IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Juan Antonio Suwardi
August 2022

Outline

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

Executive Summary

- Summary of methodologies
  - Data Collection through API
  - Data Collection with Web Scraping
  - Data Wrangling
  - Exploratory Data Analysis with SQL
  - Exploratory Data Analysis with Data Visualization
  - Interactive Visual Analytics with Folium
  - Machine Learning Prediction
- Summary of all results
  - Exploratory Data Analysis result
  - Interactive analytics in screenshots
  - Predictive Analytics result

Introduction

- Project background and context

The commercial space age is here, the most successful is SpaceX . One reason SpaceX can do this is the rocket launches are relatively inexpensive. SpaceX advertises Falcon 9 rocket launches on its website with a cost of 62 million dollars; other providers cost upwards 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. Space Y that would like to compete with SpaceX founded by Billionaire industrialist Allon Musk. The main job is to determine the price of each launch. We will gathering information about Space X and creating dashboards and will train a machine learning model and use public information to predict if SpaceX will reuse the first stage.

- Problems you want to find answers
  - What factors determine if the rocket will land successfully?
  - The interaction amongst various features that determine the success rate of a successful landing.
  - What operating conditions needs to be in place to ensure a successful landing program.

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Data was collecting using SpaceX API and web scraping from Wikipedia.
- Perform data wrangling
  - Calculate feature and create label target.
- 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

- Method for collecting data:

1. Request to the SpaceX API
2. Download as .json & normalize with using .json_normalize().
3. Cleaning data: Missing value
4. Web scraping using BeautifulSoup
5. Extract from HTML & turn into dataframe

Data Collection – SpaceX API

- I used the get request to the SpaceX API to collect data, clean the requested data and did some basic data wrangling and formatting.

- Source link:

https://github.com/Juantonios1/IBM-Data-Science-Professional-Certificate/blob/main/Applied%20Data%20Science%20Capstone/Week%201/jupyter-labs-spacex-data-collection-api.ipynb

```
static_json_url='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/API_call_spacex_api.json'
```

We should see that the request was successfull with the 200 status response code

```
response.status_code
```

200

Now we decode the response content as a Json using `.json()` and turn it into a Pandas dataframe using `.json_normalize()`

```
# Use json_normalize meethod to convert the json result into a dataframe
data = pd.json_normalize(response.json())
```

Using the dataframe `data` print the first 5 rows

```
# Get the head of the dataframe
data.head()
```

	static_fire_date_utc	static_fire_date_unix	net	window	rocket	success	failures	details	crew	ships	capsules	payloads	
0	2006-03-17T00:00:00.000Z	1.142554e+09	False	0.0	5e9d0d95eda69955f709d1eb	False	[{'time': 33, 'altitude': None, 'reason': 'merlin engine failure'}]	Engine failure at 33 seconds and loss of vehicle	[]	[]	[]	[5eb0e4b5b6c3bb0006eeb1e1]	5e9e45(

Data Collection - Scraping

- I applied web scrapping to webscrap Falcon 9 launch records with BeautifulSoup and converted it into a pandas dataframe.

- Source link:

https://github.com/Juantonios1/IBM-Data-Science-Professional-Certificate/blob/main/Applied%20Data%20Science%20Capstone/Week%201/jupyter-labs-webscraping.ipynb

```python
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(data, 'html5lib')
```

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

```python
# Use soup.title attribute
print(soup.title)
```

```
<title>List of Falcon 9 and Falcon Heavy launches - Wikipedia</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

```python
# 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.

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

```
<table class="wikitable plainrowheaders collapsible" style="width: 100%;">
<tbody><tr>
<th scope="col">Flight No.
</th>
<th scope="col">Date and<br/>time (<a href="/wiki/Coordinated_Universal_Time" title="Coordinated Universal Time">UTC</a>)
```

Data Wrangling

```
# landing_class = 0 if bad_outcome
# landing_class = 1 otherwise
landing_class = []
for outcome in df['Outcome']:
    if outcome in bad_outcomes:
        landing_class.append(0)
    else:
        landing_class.append(1)
```

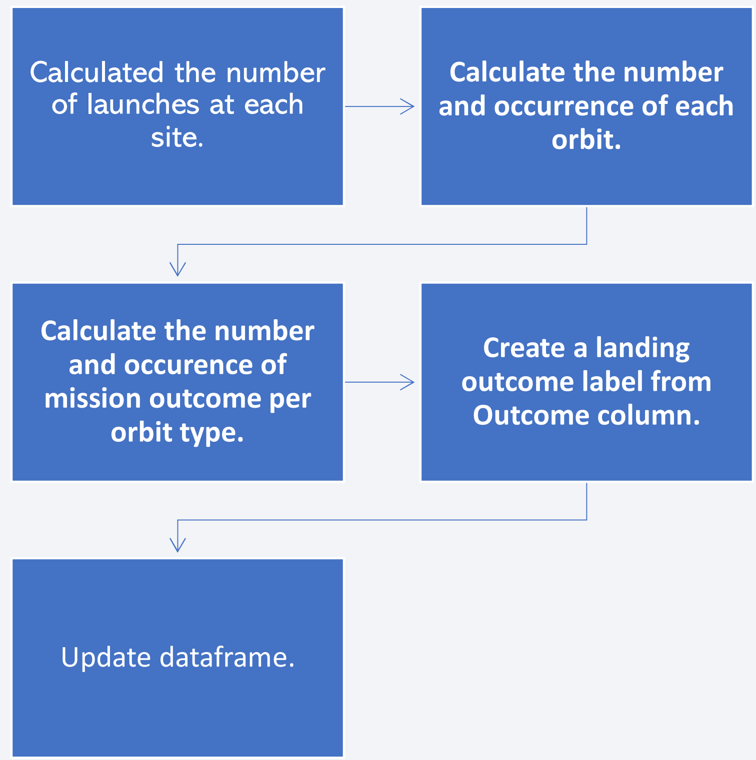
This variable will represent the classification variable that represents the outcome of each launch. If the value is zero, the first stage did not land successfully; one means the first stage landed Successfully

```
df['Class']=landing_class
df[['Class']].head(8)
```

	Class
0	0
1	0
2	0
3	0
4	0
5	0
6	1
7	1

```
df.head(5)
```

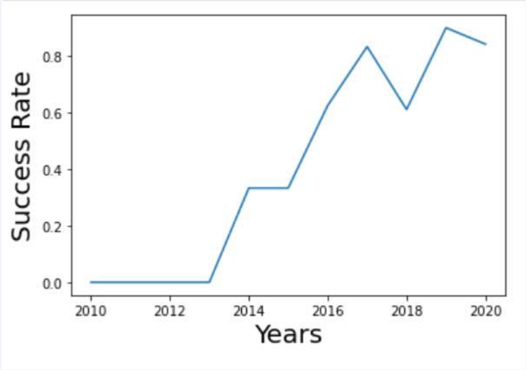
ber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block	ReusedCount	Serial	Longitude	Latitude	Class
1	2010-06-04	Falcon 9	6104.959412	LEO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B0003	-80.577366	28.561857	0
2	2012-05-22	Falcon 9	525.000000	LEO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B0005	-80.577366	28.561857	0
3	2013-03-01	Falcon 9	677.000000	ISS	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B0007	-80.577366	28.561857	0
4	2013-09-29	Falcon 9	500.000000	PO	VAFB SLC 4E	False Ocean	1	False	False	False	NaN	1.0	0	B1003	-120.610829	34.632093	0
5	2013-12-03	Falcon 9	3170.000000	GTO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B1004	-80.577366	28.561857	0

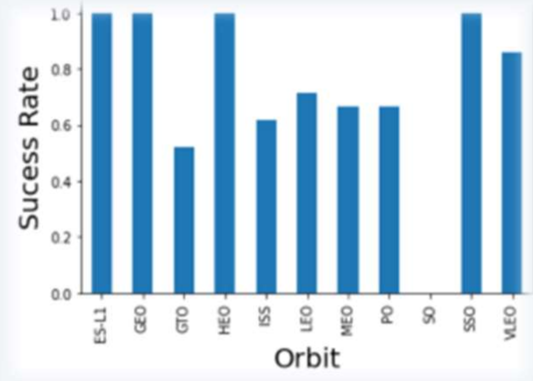


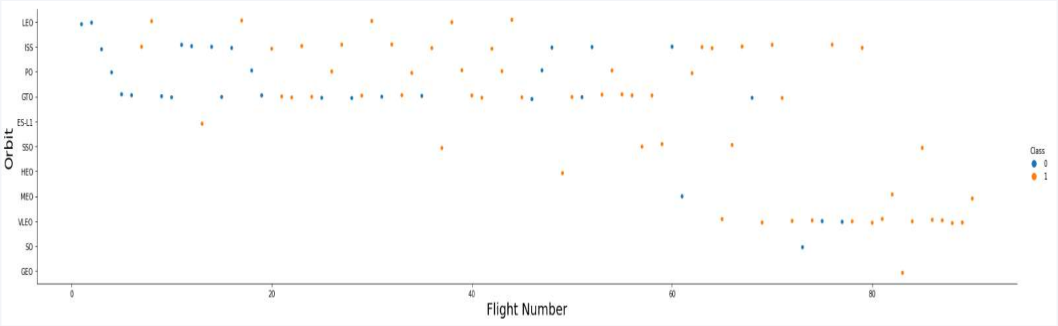
Source link:
https://github.com/Juantonios1/IBM-Data-Science-Professional-Certificate/blob/main/Applied%20Data%20Science%20Capstone/Week%201/labs-jupyter-spacex-Data%20wrangling.ipynb

EDA with Data Visualization

- Data visualization:
  - **Relationship between Flight Number and Launch Site.**
  - **Relationship between Payload and Launch Site.**
  - **relationship between success rate of each orbit type**
  - **Relationship between FlightNumber and Orbit type**
  - **Relationship between Payload and Orbit type.**
  - **Launch success yearly trend**







Source link:
https://github.com/Juantonios1/IBM-Data-Science-Professional-Certificate/blob/main/Applied%20Data%20Science%20Capstone/Week%202/jupyter-labs-eda-dataviz.ipynb

EDA with SQL

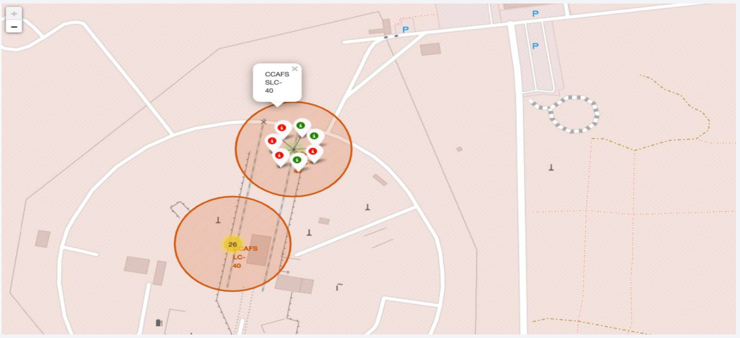
- I loaded the SpaceX dataset into using %load_ext sql in jupyter notebook.
- I applied EDA with SQL to get insight from the data. We wrote queries to find out for instance:
  - The names of unique launch sites in the space mission.
  - The total payload mass carried by boosters launched by NASA (CRS)
  - The average payload mass carried by booster version F9 v1.1
  - The total number of successful and failure mission outcomes
  - The failed landing outcomes in drone ship, their booster version and launch site names.
- Source link:

https://github.com/Juantonios1/IBM-Data-Science-Professional-Certificate/blob/main/Applied%20Data%20Science%20Capstone/Week%202/jupyter-labs-eda-sql-coursera_sqllite.ipynb

Build an Interactive Map with Folium

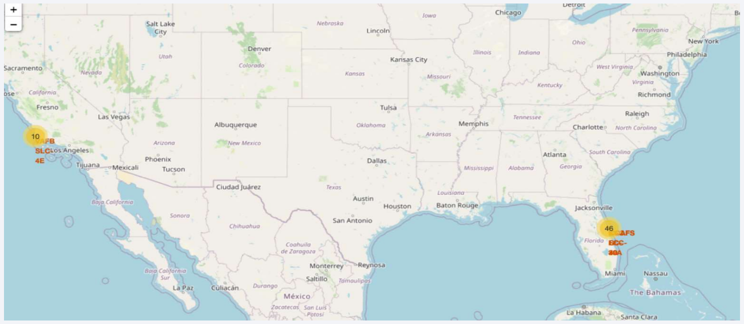
Summary:

- 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.



Finding

- Launch sites are in close proximity to railways.
- Launch sites are in close proximity to highways.
- Launch sites are in close proximity to coastline.
- Launch sites are not in close proximity to cities.



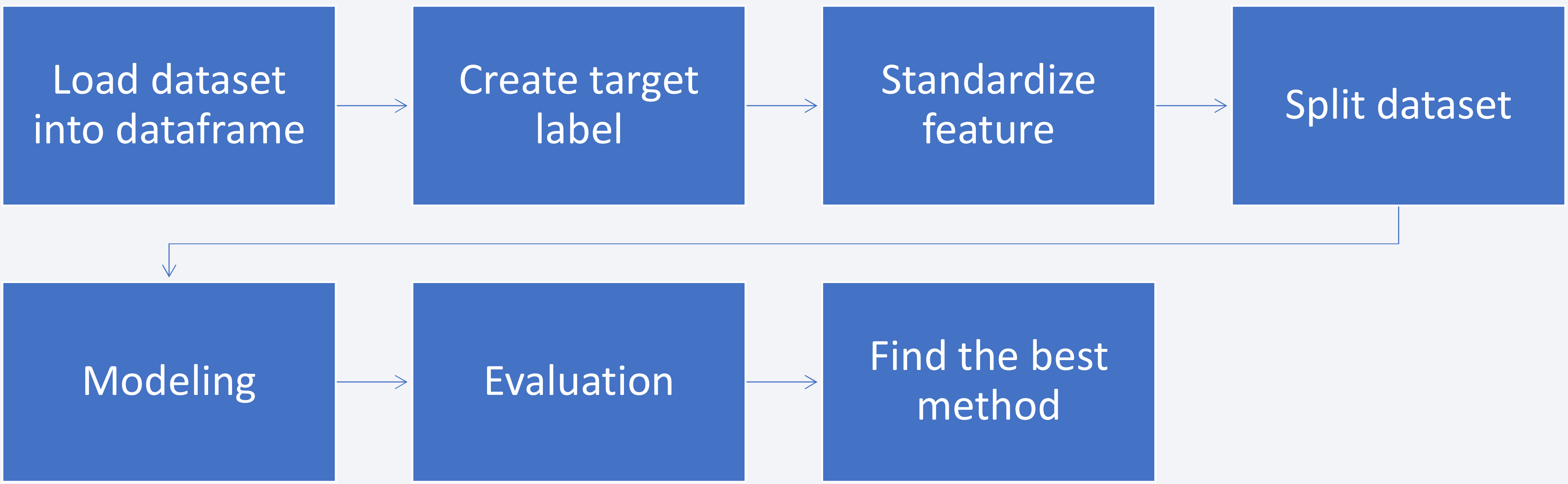
Source link:
https://github.com/Juantonios1/IBM-Data-Science-Professional-Certificate/blob/main/Applied%20Data%20Science%20Capstone/Week%203/lab_jupyter_launch_site_location.ipynb

Build a Dashboard with Plotly Dash

- We built an interactive dashboard with Plotly dash
- We plotted pie charts showing the total launches by a certain sites
- We plotted scatter graph showing the relationship with Outcome and Payload Mass (Kg) for the different booster version.
- The interactive plot make the information more easy to understand.

Source link:
https://github.com/Juantonios1/IBM-Data-Science-Professional-Certificate/blob/main/Applied%20Data%20Science%20Capstone/Week%203/lab_jupyter_launch_site_location.ipynb

Predictive Analysis (Classification)



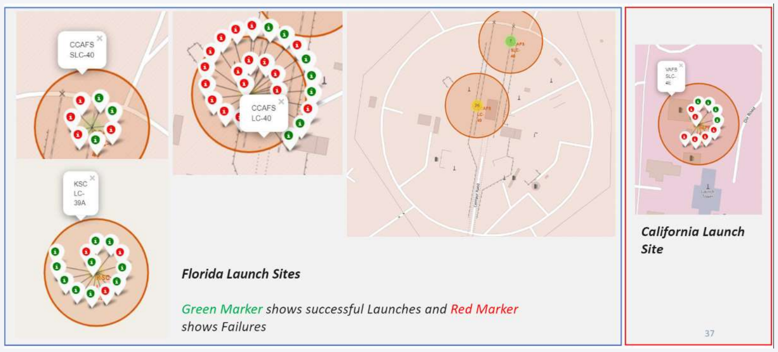
Source link:
https://github.com/Juantonios1/IBM-Data-Science-Professional-Certificate/blob/main/Applied%20Data%20Science%20Capstone/Week%204/SpaceX_Machine%20Learning%20Prediction_Part_5.ipynb

Accuracy for Support Vector Machine method: 0.8333333333333334

Results

- Exploratory data analysis results
  - Launch sites are in close proximity to railways.
  - Launch sites are in close proximity to highways.
  - Launch sites are in close proximity to coastline.
  - Launch sites are not in close proximity to cities.
- Predictive analysis results
  - Accuracy for Logistics Regression method: 0.846 .
  - Accuracy for Support Vector Machine method: 0.848.
  - Accuracy for Decision Tree method: 0.877
  - Accuracy for KNN method: 0.848.

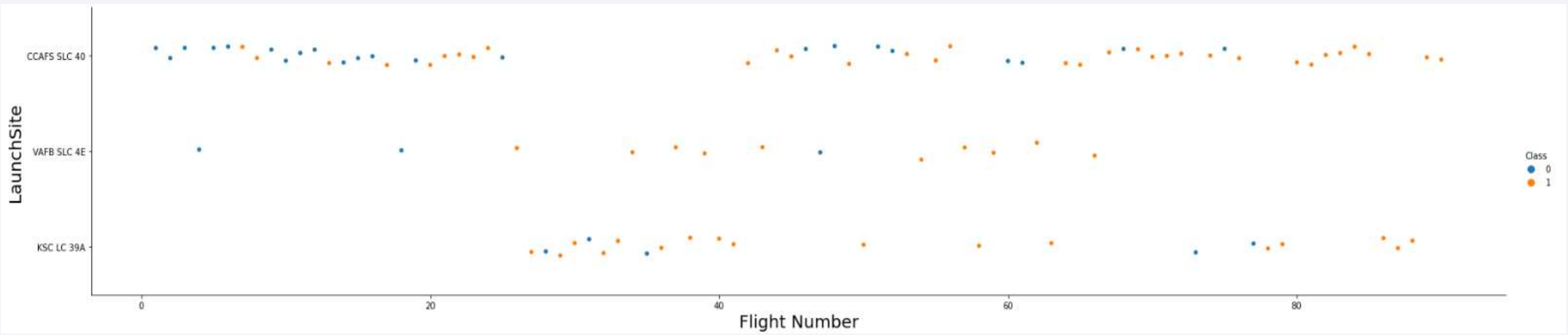
- Interactive analytics demo



Section 2

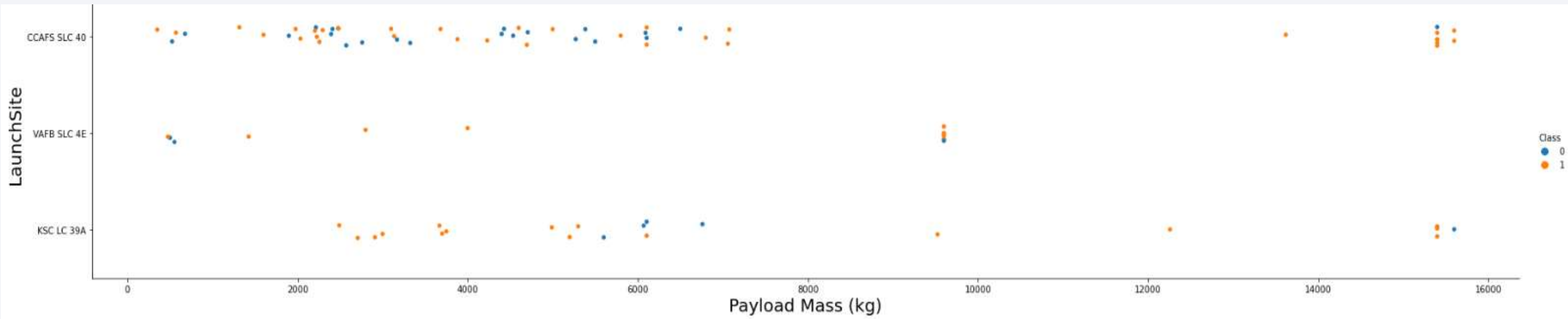
Insights drawn from EDA

Flight Number vs. Launch Site



The larger the flight amount at a launch site, the greater the success rate at a launch site.

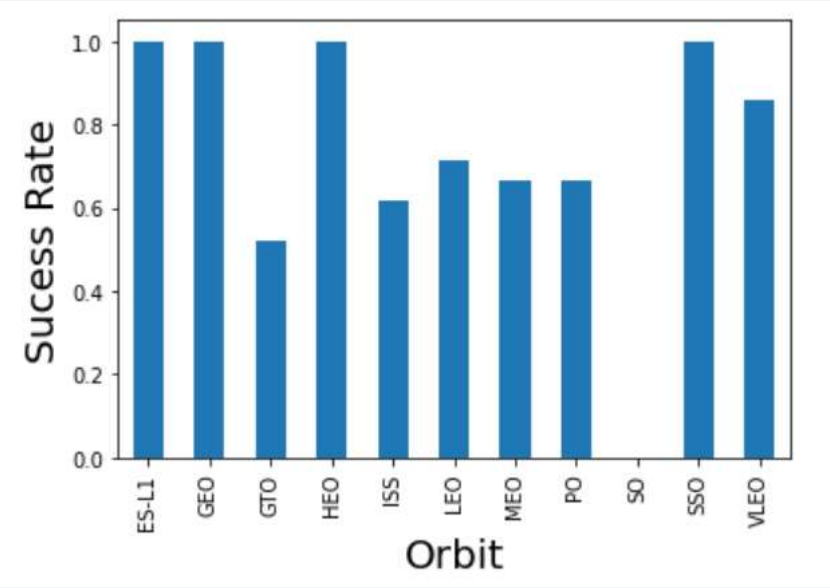
Payload vs. Launch Site



The VAFB-SLC launch site don’t have any rockets launched for heavypayload mass(greater than 10000).

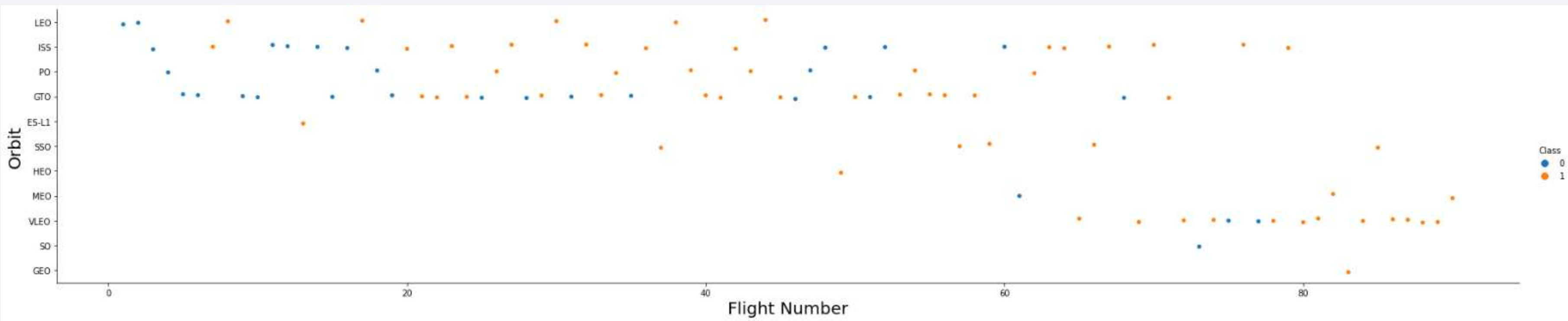
Success Rate vs. Orbit Type

The ES-L1, GEO, HEO, SSO, VLEO had the most success rate.

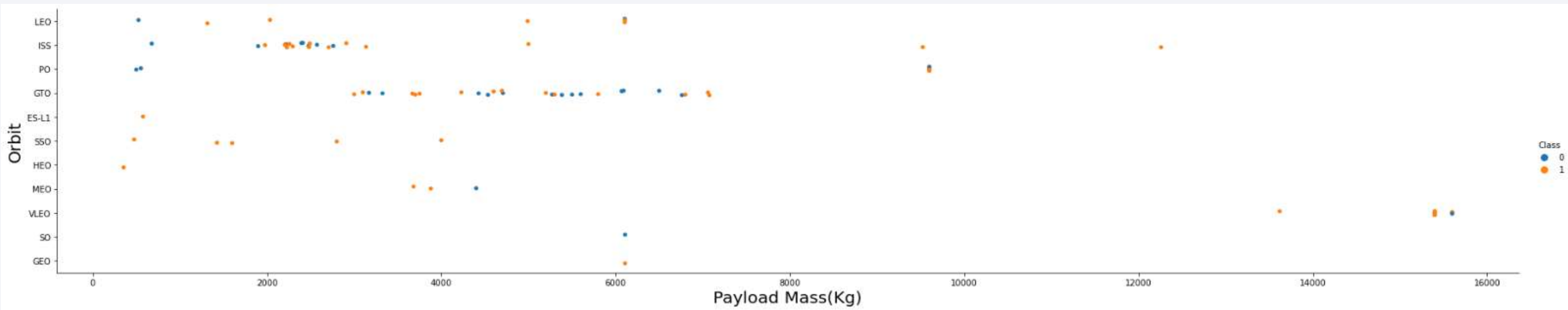


Flight Number vs. Orbit Type

The plot below shows the Flight Number vs. Orbit type. We observe that in the LEO orbit, success is related to the number of flights whereas in the GTO orbit, there is no relationship between flight number and the orbit.

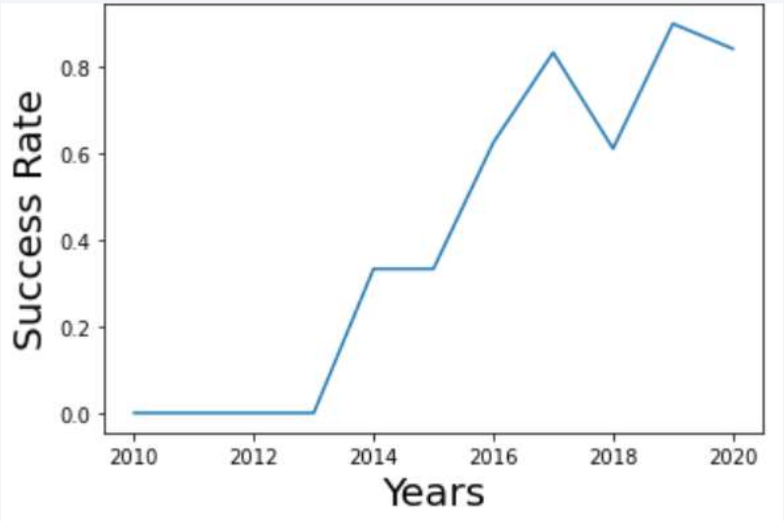


Payload vs. Orbit Type



We can observe that with heavy payloads, the successful landing are more for PO, LEO and ISS orbits.

Launch Success Yearly Trend



From the plot, we can observe that success rate since 2013 kept on increasing till 2020.

All Launch Site Names

We used the key word **DISTINCT** to show only unique launch sites from the SpaceX data.

In [13]:
```
%sql select distinct(LAUNCH_SITE) from SPACEXTBL
```

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

Out[13]:

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

Launch Site Names Begin with 'CCA'

- We define condition for Launchsite begin with ‘CCA and limit the result for only first 5 data.

In [14]:
```
%sql select * from SPACEXTBL where LAUNCH_SITE like 'CCA%' limit 5
```

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

Out[14]:

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing _Outcome
04-06-2010	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
08-12-2010	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)
22-05-2012	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
08-10-2012	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
01-03-2013	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

Total Payload Mass

Total payload can calculate with add all payload mass with condition 'NASA (CRS)'

```
In [15]: %sql select sum(PAYLOAD_MASS__KG_) from SPACEXTBL where CUSTOMER = 'NASA (CRS)'

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

Out[15]:

sum(PAYLOAD_MASS__KG_)
45596

Average Payload Mass by F9 v1.1

Average payload can calculate with add all payload mass with condition BOOSTER_VERSION = ‘F9 v1.1’ and divide with number of sample.

```
In [17]: %sql select avg(PAYLOAD_MASS__KG_) from SPACEXTBL where BOOSTER_VERSION = 'F9 v1.1'

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

Out[17]:

avg(PAYLOAD_MASS__KG_)
2928.4

First Successful Ground Landing Date

We observed that the dates of the first successful landing outcome on ground pad was 22nd December 2015

In [14]:
```
task_5 = '''
        SELECT MIN(Date) AS FirstSuccessfull_landing_date
        FROM SpaceX
        WHERE LandingOutcome LIKE 'Success (ground pad)'
        '''
create_pandas_df(task_5, database=conn)
```

Out[14]:

	firstsuccessfull_landing_date
0	2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

In [15]:

```
task_6 = '''
        SELECT BoosterVersion
        FROM SpaceX
        WHERE LandingOutcome = 'Success (drone ship)'
            AND PayloadMassKG > 4000
            AND PayloadMassKG < 6000
        '''
create_pandas_df(task_6, database=conn)
```

Out[15]:

	boosterversion
0	F9 FT B1022
1	F9 FT B1026
2	F9 FT B1021.2
3	F9 FT B1031.2

We used the WHERE clause to filter for boosters which have successfully landed on drone ship and applied the AND condition to determine successful landing with payload mass greater than 4000 but less than 6000

Total Number of Successful and Failure Mission Outcomes

List the total number of successful and failure mission outcomes

In [16]:
```
task_7a = '''
        SELECT COUNT(MissionOutcome) AS SuccessOutcome
        FROM SpaceX
        WHERE MissionOutcome LIKE 'Success%'
        '''

task_7b = '''
        SELECT COUNT(MissionOutcome) AS FailureOutcome
        FROM SpaceX
        WHERE MissionOutcome LIKE 'Failure%'
        '''
print('The total number of successful mission outcome is:')
display(create_pandas_df(task_7a, database=conn))
print()
print('The total number of failed mission outcome is:')
create_pandas_df(task_7b, database=conn)
```

```
The total number of successful mission outcome is:
```

	successoutcome
0	100

```
The total number of failed mission outcome is:
```

Out[16]:

	failureoutcome
0	1

We used wildcard like '%' to filter for WHERE MissionOutcome was a success or a failure.

Boosters Carried Maximum Payload

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

In [17]:

```
task_8 = '''
        SELECT BoosterVersion, PayloadMassKG
        FROM SpaceX
        WHERE PayloadMassKG = (
                                SELECT MAX(PayloadMassKG)
                                FROM SpaceX
                                )
        ORDER BY BoosterVersion
        '''
create_pandas_df(task_8, database=conn)
```

Out[17]:

	boosterversion	payloadmasskg
0	F9 B5 B1048.4	15600
1	F9 B5 B1048.5	15600
2	F9 B5 B1049.4	15600
3	F9 B5 B1049.5	15600
4	F9 B5 B1049.7	15600
5	F9 B5 B1051.3	15600
6	F9 B5 B1051.4	15600
7	F9 B5 B1051.6	15600
8	F9 B5 B1056.4	15600
9	F9 B5 B1058.3	15600
10	F9 B5 B1060.2	15600
11	F9 B5 B1060.3	15600

We determined the booster that have carried the maximum payload using a subquery in the **WHERE** clause and the **MAX()** function.

2015 Launch Records

- We used a combinations of the WHERE clause, LIKE, AND, and BETWEEN conditions to filter for failed landing outcomes in drone ship, their booster versions, and launch site names for year 2015

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

In [18]:
```
task_9 = '''
        SELECT BoosterVersion, LaunchSite, LandingOutcome
        FROM SpaceX
        WHERE LandingOutcome LIKE 'Failure (drone ship)'
            AND Date BETWEEN '2015-01-01' AND '2015-12-31'
        '''
create_pandas_df(task_9, database=conn)
```

Out[18]:

	boosterversion	launchsite	landingoutcome
0	F9 v1.1 B1012	CCAFS LC-40	Failure (drone ship)
1	F9 v1.1 B1015	CCAFS LC-40	Failure (drone ship)

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))

In [19]:

```
task_10 = '''
        SELECT LandingOutcome, COUNT(LandingOutcome)
        FROM SpaceX
        WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20'
        GROUP BY LandingOutcome
        ORDER BY COUNT(LandingOutcome) DESC
        '''
create_pandas_df(task_10, database=conn)
```

Out[19]:

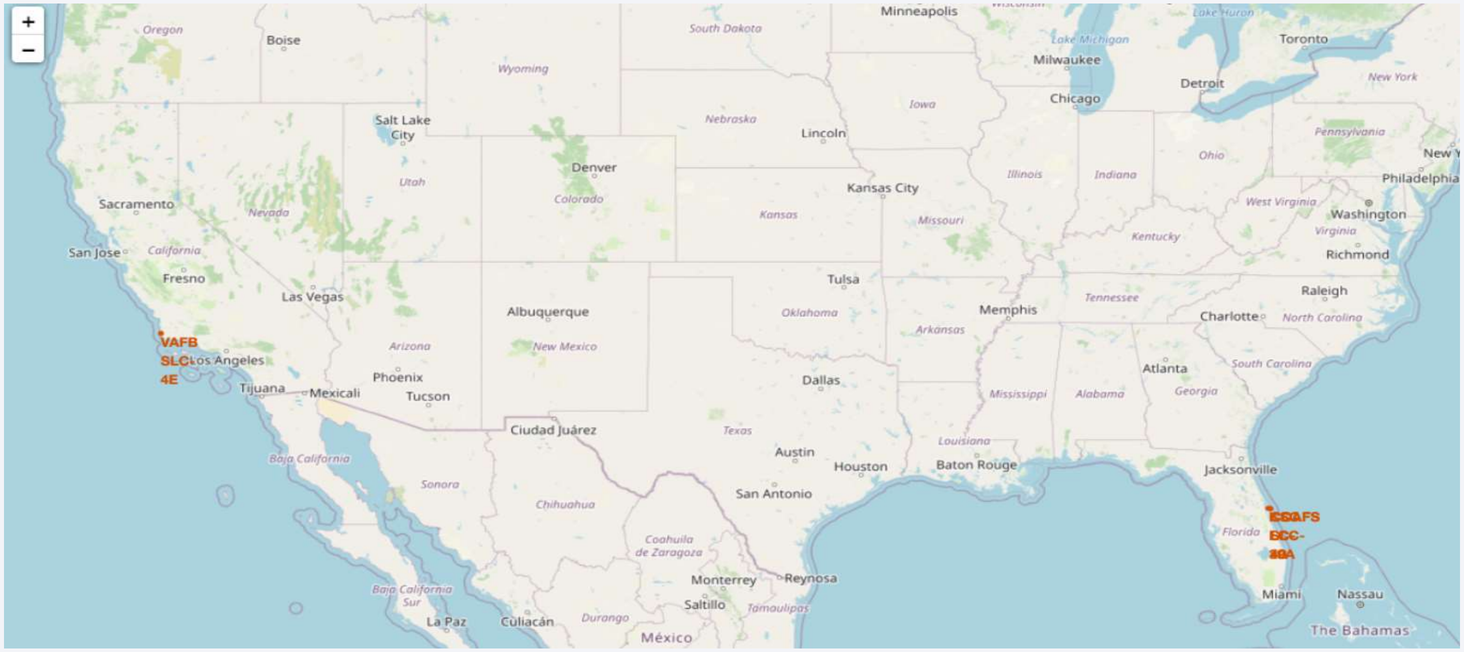
	landingoutcome	count
0	No attempt	10
1	Success (drone ship)	6
2	Failure (drone ship)	5
3	Success (ground pad)	5
4	Controlled (ocean)	3
5	Uncontrolled (ocean)	2
6	Precluded (drone ship)	1
7	Failure (parachute)	1

- We selected Landing outcomes and the COUNT of landing outcomes from the data and used the WHERE clause to filter for landing outcomes BETWEEN 2010-06-04 to 2010-03-20.
- We applied the GROUP BY clause to group the landing outcomes and the ORDER BY clause to order the grouped landing outcome in descending order.

Section 3

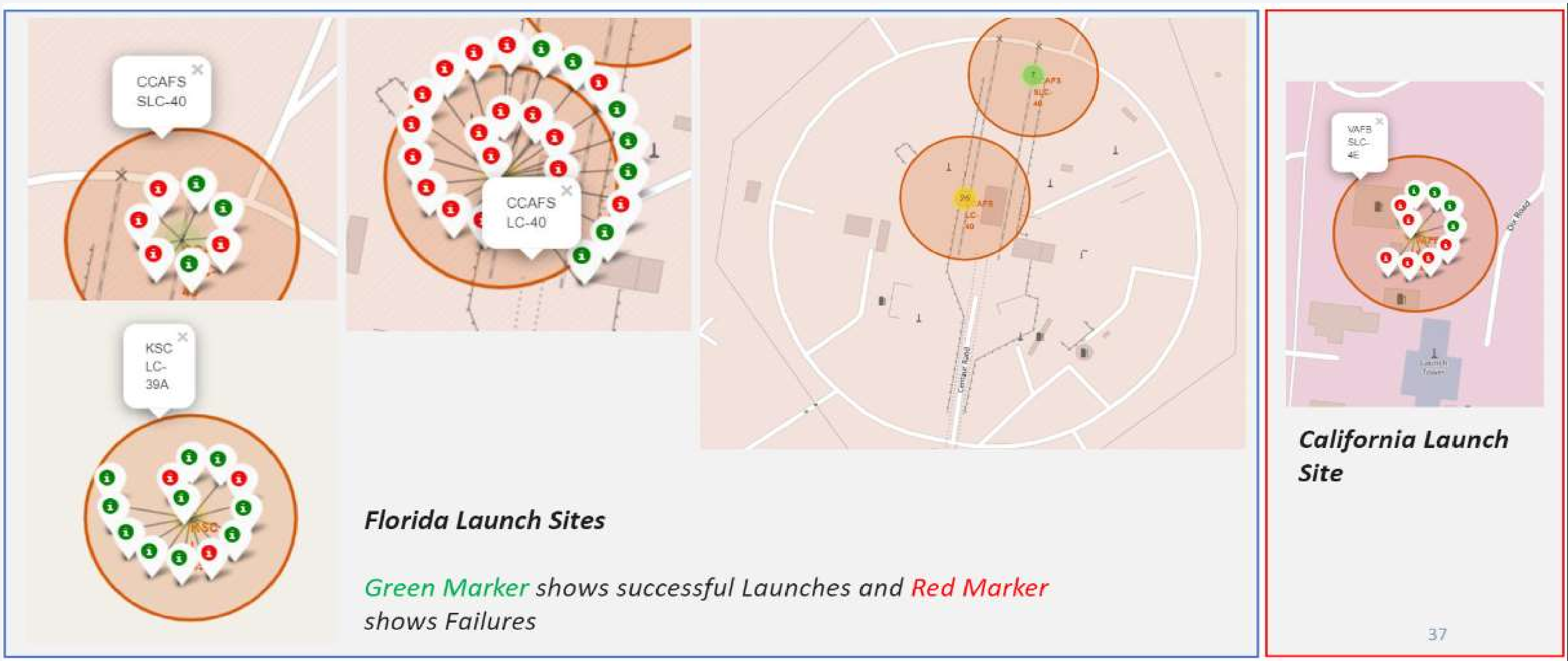
Launch Sites Proximities Analysis

All launch sites global map markers

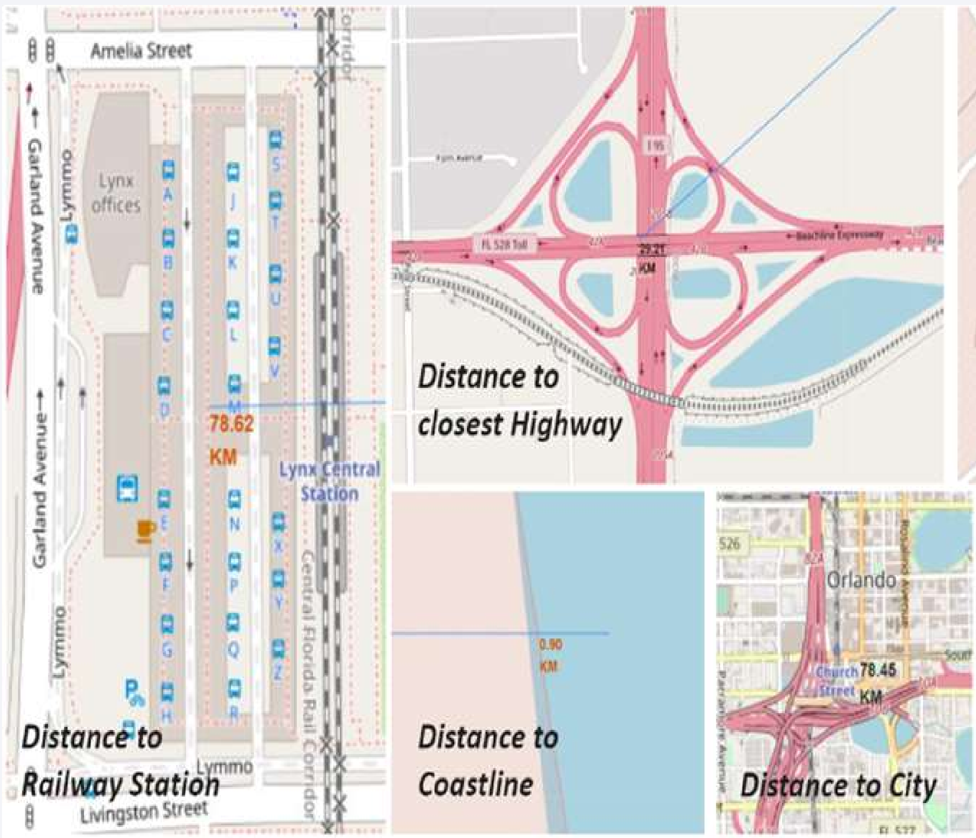


The SpaceX launch sites are in United States of America coasts (Florida & California)

Markers showing launch sites with color labels



Launch Site distance to landmarks

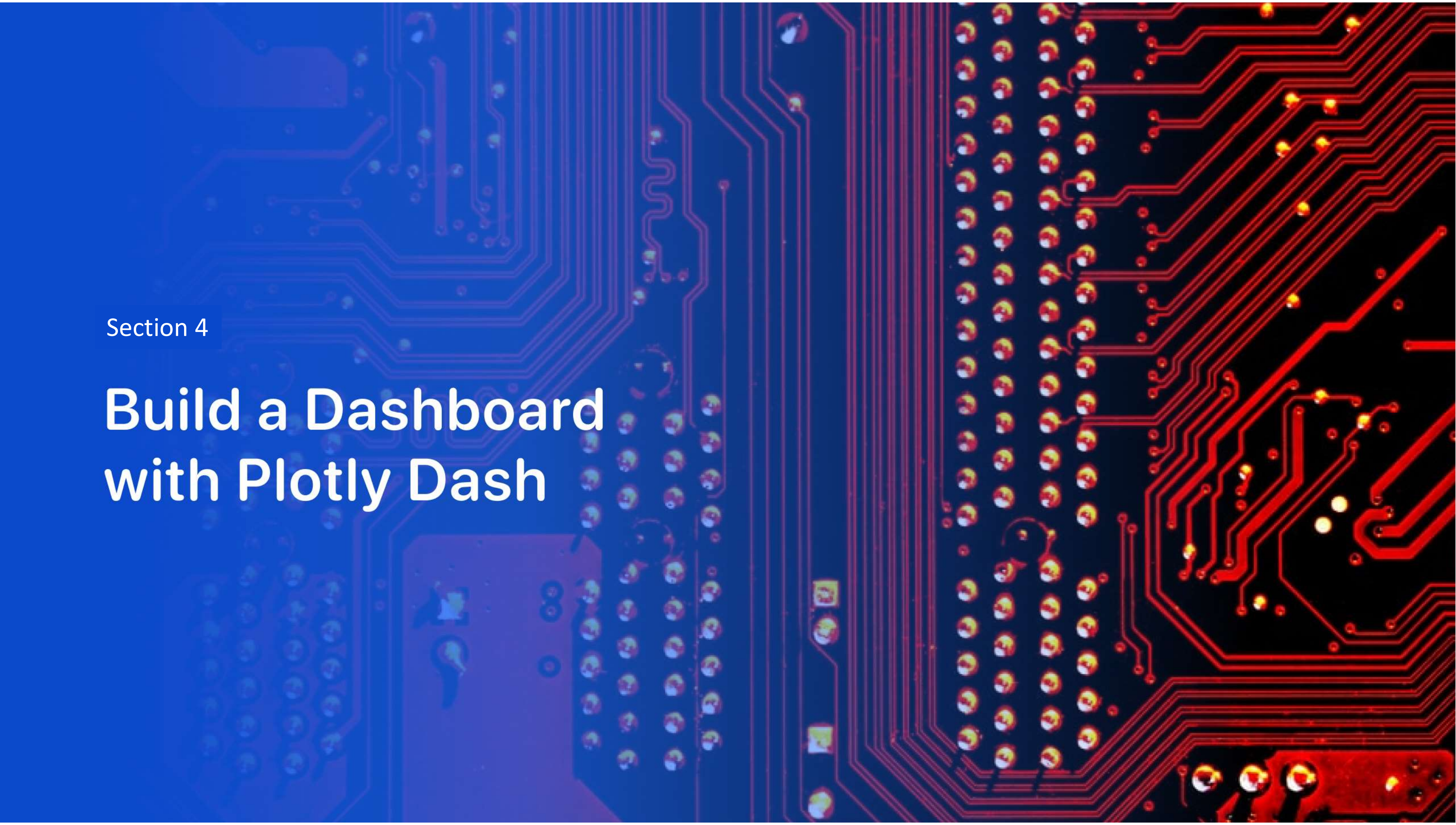


Launch sites are in close proximity to railways.

Launch sites are in close proximity to highways.

Launch sites are in close proximity to coastline.

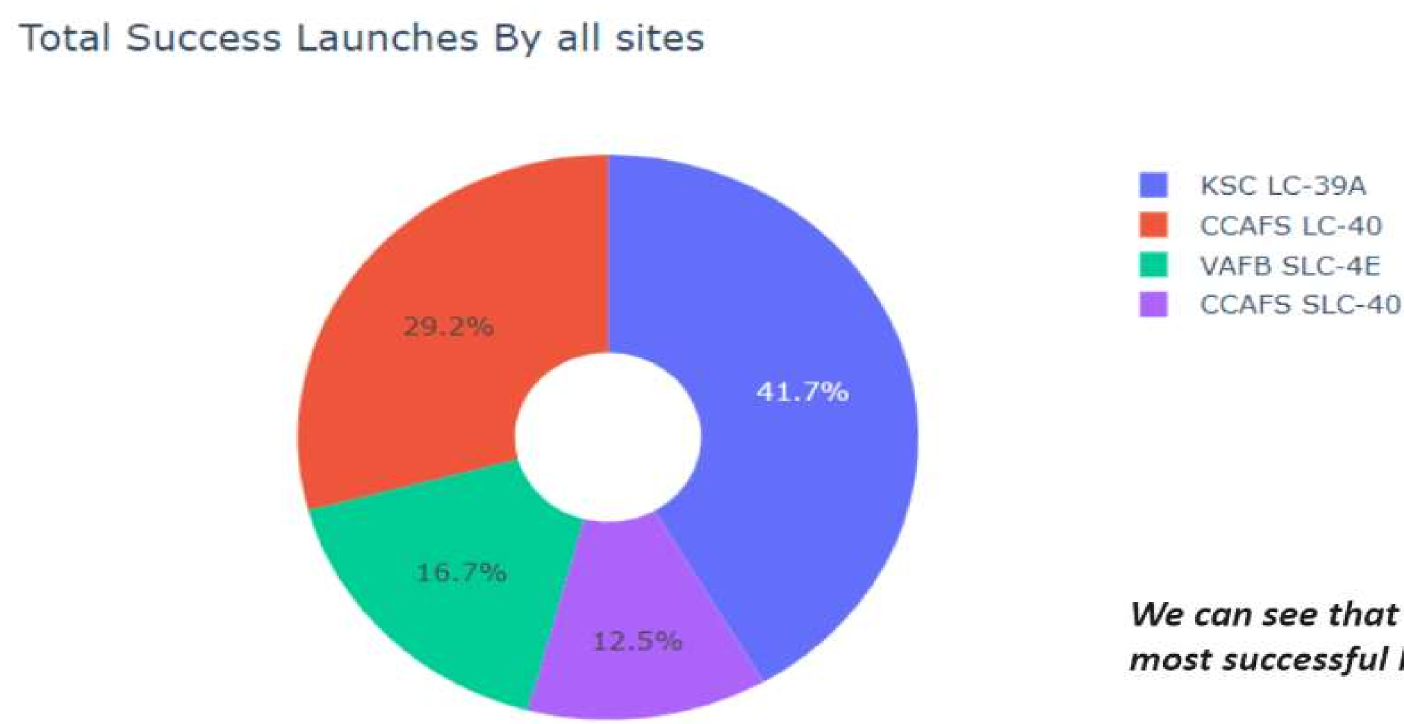
Launch sites are not in close proximity to cities.



Section 4

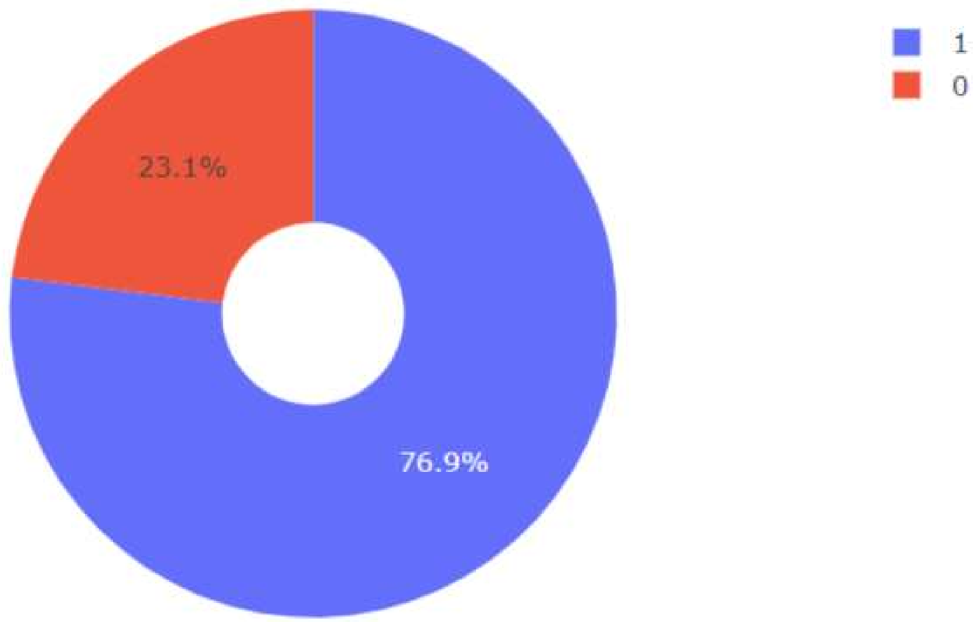
Build a Dashboard with Plotly Dash

The success percentage achieved by each launch site



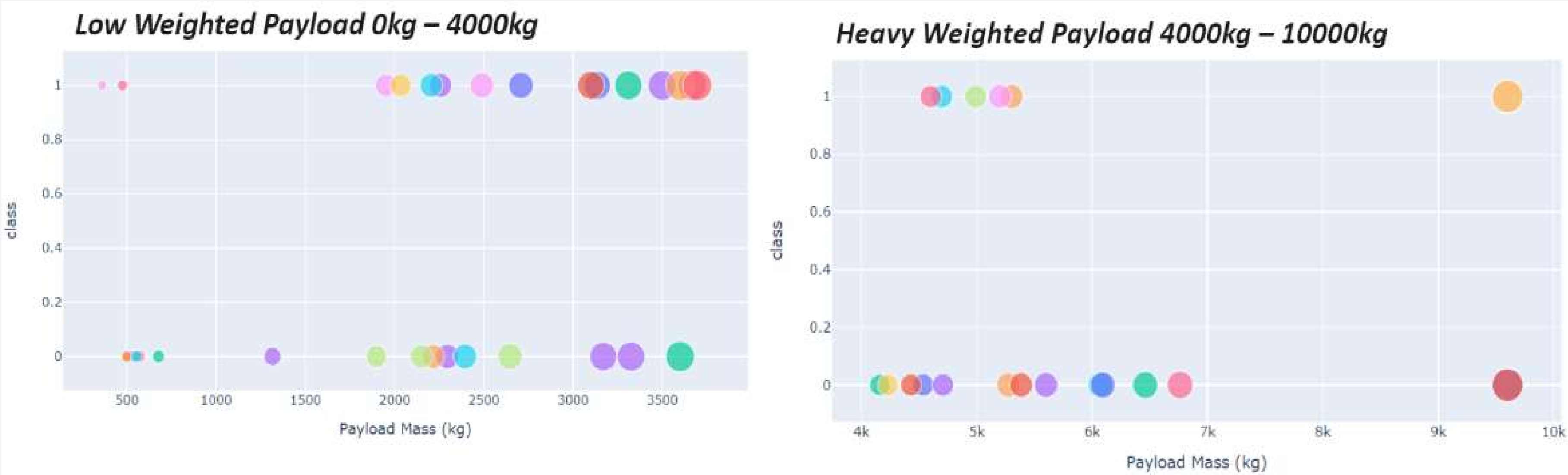
We can see that KSC LC-39A had the most successful launches from all the sites

Launch site with the highest launch success ratio



KSC LC-39A achieved a 76.9% success rate while getting a 23.1% failure rate

Payload vs Launch Outcome for all sites

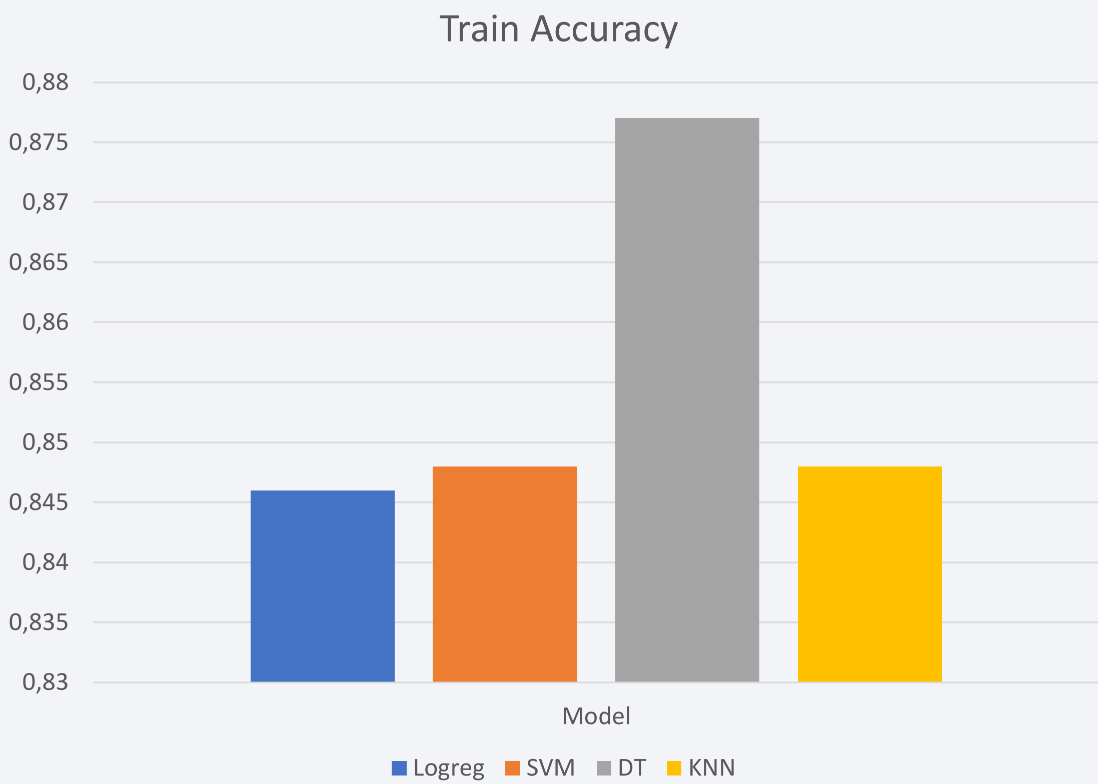


We can see the success rates for low weighted payloads is higher than the heavy weighted payloads

Section 5

Predictive Analysis (Classification)

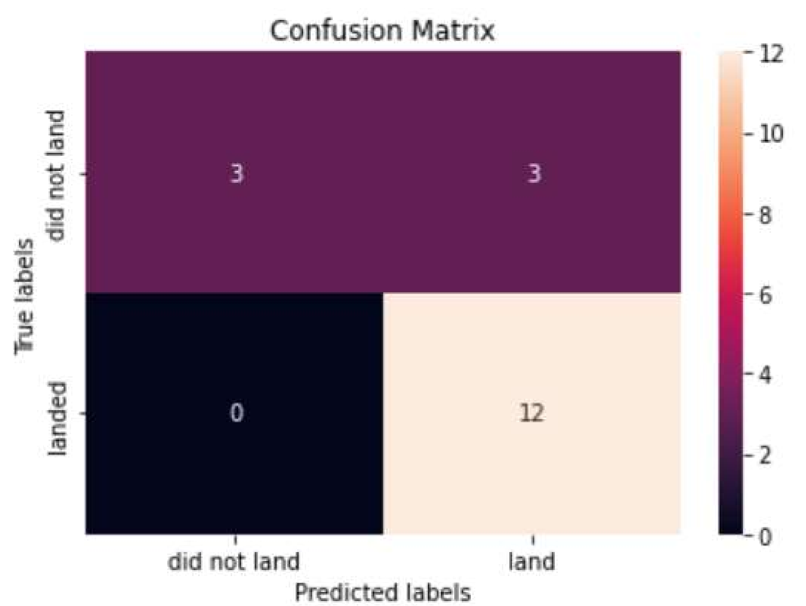
Classification Accuracy



- Test accuracy for all model: 0.834
- The best model: Decision Tree

Confusion Matrix

```
yhat = svm_cv.predict(X_test)
plot_confusion_matrix(Y_test,yhat)
```



- 12 data which predicted landed is correct.
- There no prediction for didn't land
- There are 3 which predict land actual not and vice versa.

Conclusions

- The larger the flight amount at a launch site, the greater the success rate at a launch site.
- Launch success rate started to increase in 2013 till 2020.
- Orbits ES-L1, GEO, HEO, SSO, VLEO had the most success rate.
- KSC LC-39A had the most successful launches of any sites.
- The Decision tree classifier is the best machine learning algorithm for this task.

Thank you!