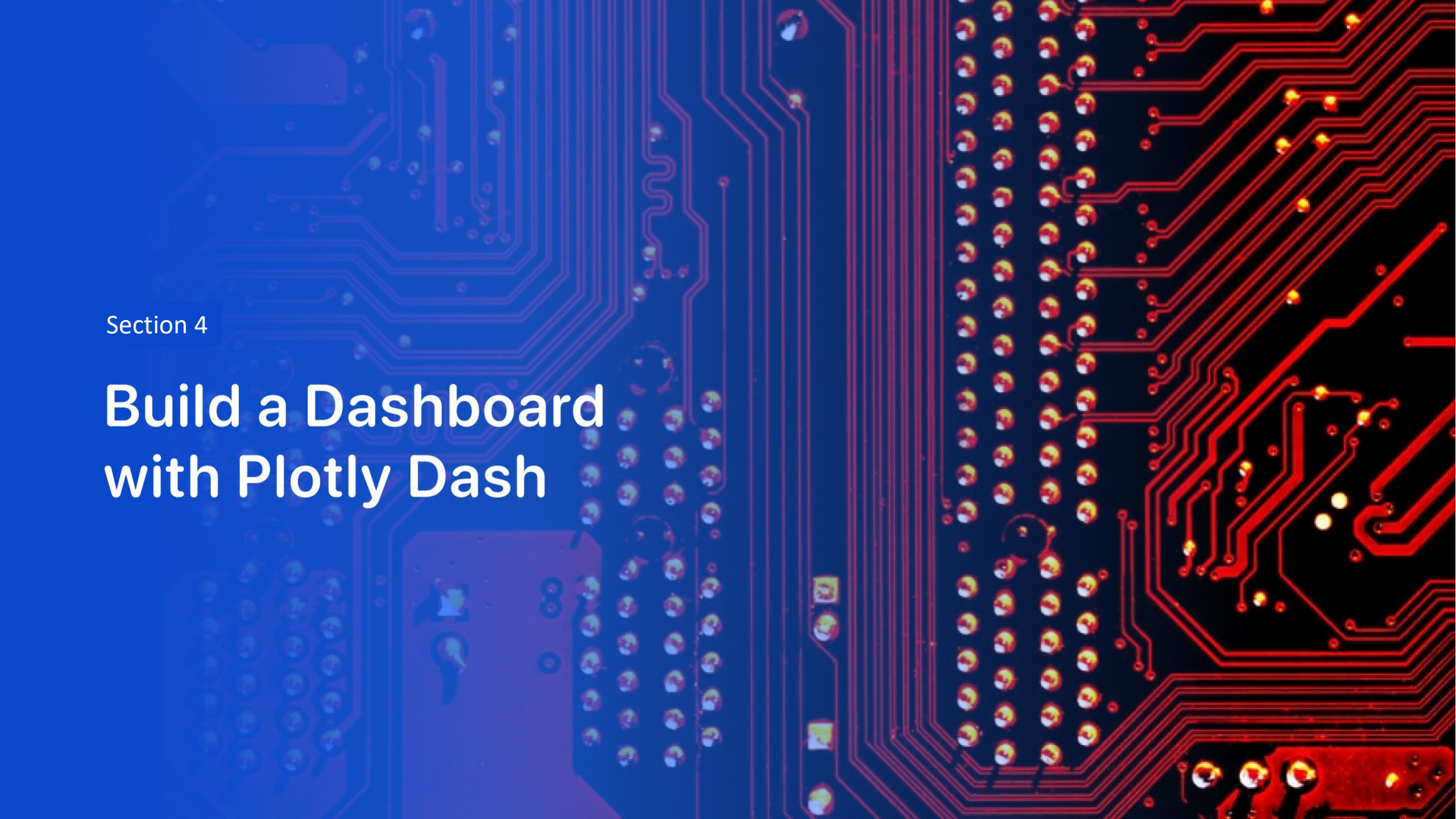
- From the color-labeled markers in marker clusters, you should be able to easily identify which launch sites have relatively high success rates.



Distance between launch site to its proximities

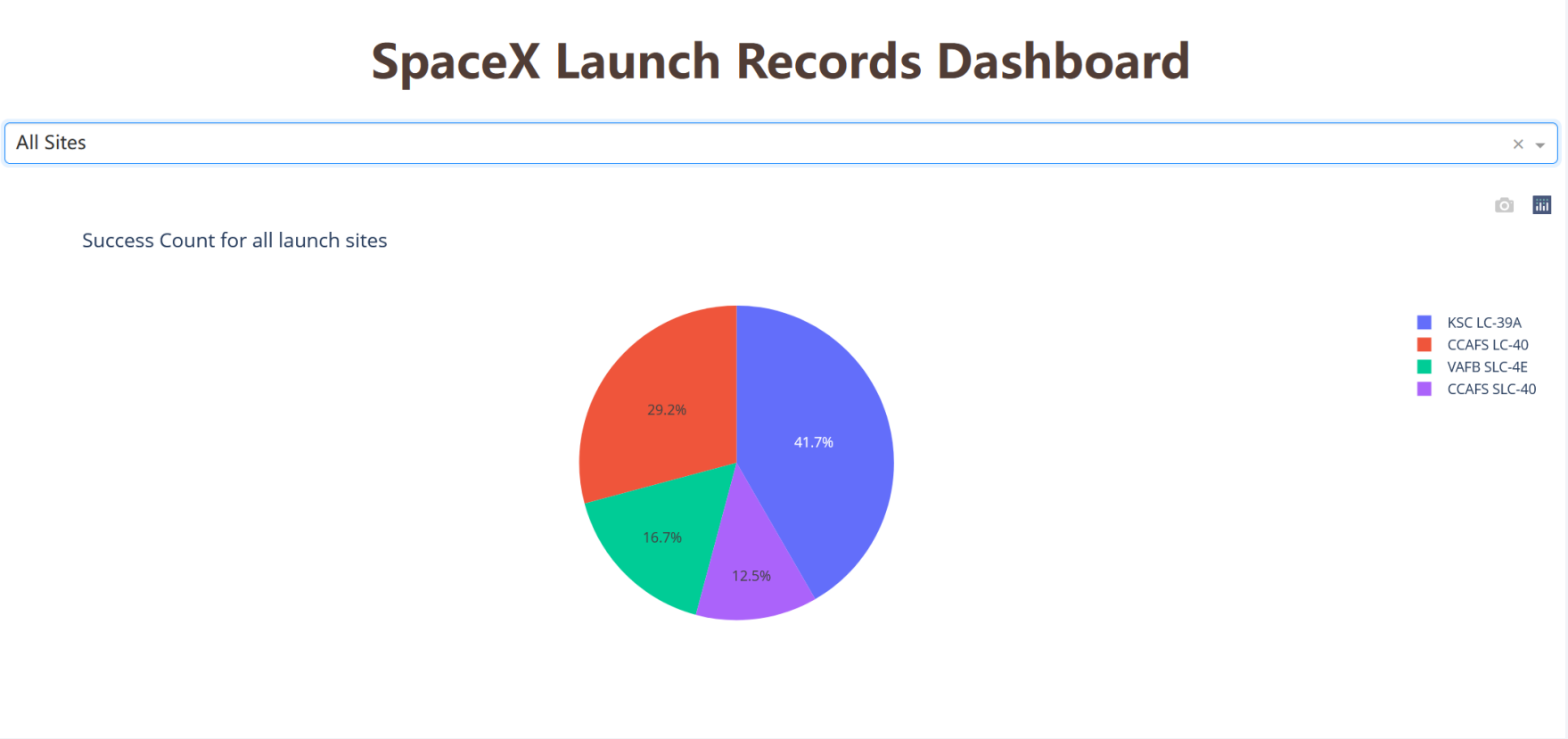
- Launch site SLC-40's proximity to coastline is 0.9km




Section 4
Build a Dashboard with Plotly Dash

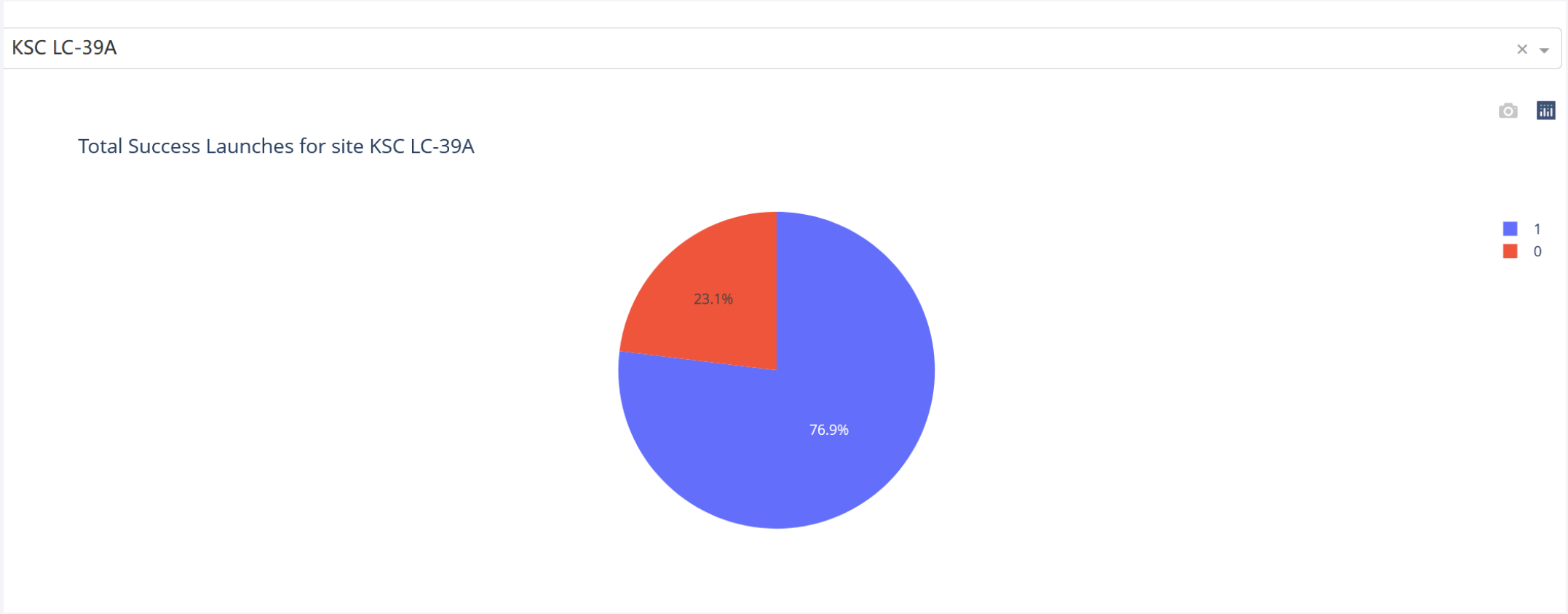
<Launch success for all sites>

- Show the screenshot of launch success count for all sites, in a piechart



Launch site with highest success ratio

- Show the screenshot of the piechart for the launch site with highest launch success ratio
- KSCLC-39A has the highest success ratio of 41.7%



Payload vs. Launch Outcome for all sites

- Show screenshots of Payload vs. Launch Outcome scatter plot for all sites, with different payload selected in the range slider

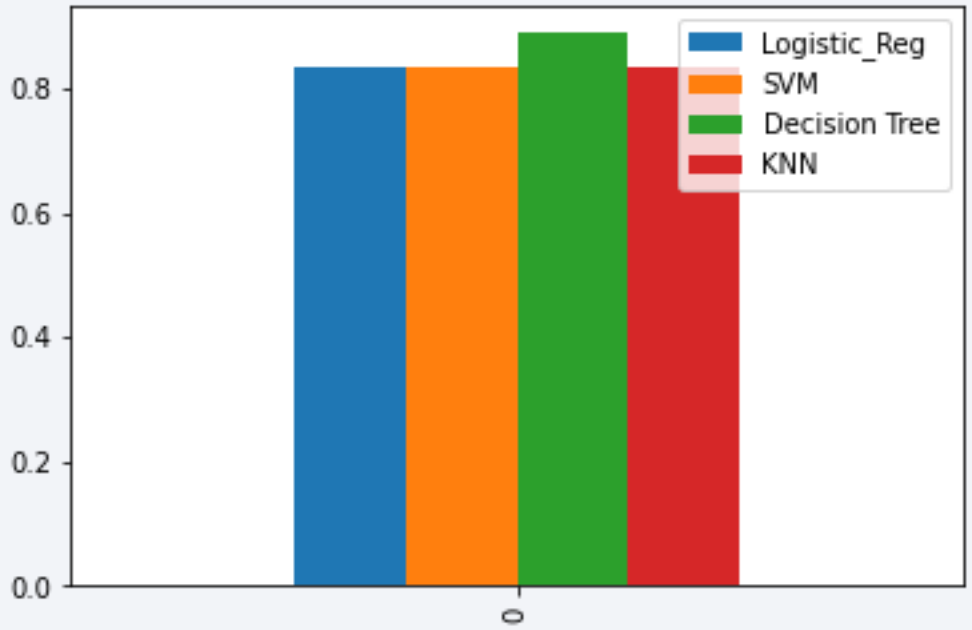
Section 5

Predictive Analysis (Classification)

Classification Accuracy

- Visualize the built model accuracy for all built classification models, in a bar chart

- Decision Tree have the highest classification accuracy



Confusion Matrix

- Show the confusion matrix of the best performing model

Conclusions

- Launch sites have varied success rates.
- An increase in flight number saw a corresponding increase in success rate.
- Four orbits, SSO, HEO, GEO and ES-L1 have the highest success rates of 100%
- There is no relationship between the number of flights and the success rate of the
- GTO orbit
- There is a relation between the success rate and the number of flights for the LEO
- orbit

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

https://github.com/icylizzie/MyProject

Thank you!