IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Zhang Sizhe
2024-12-12

Outline

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

Executive Summary

A rival company to SpaceX uses Falcon 9 rocket data to determine the rocket first stage landing success rate as well as the cost of a launch.

- Summary of methodologies
  - Data collection
  - Data wrangling
  - Exploratory data analysis with data visualization and SQL
  - Predictive analysis
- Summary of all results
  - Exploratory data analysis results
  - Interactive analytics demo in screenshots
  - Predictive analysis results

Introduction

- Project background and context
    - Part of IBM data science professional certificate
    - Demonstrate proficiency in data science and machine learning in real applications
    - SpaceX's rival uses SpaceX Falcon 9 rocket data to determine the first stage landing success rate and the cost of a launch
- Problems you want to find answers
    - How is the first stage landing success rate of SpaceX Falcon 9 rocket
    - How much is the cost of a launch (SpaceX: 62 million, Other companies: ~165 million)

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Require the data from SpaceX API
  - Collect the data from a Wikipedia page
- Perform data wrangling
  - Clean the data for visualization and information extraction for predictive models
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - Test different classification models over the data set and find the optimal solution

Data Collection

Major steps for data collection and visualization:

Step 1: collect data from SpaceX API and convert to json files → Step 2: scrap and filter data to include Falcon 9 data, assign data to data frame and dictionary formats, and export to a csv file → Step 3: plot and visualize the data

File example:

FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block	ReusedCount
1	2010-06-04	Falcon 9	6123.547647058824	LEO	CCSFS SLC 40	None None	1	False	False	False		1.0	0
2	2012-05-22	Falcon 9	525.0	LEO	CCSFS SLC 40	None None	1	False	False	False		1.0	0
3	2013-03-01	Falcon 9	677.0	ISS	CCSFS SLC 40	None None	1	False	False	False		1.0	0
4	2013-09-29	Falcon 9	500.0	PO	VAFB SLC 4E	False Ocean	1	False	False	False		1.0	0
5	2013-12-03	Falcon 9	3170.0	GTO	CCSFS SLC 40	None None	1	False	False	False		1.0	0
6	2014-01-06	Falcon 9	3325.0	GTO	CCSFS SLC 40	None None	1	False	False	False		1.0	0
7	2014-04-18	Falcon 9	2296.0	ISS	CCSFS SLC 40	True Ocean	1	False	False	True		1.0	0
8	2014-07-14	Falcon 9	1316.0	LEO	CCSFS SLC 40	True Ocean	1	False	False	True		1.0	0
9	2014-08-05	Falcon 9	4535.0	GTO	CCSFS SLC 40	None None	1	False	False	False		1.0	0
10	2014-09-07	Falcon 9	4428.0	GTO	CCSFS SLC 40	None None	1	False	False	False		1.0	0
11	2014-09-21	Falcon 9	2216.0	ISS	CCSFS SLC 40	False Ocean	1	False	False	False		1.0	0
12	2015-01-10	Falcon 9	2395.0	ISS	CCSFS SLC 40	False ASDS	1	True	False	True	5e9e3032383ecb761634e7cb	1.0	0

https://github.com/sizhe1987/IBM-Data-Science-Course/blob/main/Capstone%20project/1.%20dataset_part_1.csv

Data Collection – SpaceX API

Request response from SpaceX API using get request and convert data to json file

↓

Use custom functions to clean data

↓

Clean data and assign data to data frame and dictionary formats

↓

Filter data to include only Falcon 9 launches and export to a csv file

Task 1: Request and parse the SpaceX launch data using the GET request

To make the requested JSON results more consistent, we will use the following static response object for this project:

```
[9]: 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

```
[10]: response=requests.get(static_json_url)
```

```
[11]: response.status_code
```

```
[11]: 200
```

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

```
[12]: # 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

```
[13]: # Get the head of the dataframe
data.head(5)
```

...

You will notice that a lot of the data are IDs. For example the rocket column has no information about the rocket just an identification number.

We will now use the API again to get information about the launches using the IDs given for each launch. Specifically we will be using columns `rocket`, `payloads`, `launchpad`, and `cores`.

```
[14]: # Lets take a subset of our dataframe keeping only the features we want and the flight_number, and date_utc.
data = data[['rocket', 'payloads', 'launchpad', 'cores', 'flight_number', 'date_utc']]

# We will remove rows with multiple cores because those are falcon rockets with 2 extra rocket boosters and rows that have multiple payloads in a single rocket.
data = data[data['cores'].map(len)==1]
data = data[data['payloads'].map(len)==1]

# Since payloads and cores are lists of size 1 we will also extract the single value in the list and replace the feature.
data['cores'] = data['cores'].map(lambda x : x[0])
data['payloads'] = data['payloads'].map(lambda x : x[0])

# We also want to convert the date_utc to a datetime datatype and then extracting the date leaving the time
data['date'] = pd.to_datetime(data['date_utc']).dt.date

# Using the date we will restrict the dates of the launches
data = data[data['date'] <= datetime.date(2020, 11, 13)]
```

- From the `rocket` we would like to learn the booster name

https://github.com/sizhe1987/IBM-Data-Science-Course/blob/main/Capstone%20project/1.%20Spacex-data-collection-api.ipynb

Data Collection - Scraping

Perform http get to request Falcon 9 html page and create Beautiful Soup object

↓

Extract all column names from the html table header

↓

Create a data frame by parsing the launch html tables

↓

Export data into a csv file

TASK 1: Request the Falcon9 Launch Wiki page from its URL

First, let's perform an HTTP GET method to request the Falcon9 Launch HTML page, as an HTTP response.

```
[6]: # use requests.get() method with the provided static_url
     # assign the response to a object
     response = requests.get(static_url).text
```

Create a `BeautifulSoup` object from the HTML `response`

```
[7]: # Use BeautifulSoup() to create a BeautifulSoup object from a response text content
     soup = BeautifulSoup(response, 'html.parser')
```

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

```
[8]: # 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

```
[9]: # 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")
     print(html_tables)

•••
```

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

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

•••
```

You should able to see the columns names embedded in the table header elements `<th>` as follows:

https://github.com/sizhe1987/IBM-Data-Science-Course/blob/main/Capstone%20project/2.%20Spacex_webscraping.ipynb

Data Wrangling

Load data from csv file and calculate the number of launches on each site

Calculate the number and the occurrence of each orbit

Calculate the number and occurrence of mission outcome of the orbits

Create a landing outcome label from outcome column and export to a csv file

TASK 1: Calculate the number of launches on each site

The data contains several Space X launch facilities: Cape Canaveral Space Launch Complex 40 VA
.The location of each Launch Is placed in the column `LaunchSite`

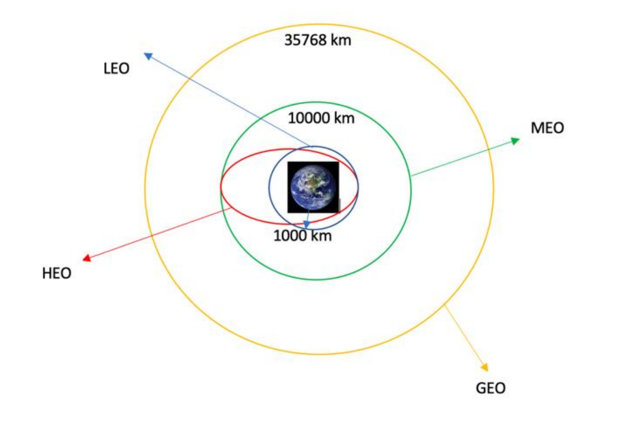
Next, let's see the number of launches for each site.

Use the method `value_counts()` on the column `LaunchSite` to determine the number of lau

[6]:
```
# Apply value_counts() on column LaunchSite
launches = df['LaunchSite'].value_counts()
launches
```

[6]:
```
LaunchSite
CCAFS SLC 40    55
KSC LC 39A      22
VAFB SLC 4E     13
Name: count, dtype: int64
```

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



[14]:
```
df.head(5)
```

[14]:

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block	ReusedCount	Serial	Longitude	Latitude	Class
0	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
1	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
2	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
3	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
4	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

We can use the following line of code to determine the success rate:

[15]:
```
df["Class"].mean()
```

[15]:
```
np.float64(0.6666666666666666)
```

https://github.com/sizhe1987/IBM-Data-Science-Course/blob/main/Capstone%20project/3.%20Spacex-Data%20wrangling.ipynb

EDA with Data Visualization

- Summarize what charts were plotted and why you used those charts

Visualize the relationship between flight number and launch site → Visualize the relationship between payload and launch site → Visualize the relationship between success rate of each orbit type → Visualize the relationship between flight number and orbit type → Visualize the relationship between payload and orbit type → Visualize the launch success yearly trend → Create dummy variables to categorical columns → Cast all numerical columns to float64 datatype

https://github.com/sizhe1987/IBM-Data-Science-Course/blob/main/Capstone%20project/5.%20Eda-dataviz.ipynb

EDA with SQL

- The workflow of EDA with SQL
  - Display the names of the unique launch sites in the space mission
  - Display 5 records where launch sites begin with the string ‘CCA’
  - Display average payload mass carried by booster launched by NASA (CRS)
  - Display average payload mass carried by booster version F9 v1.1
  - List the date when the first successful landing outcome in ground pad was achieved
  - List the names of the boosters which have success in drone ship and mass > 4000 & <6000
  - List the total number of successful and failure mission outcomes
  - List the names of the booster which have carried the max payload mass
  - List the records which display the month, failure landing, booster version, etc.
  - Rank the count of landing outcomes or success

https://github.com/sizhe1987/IBM-Data-Science-Course/blob/main/Capstone%20project/4.%20Eda-sql-coursera_sqllite.ipynb

Build an Interactive Map with Folium

Mark all launch sites on a map created using Folium by adding markers with circle, popup label and text label to each site using tis longitude and latitude coordinates to show the geographical location approximately to the equator

→ Mark the success/failed launches for each site on the map using coloured markers

→ Calculate the distance between a launch site to its proximities

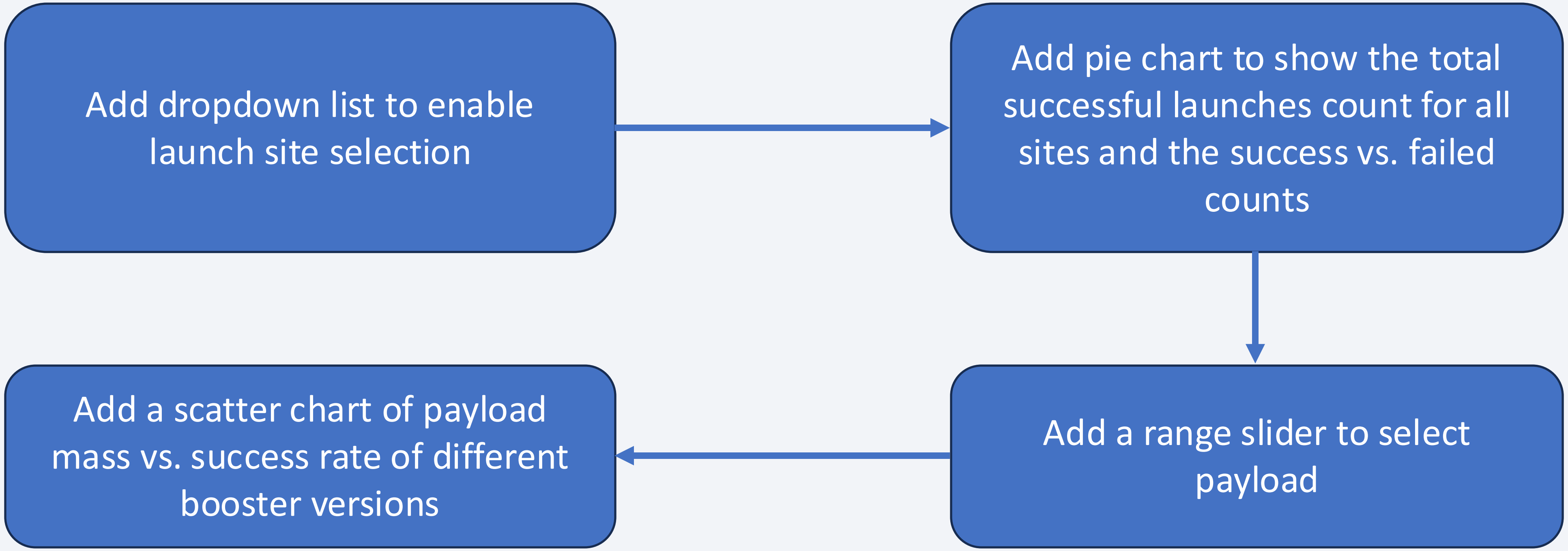
*** Explanation:**

From the visulisation analysis of the launch site, we can clearly see that:

- Relative close to railway (15.23 km)
- Relative close to highway (20.28 km)
- Relative close to coastline (14.99 km)
- Also the launch site KSC LC-39A is relative close to its closet city Titusville (16.32 km)
- Failed rocket with its high speed can cover distances like 15-20 km in few seconds. It could be potentially dangerous to populated areas.

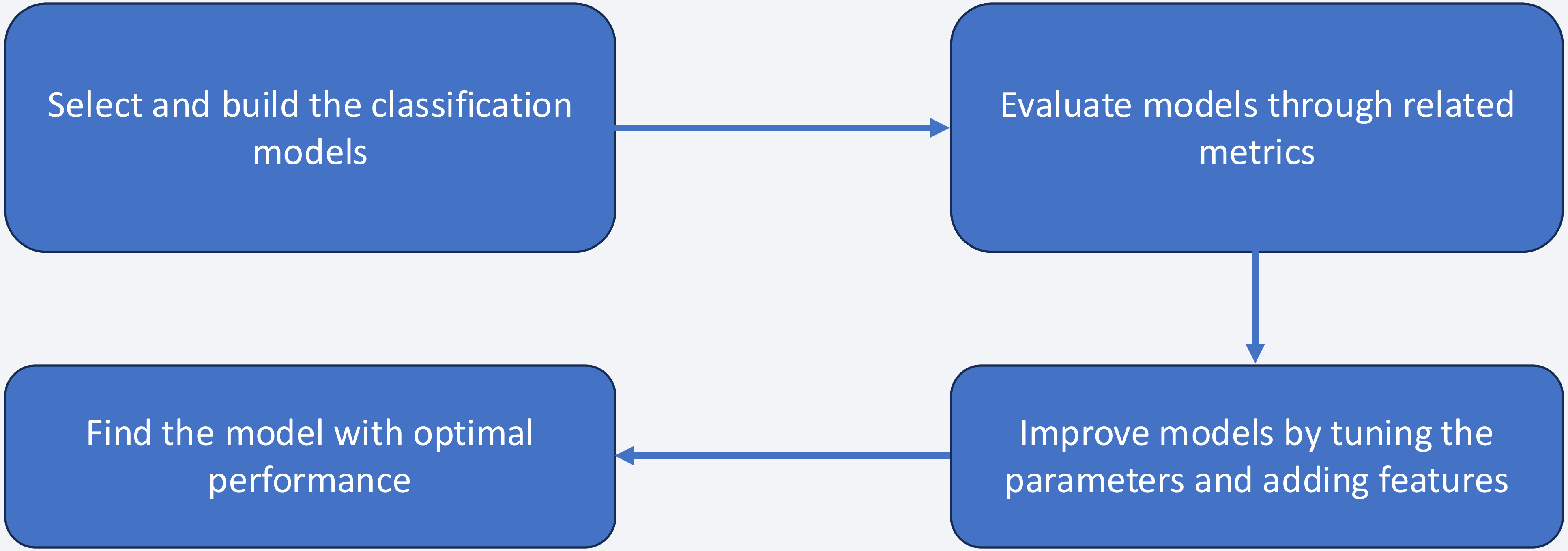
https://github.com/sizhe1987/IBM-Data-Science-Course/blob/main/Capstone%20project/6.%20launch_site_location.ipynb

Build a Dashboard with Plotly Dash



https://github.com/sizhe1987/IBM-Data-Science-Course/blob/main/Capstone%20project/7.%20spacex_dash_app.py

Predictive Analysis (Classification)



https://github.com/sizhe1987/IBM-Data-Science-Course/blob/main/Capstone%20project/8.%20SpaceX_Machine%20Learning%20Prediction_Part_5.ipynb

Predictive Analysis (Classification): Details

1. Load the Pandas data frame
2. Create a Numpy array from the column class in the data
3. Standardise the data in X (i.e., features) and assign to variable Y (i.e., target)
4. Use train_test_split function to split the dataset into training and test data subset
5. Create a logistic regression object and a grid search cross validation object, then train the model on the training data
6. Calculate the accuracy on the test data using multiple score methods for the logistic regression model
7. Create a SVM object and a grid search cross validation object, then train the model on the training data
8. Calculate the accuracy on the test data using multiple score methods for the SVM model
9. Create a decision tree object and a grid search cross validation object, then train the model on the training data
10. Calculate the accuracy on the test data using multiple score methods for the decision tree model
11. Create a K-NN object and a grid search cross validation object, then train the model on the training data
12. Calculate the accuracy on the test data using multiple score methods for the K-NN model
13. Compare results from different models and find the optimal solution

Results

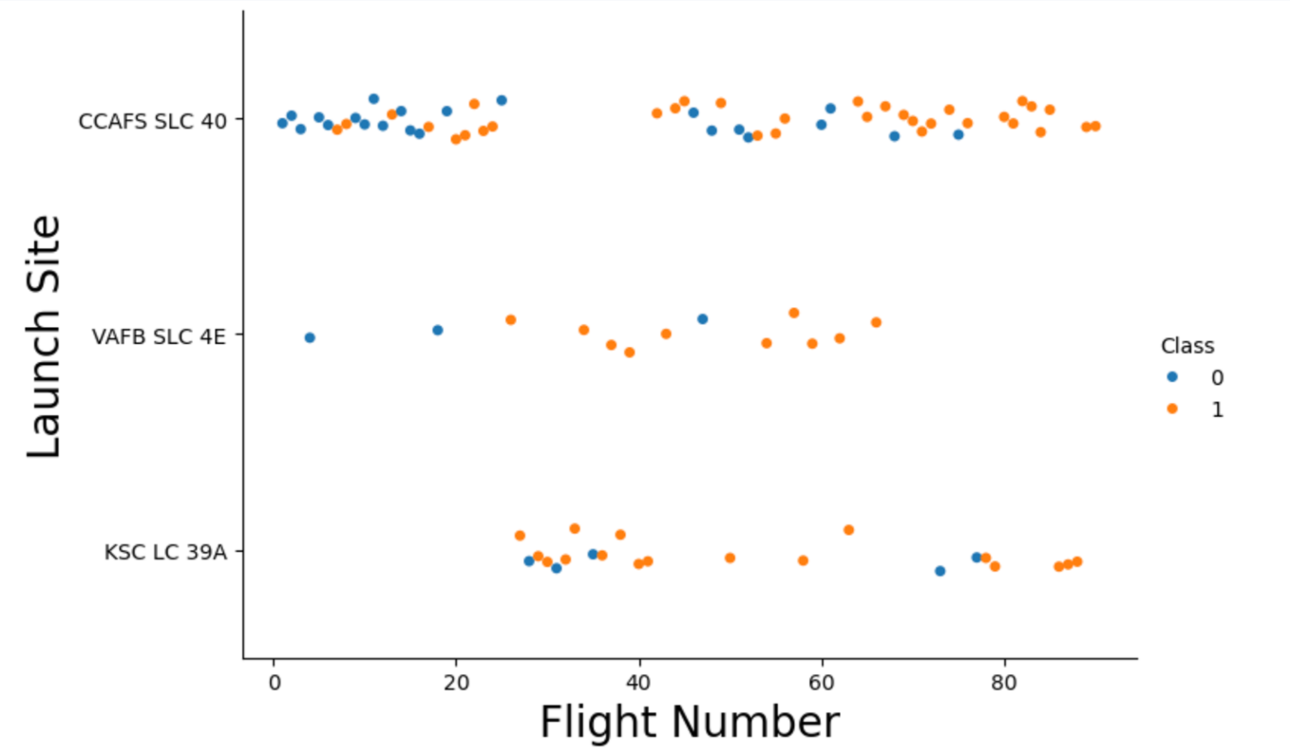
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results

Section 2

Insights drawn from EDA

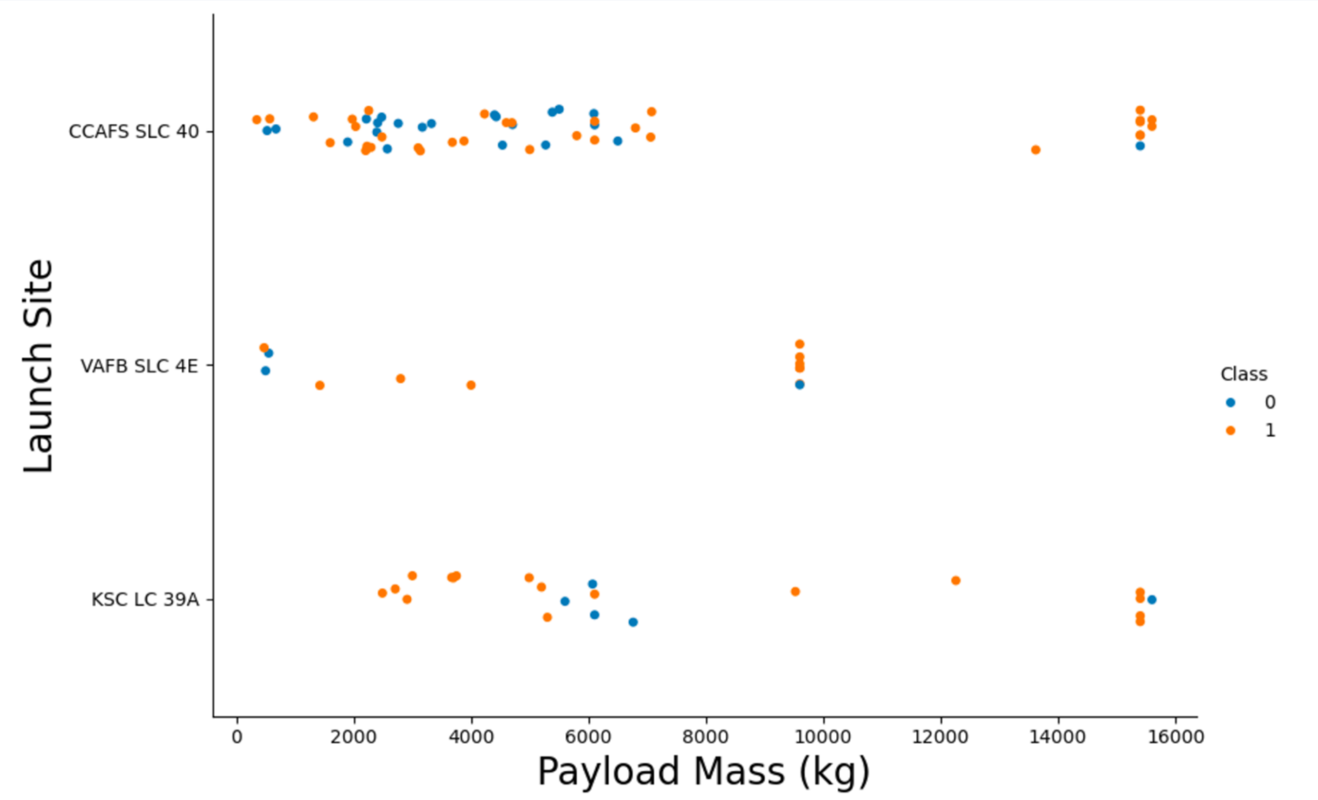
Flight Number vs. Launch Site

- The majority of the flights were launched from CCAFS SLC 40 sites.
- The VAFB SLC 4E sites have a higher success rate than other sites.



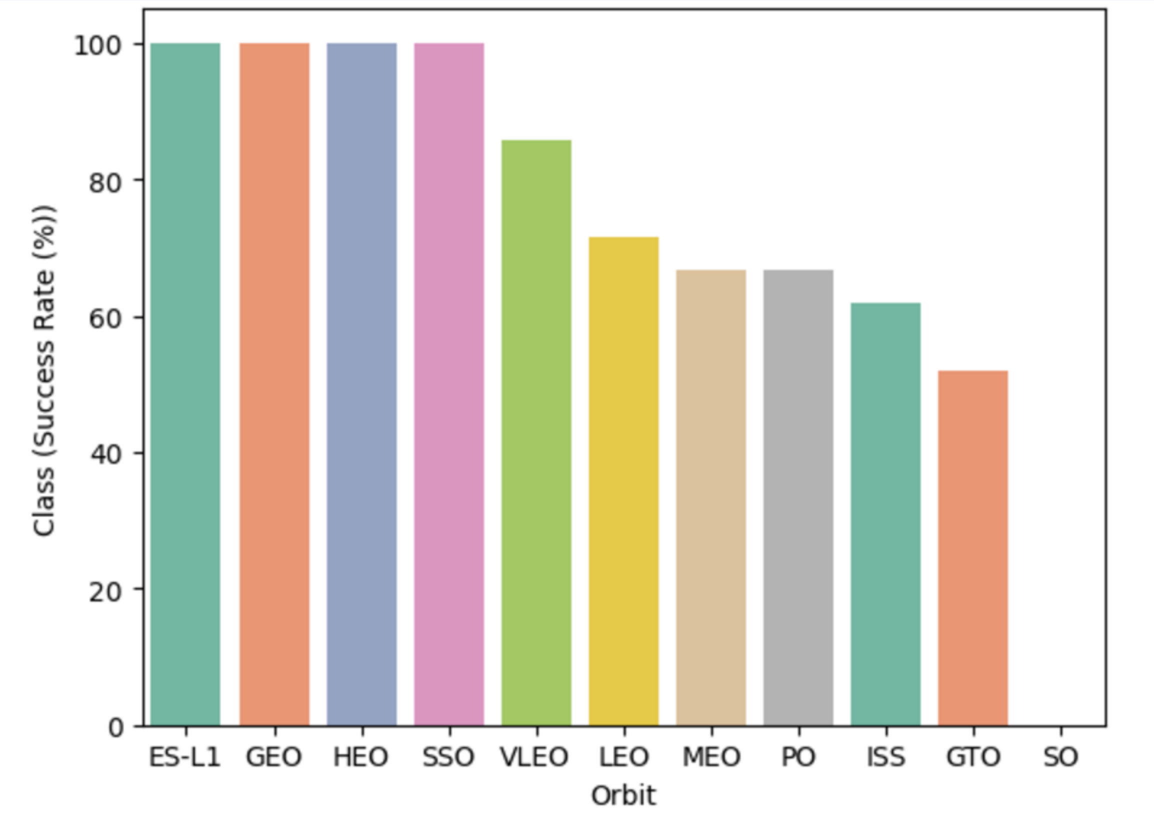
Payload vs. Launch Site

- The majority of the flights with payload greater than 7000kg were launched successfully.
- Flights with payload less than 5500kg were 100% launched from KSC LC 39A sites.



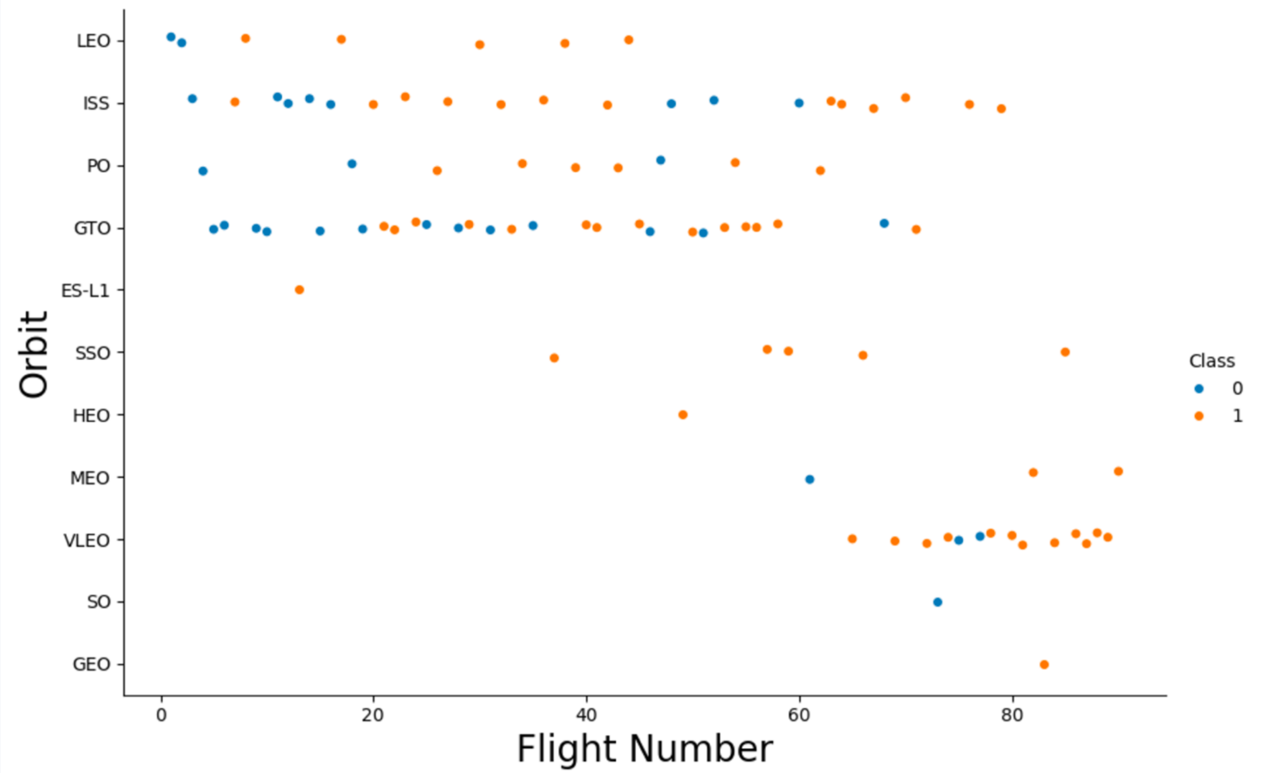
Success Rate vs. Orbit Type

- ES-L1, GEO, HEO and SSO orbits have 100% success rate.
- In comparison, SO orbit has 0% success rate.



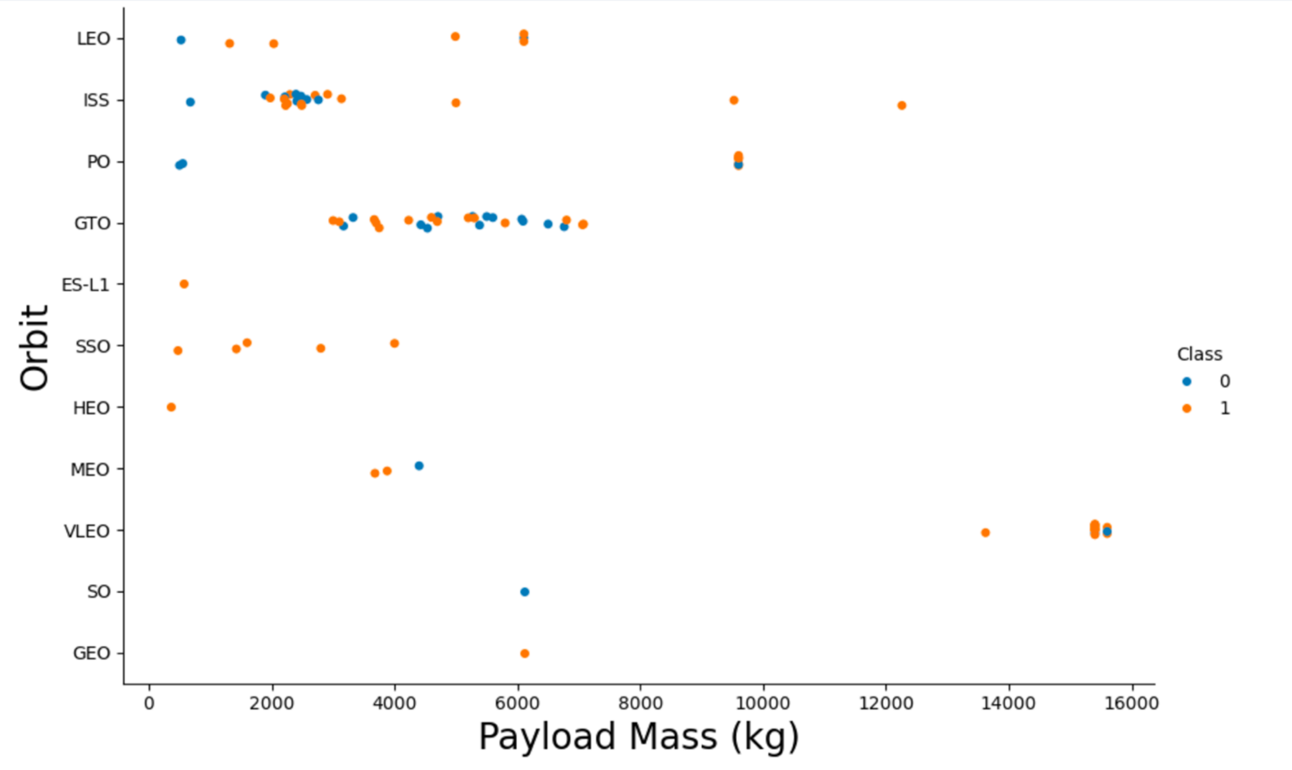
Flight Number vs. Orbit Type

- The majority of the flights were launched to ISS and GTO orbits.
- Although ES-L1 and GEO orbits have 100% success rate, only 1 flight was launched to them, respectively.



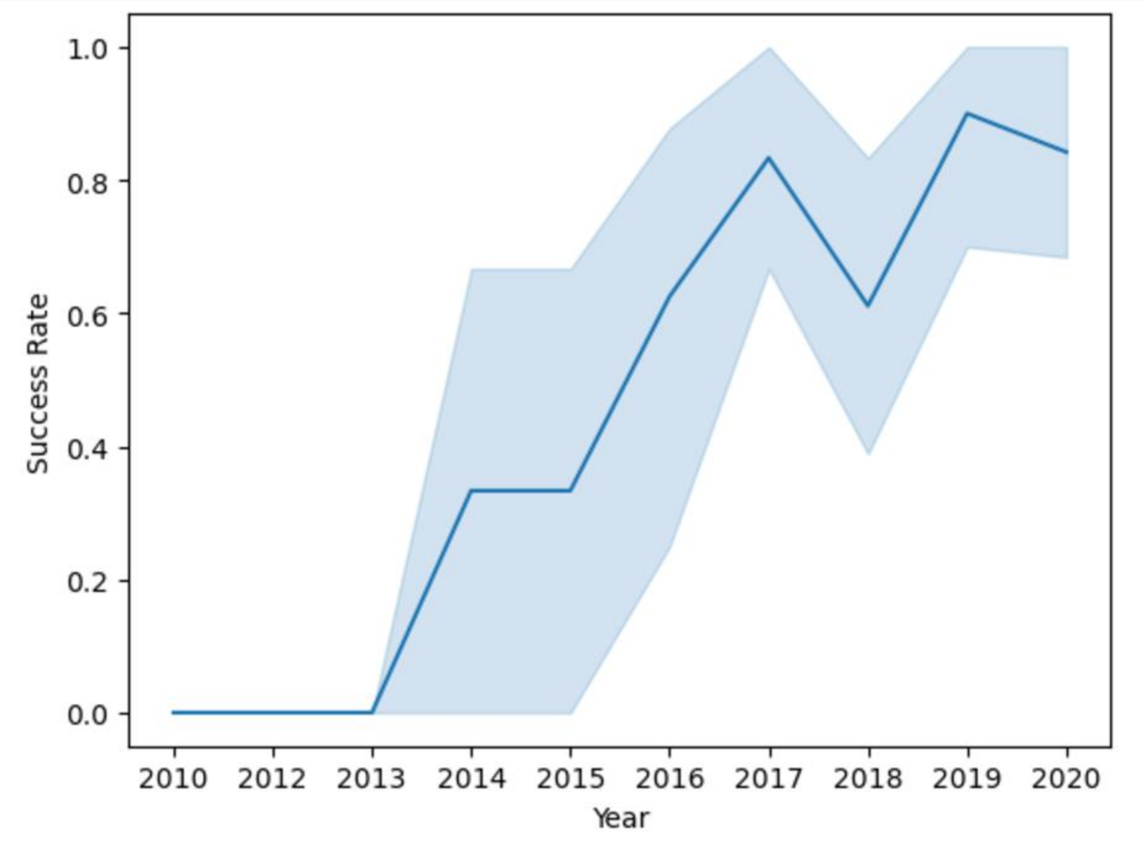
Payload vs. Orbit Type

- Payload above 10,000kg were placed in PO, ISS and LEO orbits.
- Payload above 4000kg and less than 8000kg were placed in GTO orbit.



Launch Success Yearly Trend

- The long-term trend of success rate increases steadily since 2013.
- There was a drop in the year 2018.



All Launch Site Names

- We used ‘Distinct’ to get the unique names of launch sites from the column ‘LAUNCH_SITE’

Task 1

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

```
[8]: %sql SELECT DISTINCT LAUNCH_SITE as "Launch_Sites" FROM SPACEXTBL;

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

[8]:

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

Launch Site Names Begin with 'CCA'

- We used ‘Like’ to retrieve the data related to launch sites with names begin with ‘CCA’.
- We also used ‘Limit’ to retrieve top data in an ascending order.

Task 2

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

```
[9]: %sql SELECT * FROM 'SPACEXTBL' WHERE Launch_Site LIKE 'CCA%' LIMIT 5;
```

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

[9]:

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PA
2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	
2010-12-08	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of Brouere cheese	
2012-05-22	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	
2012-10-08	0:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	
2013-03-01	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	

Total Payload Mass

- We used ‘Sum’ to retrieve the total payload mass of rockets for NASA.

```
[10]: %sql SELECT SUM(PAYLOAD_MASS__KG_) as "Total Payload Mass(Kgs)", Customer FROM 'SPACEXTBL' WHERE Customer = 'NASA (CRS)';

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

[10]:

Total Payload Mass(Kgs)	Customer
45596	NASA (CRS)

Average Payload Mass by F9 v1.1

- We used 'Avg' to retrieve the average payload mass of rocket booster version is F9 V1.1.

```
[11]: %sql SELECT AVG(PAYLOAD_MASS__KG_) as "Payload Mass Kgs", Customer, Booster_Version FROM 'SPACEXTBL' WHERE Booster_Version LIKE 'F9 v1.1%';

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

[11]:

Payload Mass Kgs	Customer	Booster_Version
2534.6666666666665	MDA	F9 v1.1 B1003

First Successful Ground Landing Date

- We used ‘Min’ to retrieve the earliest launch landing date from the database.

```
[14]: %sql SELECT MIN(DATE) FROM 'SPACEXTBL' WHERE "Landing_Outcome" = "Success (ground pad)";

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

[14]:

MIN(DATE)
2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

- We used multiple key words to retrieve the earliest launch landing date from the database.
- ‘Distinct’ helps us to retrieve unique records, while ‘Where’ helps to filter out corresponding records meet our specific requirements.

```
[16]: %%sql SELECT DISTINCT Booster_Version, Payload FROM SPACEXTBL
          WHERE "Landing_Outcome" = "Success (drone ship)" AND PAYLOAD_MASS__KG_ > 4000 AND PAYLOAD_MASS__KG_ < 6000;

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

[16]:

Booster_Version	Payload
F9 FT B1022	JCSAT-14
F9 FT B1026	JCSAT-16
F9 FT B1021.2	SES-10
F9 FT B1031.2	SES-11 / EchoStar 105

Total Number of Successful and Failure Mission Outcomes

- We used aggregated function 'group by' to retrieve the number of mission outcomes for every scenario.

```
[18]: %%sql SELECT "Mission_Outcome", COUNT("Mission_Outcome") as Total
          FROM SPACEXTBL GROUP BY "Mission_Outcome";

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

[18]:

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

Boosters Carried Maximum Payload

- We combined and integrated ‘Select’ and ‘Max’ in ‘Where’ clause to get the corresponding records.

```
[19]: %%sql SELECT "Booster_Version",Payload, "PAYLOAD_MASS__KG_" FROM SPACEXTBL
          WHERE "PAYLOAD_MASS__KG_" = (SELECT MAX("PAYLOAD_MASS__KG_") FROM SPACEXTBL);

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

[19]:

Booster_Version	Payload	PAYLOAD_MASS__KG_
F9 B5 B1048.4	Starlink 1 v1.0, SpaceX CRS-19	15600
F9 B5 B1049.4	Starlink 2 v1.0, Crew Dragon in-flight abort test	15600
F9 B5 B1051.3	Starlink 3 v1.0, Starlink 4 v1.0	15600
F9 B5 B1056.4	Starlink 4 v1.0, SpaceX CRS-20	15600
F9 B5 B1048.5	Starlink 5 v1.0, Starlink 6 v1.0	15600
F9 B5 B1051.4	Starlink 6 v1.0, Crew Dragon Demo-2	15600
F9 B5 B1049.5	Starlink 7 v1.0, Starlink 8 v1.0	15600
F9 B5 B1060.2	Starlink 11 v1.0, Starlink 12 v1.0	15600
F9 B5 B1058.3	Starlink 12 v1.0, Starlink 13 v1.0	15600
F9 B5 B1051.6	Starlink 13 v1.0, Starlink 14 v1.0	15600
F9 B5 B1060.3	Starlink 14 v1.0, GPS III-04	15600
F9 B5 B1049.7	Starlink 15 v1.0, SpaceX CRS-21	15600

2015 Launch Records

- The records we retrieved here satisfied our specific requirements – failed landing on drone ship and occurred in year 2015.

```
[20]: %%sql SELECT substr(Date,0,5) as 'year', substr(Date, 6, 2) as 'month',
          "Booster_Version", "Launch_Site", Payload, "PAYLOAD_MASS__KG_", "Mission_Outcome", "Landing_Outcome"
          FROM SPACEXTBL
          WHERE substr(Date,0,5)='2015' AND "Landing_Outcome" = 'Failure (drone ship)';

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

[20]:

year	month	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Mission_Outcome	Landing_Outcome
2015	01	F9 v1.1 B1012	CCAFS LC-40	SpaceX CRS-5	2395	Success	Failure (drone ship)
2015	04	F9 v1.1 B1015	CCAFS LC-40	SpaceX CRS-6	1898	Success	Failure (drone ship)

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

- We rank the success mission outcomes between 2010-06-04 and 2017-03-20 in a descending order .

```
[21]: %%sql SELECT * FROM SPACEXTBL
          WHERE "Landing_Outcome" LIKE 'Success%' AND (Date BETWEEN '2010-06-04' AND '2017-03-20')
          ORDER BY Date DESC;

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

[21]:

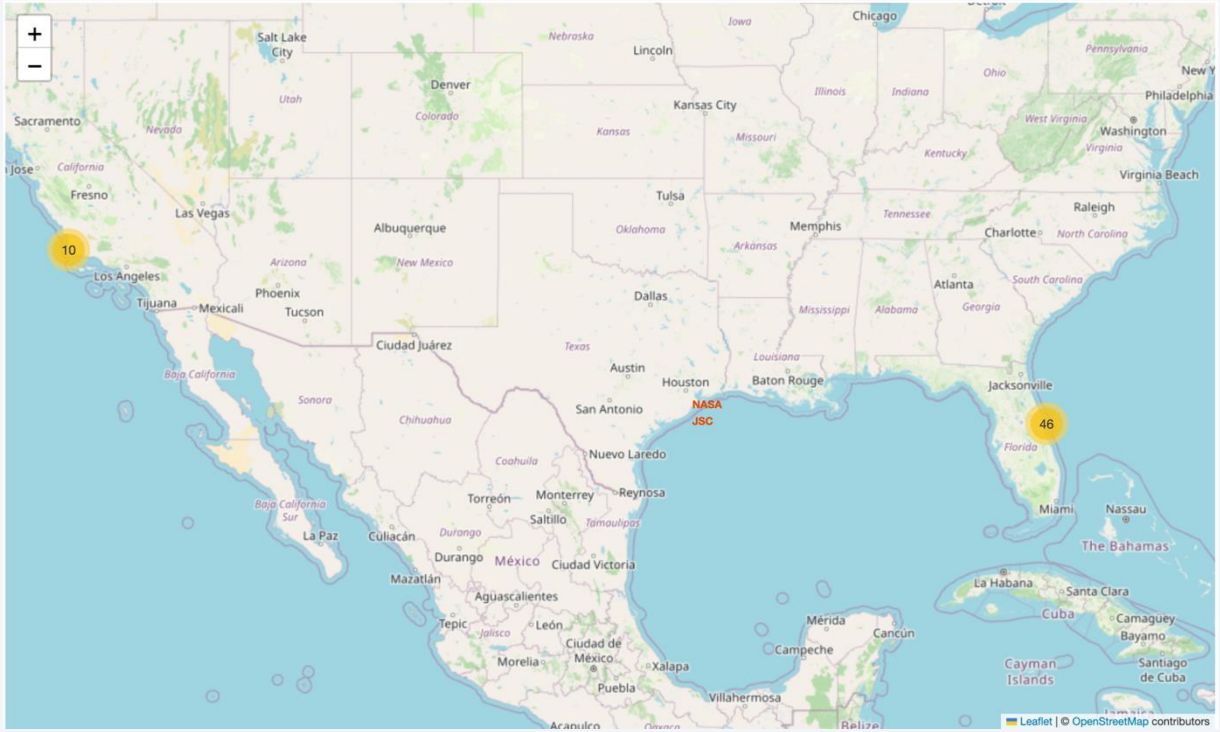
Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing_Outcome
2017-02-19	14:39:00	F9 FT B1031.1	KSC LC-39A	SpaceX CRS-10	2490	LEO (ISS)	NASA (CRS)	Success	Success (ground pad)
2017-01-14	17:54:00	F9 FT B1029.1	VAFB SLC-4E	Iridium NEXT 1	9600	Polar LEO	Iridium Communications	Success	Success (drone ship)
2016-08-14	5:26:00	F9 FT B1026	CCAFS LC-40	JCSAT-16	4600	GTO	SKY Perfect JSAT Group	Success	Success (drone ship)
2016-07-18	4:45:00	F9 FT B1025.1	CCAFS LC-40	SpaceX CRS-9	2257	LEO (ISS)	NASA (CRS)	Success	Success (ground pad)
2016-05-27	21:39:00	F9 FT B1023.1	CCAFS LC-40	Thaicom 8	3100	GTO	Thaicom	Success	Success (drone ship)
2016-05-06	5:21:00	F9 FT B1022	CCAFS LC-40	JCSAT-14	4696	GTO	SKY Perfect JSAT Group	Success	Success (drone ship)
2016-04-08	20:43:00	F9 FT B1021.1	CCAFS LC-40	SpaceX CRS-8	3136	LEO (ISS)	NASA (CRS)	Success	Success (drone ship)
2015-12-22	1:29:00	F9 FT B1019	CCAFS LC-40	OG2 Mission 2 11 Orbcomm-OG2 satellites	2034	LEO	Orbcomm	Success	Success (ground pad)

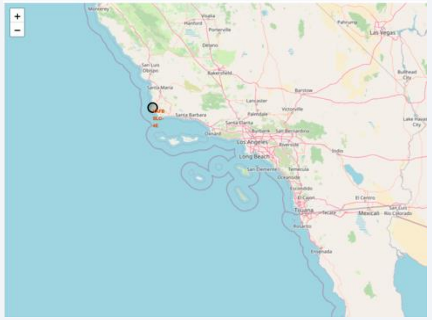
Section 3

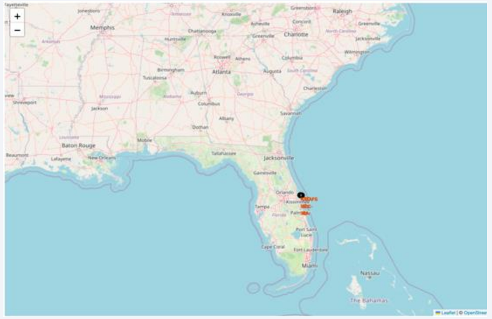
Launch Sites Proximities Analysis

US Launch sites' location in CA and FL

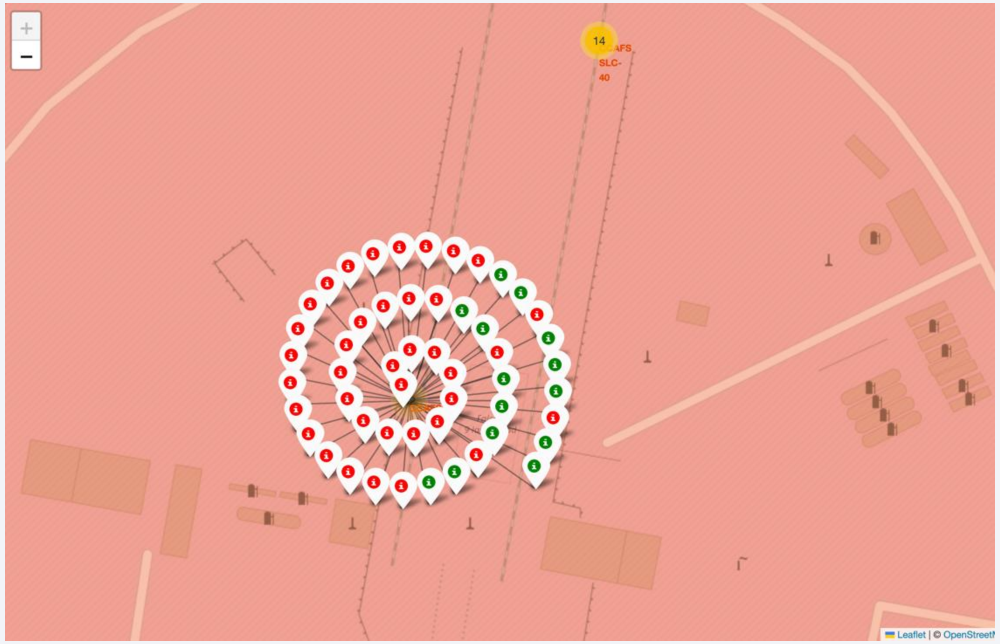
All launch sites considered in this project are in very close proximity to the coast While starting rockets towards the ocean we minimize the risk of having any debris dropping or exploding near people.

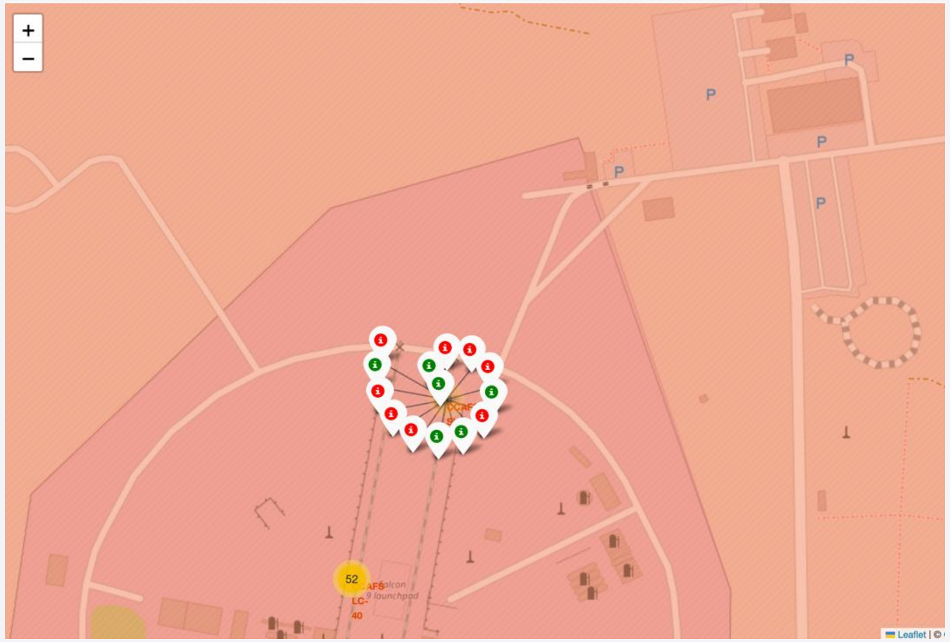






Color Labels Showing the Launch Sites on a Map

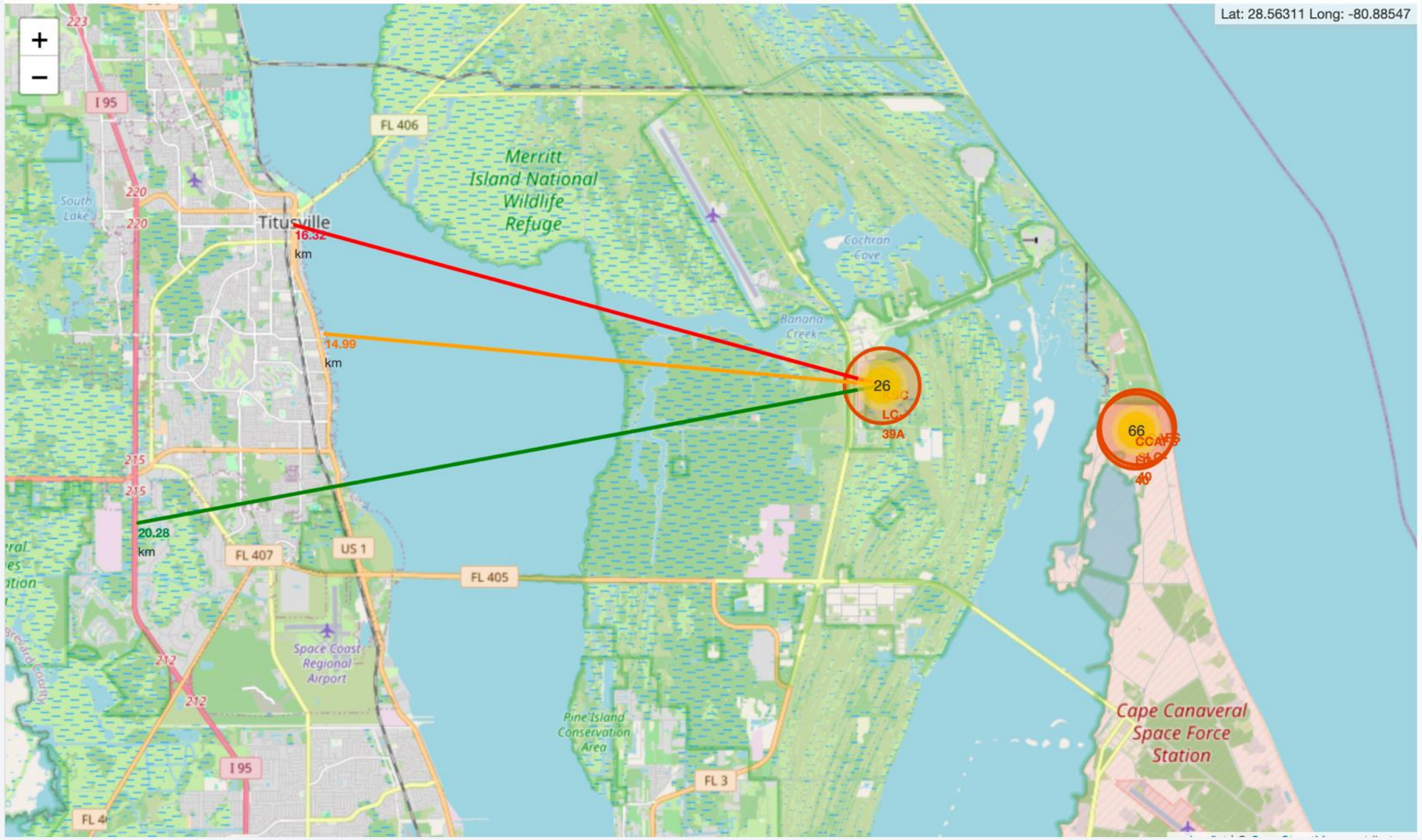




Green: successful launch
Red: failed launch

Safe Distance to Launch Site

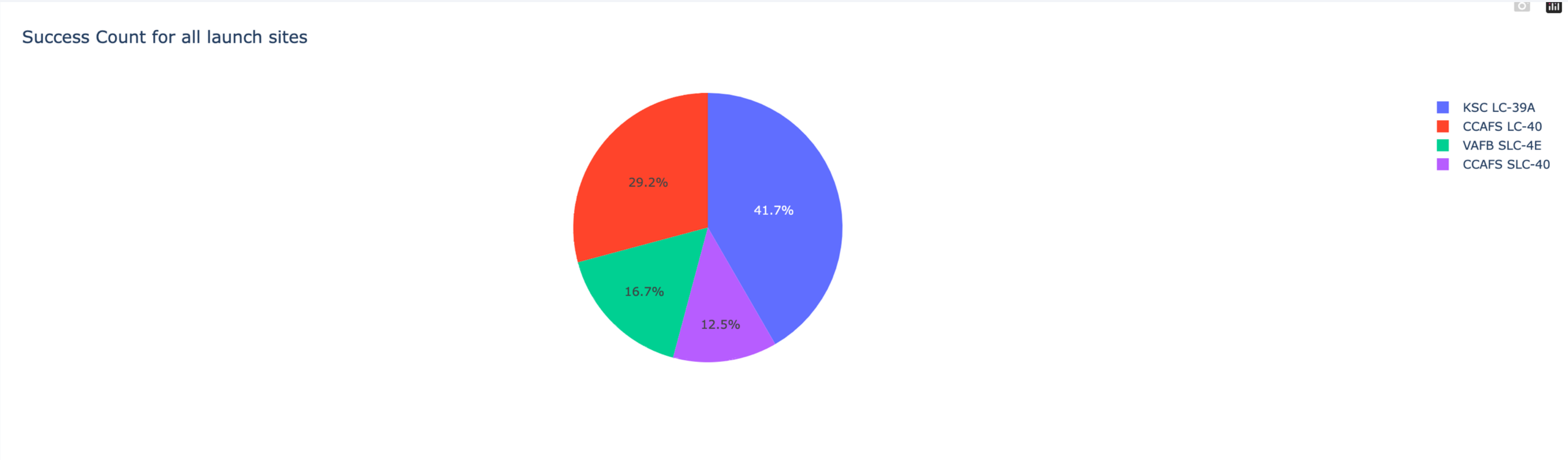
The obtained results indicate that all launch sites are at safe distance from highway Lines, coastline and cities.



Section 4

Build a Dashboard with Plotly Dash

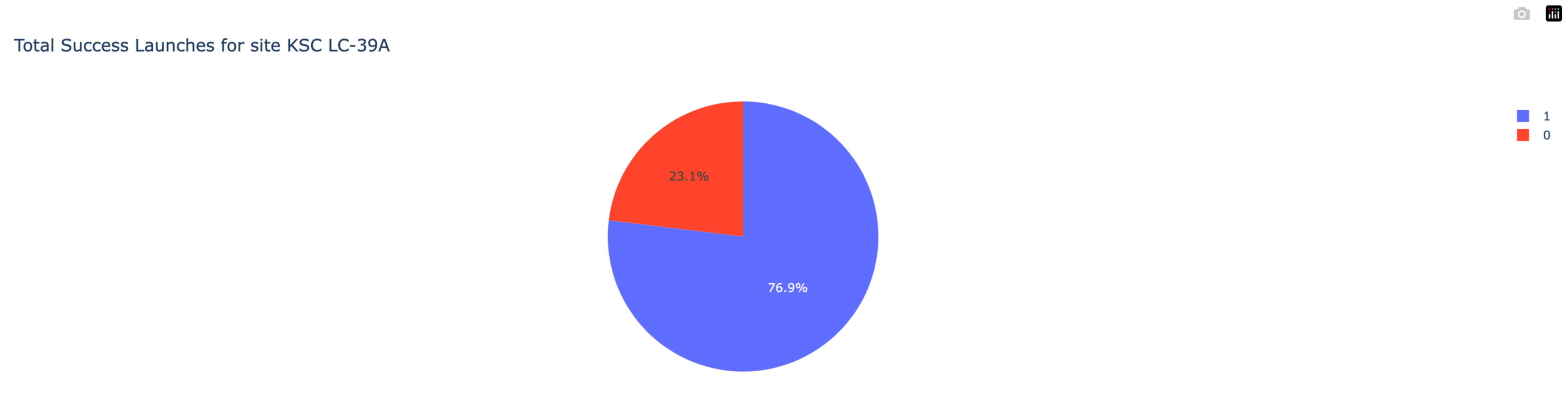
Total Launch Success for All Sites



The highest two success rates for all sites are:

1. KSC LC-39A: 41.7%
2. CCAFS LC-40: 29.2%

<Dashboard Screenshot 2>

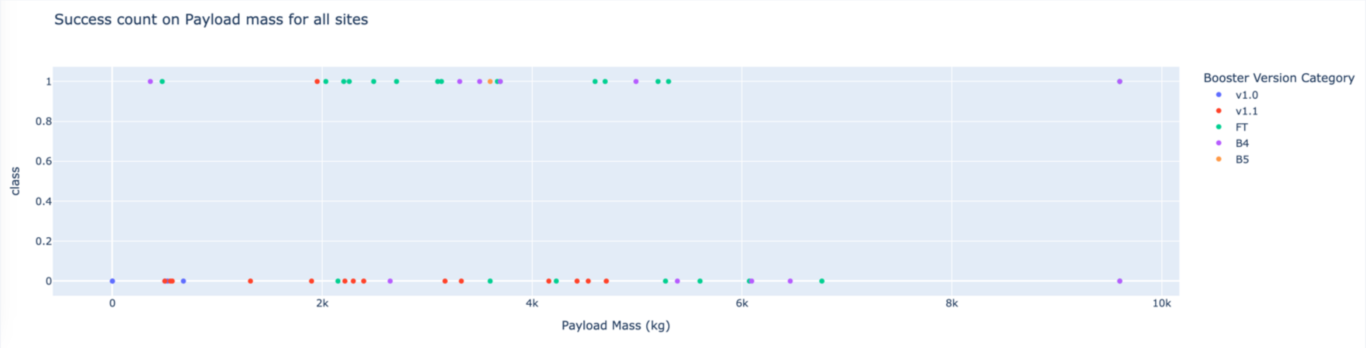


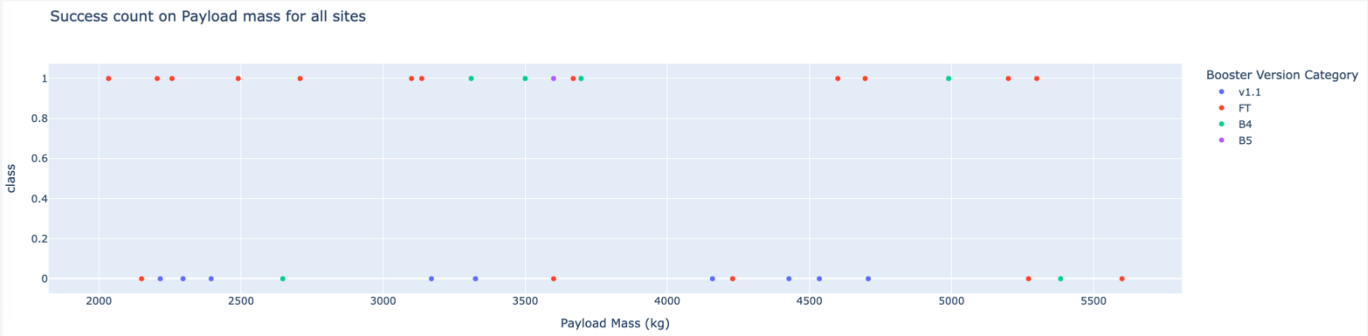
Site KSC LC-39 success rate is 76.9%

Payload vs. Launch Outcome for All Sites

Highest success rate for payload is between 2000 and 6000 Kg

Up: all payload
Down: payload between 2000 and 6000kg





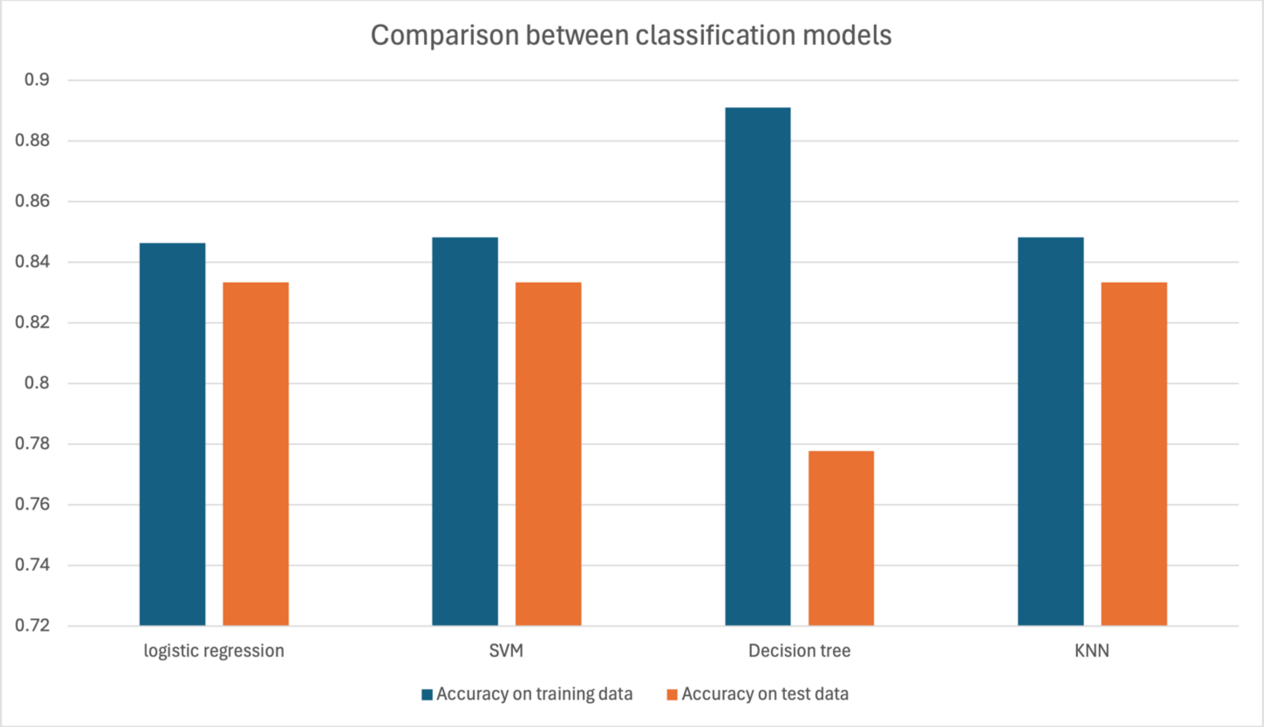
Section 5

Predictive Analysis (Classification)

Classification Accuracy

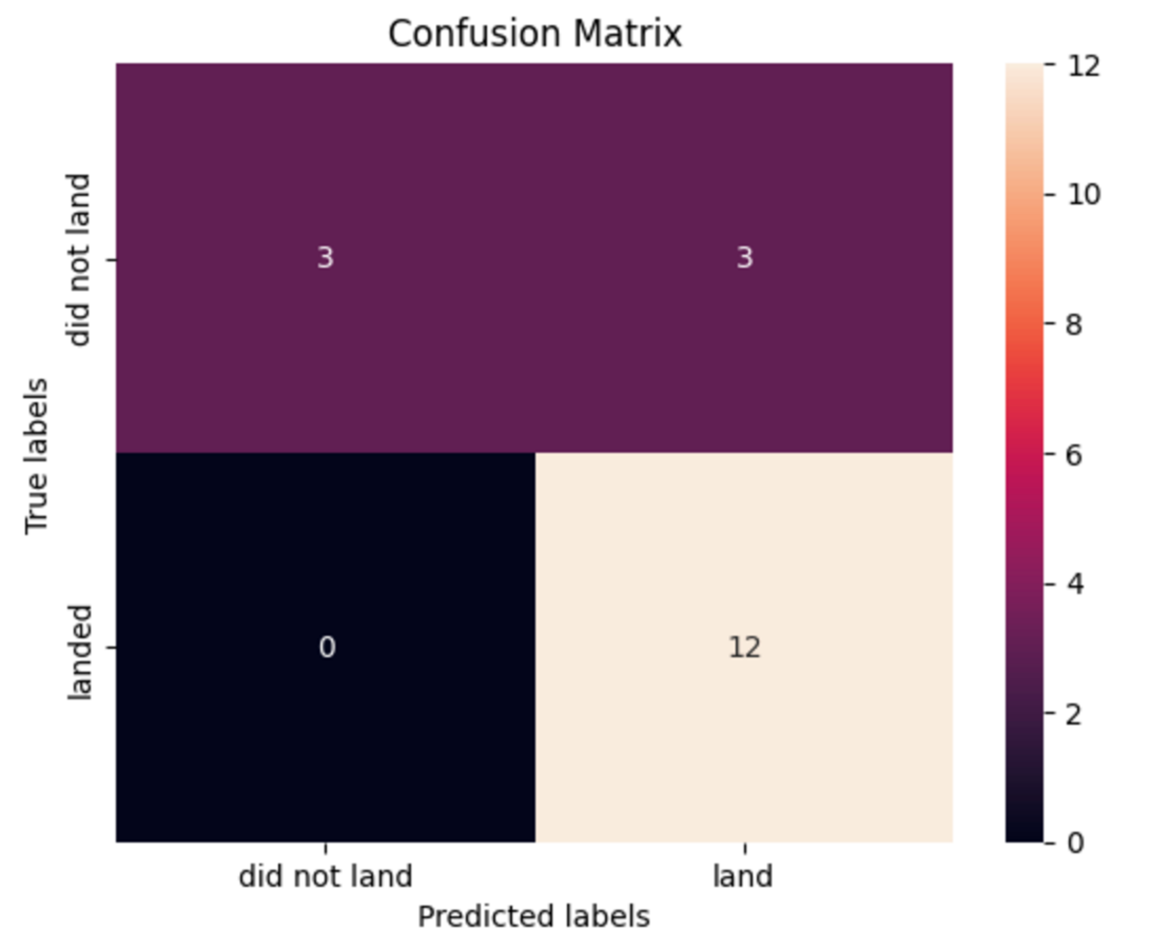
- Logistic regression, SVM and KNN perform similar in training and test data.

- Decision tree model performs much better than other models in the training data, but much worse in the test data, implying an overfitting issue.



Confusion Matrix

- Based on the results of confusion matrix, logistic regression model performs the best.
- 12 true positive and 3 true negative predictions over 18 predicted values, implying an 83.33% of accuracy.



Conclusions

- The success rate for the rocket launches increased after 2013.
- Orbits GEO, HEO, ES-L1 and SSO have 100% launch success rate.
- Launch site KSC LC-39A has the highest success rate.
- The Decision Tree model is the best ML algorithm for analysing the SpaceX data set and provided the best accuracy results

Appendix

- Github: https://github.com/sizhe1987/IBM-Data-Science-Course/tree/main/Capstone%20project

Thank you!