Winning Space Race with Data Science

Yu, Zhu

2021-09-20

Outline

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

Executive Summary

- Summary of methodologies
  - Data Collection via API, SQL and Web Scraping
  - Data Wrangling
  - EDA with matplotlib and seaborn
  - EDA with SQL
  - Building an interactive map with Folium
  - Building a dashboard with Plotly Dash
  - Predictive analysis – ML for classification

- Summary of all results
  - Exploratory Data Analysis Result
  - Interactive Visualization
  - Best Model for Classification Prediction

Introduction

- ## Project background and context

  According the website of SpaceX, Falcon 9 rocket launches 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.

- ## Problems you want to find answers

  - What factors (variables) can lead to a successful landing?
  - What effects of each variable can have on the landing outcome?
  - What are the best combinations of factors to ensure the best result?

Example of a successful landing

Examples of unsuccessful landings

Section 1

Methodology

Methodology

Executive Summary of Methodologies

- Data collection methodology:
  - SpaceX Rest APIs
  - Web Scraping (Wikipedia)
- Perform data wrangling
  - Dealing with NaN values
  - Feature Selection
  - One Hot Encoding
- Perform exploratory data analysis (EDA) using visualization and SQL

Methodology

Executive Summary

- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - Model Building
  - Model Tuning
  - Model Evaluation
  - Model Selection

Data Collection

Data collection is a process of gathering relevant information for answering the questions of interest. Usually it is the first step for data analysis. Data can be collected in many ways according to the location and method they're stored. For this project, we collect the data by REST APIs and web scraping from Wikipedia page.

Data Collection – SpaceX API

```
spacex_url="https://api.spacexdata.com/v4/launches/past"

response = requests.get(spacex_url)
```

```
# Call getBoosterVersion
getBoosterVersion(data)

# Call getLaunchSite
getLaunchSite(data)

# Call getPayloadData
getPayloadData(data)

# Call getCoreData
getCoreData(data)
```

```
# Hint data['BoosterVersion']!='Falcon 1'
data_falcon9 = data[data.BoosterVersion!='Falcon 1']

data_falcon9.loc[:,'FlightNumber'] = list(range(1, data_falcon9.shape[0]+1))

data_falcon9.to_csv('dataset_part\_1.csv', index=False)
```

Get response from API

Convert response to .json file

Apply customized functions to clean data

Create dataframe based on dict structure

Filtering and export to flat files

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

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

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block	Reuse
4	1	2010-06-04	Falcon 9	NaN	LEO	CCSFS SLC 40	None None	1	False	False	False	None	1.0	
5	2	2012-05-22	Falcon 9	525.0	LEO	CCSFS SLC 40	None None	1	False	False	False	None	1.0	
6	3	2013-03-01	Falcon 9	677.0	ISS	CCSFS SLC 40	None None	1	False	False	False	None	1.0	
7	4	2013-09-29	Falcon 9	500.0	PO	VAFB SLC 4E	False Ocean	1	False	False	False	None	1.0	
8	5	2013-12-03	Falcon 9	3170.0	GTO	CCSFS SLC 40	None None	1	False	False	False	None	1.0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
89	86	2020-09-03	Falcon 9	15600.0	VLEO	KSC LC 39A	True ASDS	2	True	True	True	5e9e3032383ecb6bb234e7ca	5.0	
90	87	2020-10-06	Falcon 9	15600.0	VLEO	KSC LC 39A	True ASDS	3	True	True	True	5e9e3032383ecb6bb234e7ca	5.0	
91	88	2020-10-18	Falcon 9	15600.0	VLEO	KSC LC 39A	True ASDS	6	True	True	True	5e9e3032383ecb6bb234e7ca	5.0	
92	89	2020-10-24	Falcon 9	15600.0	VLEO	CCSFS SLC 40	True ASDS	3	True	True	True	5e9e3033383ecbb9e534e7cc	5.0	

```
#Global variables
BoosterVersion = []
PayloadMass = []
Orbit = []
LaunchSite = []
Outcome = []
Flights = []
GridFins = []
Reused = []
Legs = []
LandingPad = []
Block = []
ReusedCount = []
Serial = []
Longitude = []
Latitude = []
```

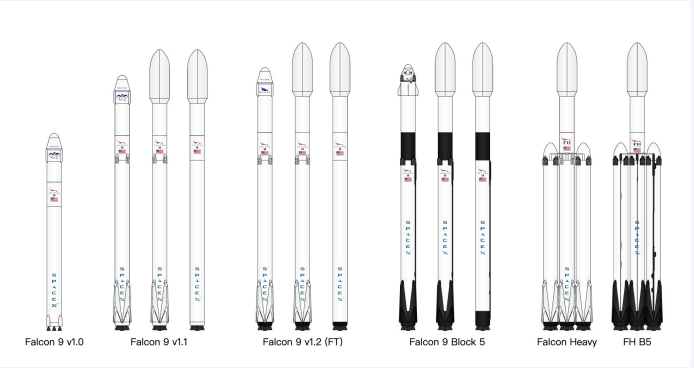
```
launch_dict = {'FlightNumber': list(data['flight_number']),
'Date': list(data['date']),
'BoosterVersion':BoosterVersion,
'PayloadMass':PayloadMass,
'Orbit':Orbit,
'LaunchSite':LaunchSite,
'Outcome':Outcome,
'Flights':Flights,
'GridFins':GridFins,
'Reused':Reused,
'Legs':Legs,
'LandingPad':LandingPad,
'Block':Block,
'ReusedCount':ReusedCount,
'Serial':Serial,
'Longitude': Longitude,
'Latitude': Latitude}

# Create a data from launch_dict
data = pd.DataFrame(launch_dict)
```

Link to Github

Data Collection - Scraping

- Get response from HTML
- Create Beautiful Soup Object
- Find tables
- Get column names
- Create dict-list structure to store data
- Convert dict to dataframe
- Export to flat files (.csv)

Resource: Wikipedia page



2020 [edit]

In late 2019, Gwynne Shotwell stated that SpaceX hoped for as many as 24 launches for Starlink satellites in 2020,[490] in addition to 14 or 15 non-Starlink launches. At 26 launches, 13 of which for Starlink satellites, Falcon 9 had its most prolific year, and Falcon rockets were second most prolific rocket family of 2020, only behind China's Long March rocket family.[491]

[hide] Flight No.	Date and time (UTC)	Version, Booster[b]	Launch site	Payload[c]	Payload mass	Orbit	Customer	Launch outcome	Booster landing
78	7 January 2020, 02:19:21[492]	F9 B5 △ B1049.4	CCAFS, SLC-40	Starlink 2 v1.0 (60 satellites)	15,600 kg (34,400 lb)[5]	LEO	SpaceX	Success	Success (drone ship)
	Third large batch and second operational flight of Starlink constellation. One of the 60 satellites included a test coating to make the satellite less reflective, and thus less likely to interfere with ground-based astronomical observations.[493]								
79	19 January 2020, 15:30[494]	F9 B5 △ B1046.4	KSC, LC-39A	Crew Dragon in-flight abort test[495] (Dragon C205.1)	12,050 kg (26,570 lb)	Sub-orbital[496]	NASA (CTS)[497]	Success	No attempt
	An atmospheric test of the Dragon 2 abort system after Max Q. The capsule fired its SuperDraco engines, reached an apogee of 40 km (25 mi), deployed parachutes after reentry, and splashed down in the ocean 31 km (19 mi) downrange from the launch site. The test was previously slated to be accomplished with the Crew Dragon Demo-1 capsule;[498] but that test article exploded during a ground test of SuperDraco engines on 20 April 2019.[419] The abort test used the capsule originally intended for the first crewed flight.[499] As expected, the booster was destroyed by aerodynamic forces after the capsule aborted.[500] First flight of a Falcon 9 with only one functional stage — the second stage had a mass simulator in place of its engine.								
80	29 January 2020, 14:07[501]	F9 B5 △ B1051.3	CCAFS, SLC-40	Starlink 3 v1.0 (60 satellites)	15,600 kg (34,400 lb)[5]	LEO	SpaceX	Success	Success (drone ship)
	Third operational and fourth large batch of Starlink satellites, deployed in a circular 290 km (180 mi) orbit. One of the fairing halves was caught, while the other was fished out of the ocean.[502]								
81	17 February 2020, 15:05[503]	F9 B5 △ B1056.4	CCAFS, SLC-40	Starlink 4 v1.0 (60 satellites)	15,600 kg (34,400 lb)[5]	LEO	SpaceX	Success	Failure (drone ship)
	Fourth operational and fifth large batch of Starlink satellites. Used a new flight profile which deployed into a 212 km × 386 km (132 mi × 240 mi) elliptical orbit instead of launching into a circular orbit and firing the second stage engine twice. The first stage booster failed to land on the drone ship[504] due to incorrect wind data.[505] This was the first time a flight proven booster failed to land.								
82	7 March 2020, 04:50[506]	F9 B5 △ B1059.2	CCAFS, SLC-40	SpaceX CRS-20 (Dragon C112.3 △)	1,977 kg (4,359 lb)[507]	LEO (ISS)	NASA (CRS)	Success	Success (ground pad)
	Last launch of phase 1 of the CRS contract. Carries *Bartolomeo*, an ESA platform for hosting external payloads onto ISS.[508] Originally scheduled to launch on 2 March 2020, the launch date was pushed back due to a second stage engine failure. SpaceX decided to swap out the second stage instead of replacing the faulty part.[509] It was SpaceX's 50th successful landing of a first stage booster, the third flight of the Dragon C112 and the last launch of the cargo Dragon spacecraft.								
83	18 March 2020, 12:16[510]	F9 B5 △ B1048.5	KSC, LC-39A	Starlink 5 v1.0 (60 satellites)	15,600 kg (34,400 lb)[5]	LEO	SpaceX	Success	Failure (drone ship)
	Fifth operational launch of Starlink satellites. It was the first time a first stage booster flew for a fifth time and the second time the fairings were reused (Starlink flight in May 2019).[511] Towards the end of the first stage burn, the booster suffered premature shut down of an engine, the first of a Merlin 1D variant and first since the CRS-1 mission in October 2012. However, the payload still reached the targeted orbit.[512] This was the second Starlink launch booster landing failure in a row, later revealed to be caused by residual cleaning fluid trapped inside a sensor.[513]								
84	22 April 2020, 19:30[514]	F9 B5 △ B1051.4	KSC, LC-39A	Starlink 6 v1.0 (60 satellites)	15,600 kg (34,400 lb)[5]	LEO	SpaceX	Success	Success (drone ship)

Data Collection - Scraping

Get response from HTML

```
static_url = "https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922"

response = requests.get(static_url)
```

Create Beautiful Soup Object

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

Find tables

```
# 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')
```

Get column names

```
column_names = []

for head in first_launch_table.find_all('th'):
    col_name = extract_column_from_header(head)
    if col_name is not None and len(col_name)>0:
        column_names.append(col_name)
```

Create dict-list structure to store data

```
launch_dict= dict.fromkeys(column_names)
```

Convert dict to dataframe

```
df=pd.DataFrame(launch_dict)
df
```

	Flight No.	Launch site	Payload	Payload mass	Orbit	Customer	Launch outcome	Version Booster	Booster landing	Date	Time
0	1	CCAFS	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success\n	F9 v1.0B0003.1	Failure	4 June 2010	18:45
1	2	CCAFS	Dragon	0	LEO	NASA	Success	F9 v1.0B0004.1	Failure	8 December 2010	15:43
2	3	CCAFS	Dragon	525 kg	LEO	NASA	Success	F9 v1.0B0005.1	No attempt\n	22 May 2012	07:44
3	4	CCAFS	SpaceX CRS-1	4,700 kg	LEO	NASA	Success\n	F9 v1.0B0006.1	No attempt	8 October 2012	00:35
4	5	CCAFS	SpaceX CRS-2	4,877 kg	LEO	NASA	Success\n	F9 v1.0B0007.1	No attempt\n	1 March 2013	15:10
...	...	...	...	...	...	...	...	...	...	...	...
116	117	CCSFS	Starlink	15,600 kg	LEO	SpaceX	Success\n	F9 B5B1051.10	Success	9 May 2021	06:42
117	118	KSC	Starlink	~14,000 kg	LEO	SpaceX	Success\n	F9 B5B1058.8	Success	15 May 2021	22:56
118	119	CCSFS	Starlink	15,600 kg	LEO	SpaceX	Success\n	F9 B5B1063.2	Success	26 May 2021	18:59
119	120	KSC	SpaceX CRS-22	3,328 kg	LEO	NASA	Success\n	F9 B5B1067.1	Success	3 June 2021	17:29
120	121	CCSFS	SXM-8	7,000 kg	GTO	Sirius XM	Success\n	F9 B5	Success	6 June 2021	04:26

Export to flat files (.csv)

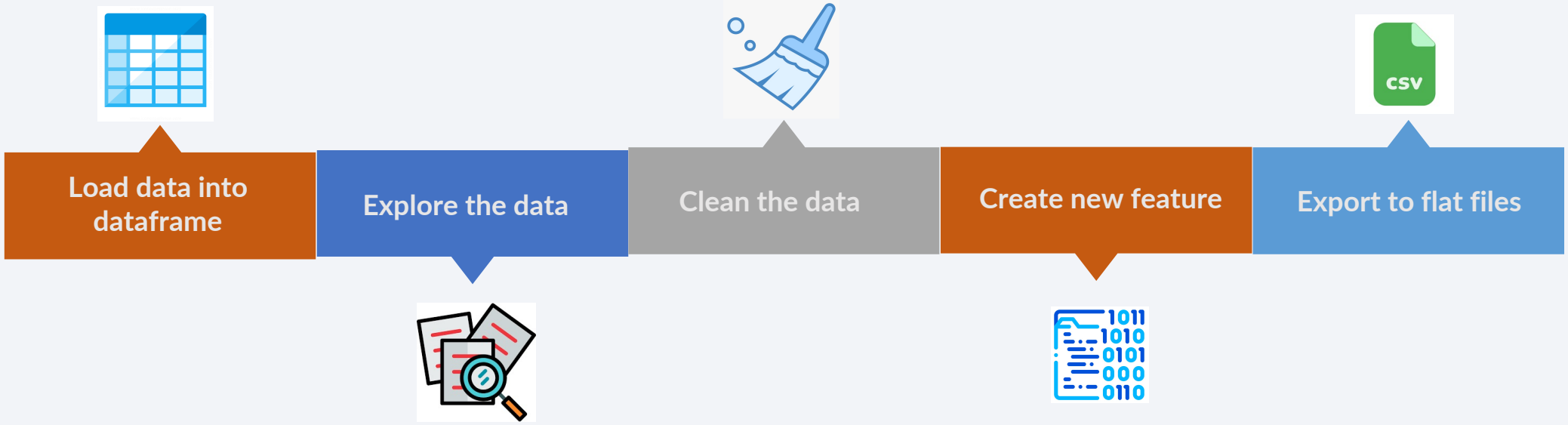
```
df.to_csv('spacex_web_scraped.csv', index=False)
```

Link to Github

Data Wrangling

The final goal of this project is to apply the dataset to a supervised model to decide whether the booster can land successfully or not. Normally this can be determined by a lot of factors. First let's explore and clean some features (data wrangling) to make them easier for us to feed in the machine learning algorithm.

Data wrangling is the process of cleaning and unifying messy and complex data sets for easy access and analysis.



Data Wrangling

Features of interest

- LaunchSite
- Orbit
- Outcome
- Derived Outcome label

Link to Github

Data Wrangling

Features of interest

- LaunchSite
- Orbit
- Outcome
- Derived Outcome label

Calculate the **number of launches on each site**

```
# Apply value_counts() on column LaunchSite
df.LaunchSite.value_counts()
```

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

VAFB SLC 4E - Cape Canaveral Space Launch Complex 40

SLC-4E - Vandenberg Air Force Base Space Launch Complex 4E

KSC LC 39A - Kennedy Space Center Launch Complex 39A

Link to Github

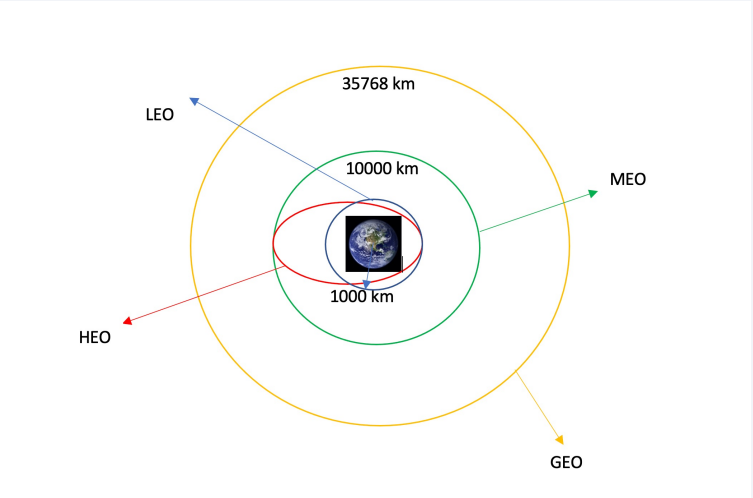
Data Wrangling

Features of interest

- LaunchSite
- Orbit
- Outcome
- Derived Outcome label

Calculate the **number and occurrence of each orbit**

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




```
# Apply value_counts on Orbit column
df.Orbit.value_counts()
```

```
GTO      27
ISS      21
VLEO     14
PO        9
LEO       7
SSO       5
MEO       3
ES-L1     1
SO        1
GEO       1
HEO       1
Name: Orbit, dtype: int64
```

Link to Github

Data Wrangling

Features of interest

- LaunchSite
- Orbit
- Outcome
- Derived Outcome label

Link to Github

Calculate the **number & occurences of mission outcome per orbit type**

```
: # landing_outcomes = values on Outcome column
  landing_outcomes = df.Outcome.value_counts()
  landing_outcomes
```

```
: True ASDS      41
  None None      19
  True RTLS      14
  False ASDS      6
  True Ocean      5
  None ASDS       2
  False Ocean     2
  False RTLS      1
  Name: Outcome, dtype: int64
```

True Ocean - mission outcome was successfully landed to a specific region of the ocean.
False Ocean - mission outcome was unsuccessfully landed to a specific region of the ocean.
True RTLS - mission outcome was successfully landed to a ground pad.
False RTLS - mission outcome was unsuccessfully landed to a ground pad.
True ASDS - mission outcome was successfully landed to a drone ship.
False ASDS - mission outcome was unsuccessfully landed to a drone ship.
None ASDS / **None None** - represent a failure to land.

Data Wrangling

Features of interest

- LaunchSite
- Orbit
- Outcome
- Derived Outcome label

Link to Github

Create a landing **outcome label** from **Outcome** column

```
bad_outcomes=set(landing_outcomes.keys()[[1,3,5,6,7]])
landing_class = [0 if outcome in bad_outcomes else 1 for outcome in df.Outcome]
df['Class']=landing_class
```

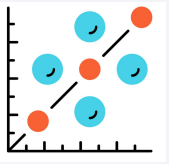
```
df.sample(5)
```

loadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block	ReusedCount	Serial	Longitude	Latitude	Class
75.000000	SSO	VAFB SLC 4E	True ASDS	1	True	False	True	5e9e3033383ecbb9e534e7cc	3.0	1	B1038	-120.610829	34.632093	1
35.000000	GTO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B1008	-80.577366	28.561857	0
00.000000	SSO	CCAFS SLC 40	True RTLS	4	True	True	True	5e9e3032383ecb267a34e7c7	5.0	3	B1059	-80.577366	28.561857	1
25.000000	GTO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B1005	-80.577366	28.561857	0
04.959412	LEO	CCAFS SLC 40	True RTLS	1	True	False	True	5e9e3032383ecb267a34e7c7	4.0	1	B1043	-80.577366	28.561857	1

```
# success rate
df["Class"].mean()
```

```
0.6666666666666666
```

EDA with Data Visualization

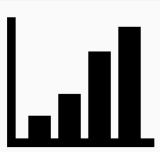


Scatter Plots

Scatter plots can visualize dependencies of attributes with each other. This is usually called correlation. Once a pattern is determined, it will help us to predict which factor would lead to the success of landing outcome.

- Flight Number and Payload
- Flight Number and Launch Site
- Payload and Launch Site
- Flight Number and Orbit type
- Payload and Orbit type

Link to Github



Bar Graphs

Bar graph is a great way to visualize categorical data type. Each bar represents a member under that categorical feature, and it is easy to compare which member has the largest amount and which the least.

- success rate of each orbit type

Line Plot

Line Plot is similar to scatter plot while the former is often used to visualize continuous numeric data type. From line plot, we can easily tell the trend between two variables thus helping us to make prediction about the future.

- launch success yearly trend

EDA with SQL

SQL is an indispensable tool for data scientist and data analyst as most of the real-world data are stored in databases. SQL with its full name of *Structured Query Language*, can easily help us visit the data as well as analyzing and drawing useful insights from it. In this project, we use **IBM Db2**, which is a fully managed SQL database provided as a service.

The following questions are explored:

- Display the names of the unique launch sites in the space mission.
- Display 5 records where launch sites begin with the string 'CCA'.
- Display the total payload mass carried by boosters 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 have payload mass greater than 4000 but less than 6000.
- List the total number of successful and failure mission outcomes.
- List the names of the booster versions which have carried the maximum payload mass. Use a subquery.
- List the failed landing outcomes in drone ship, their booster versions, and launch site names for in year 2015.
- 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.

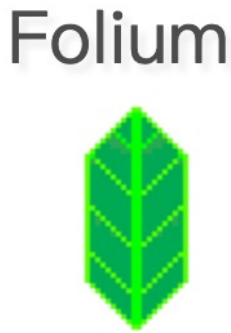
Link to Github

Build an Interactive Map with Folium

Folium makes it easy to visualize geospatial data that' s been manipulated in Python on an interactive leaflet map. It enables both the binding of data to a map for choropleth visualizations as well as passing rich vector/raster/HTML visualizations as markers on the map.

In this project, we:

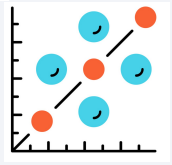
- Place each launch site on the map as a marker.
- Render successful land outcome as **GREEN** color and unsuccessful one as **RED** color.
- Enable launch sites with the same coordinates to fold/unfold based on zoom level.
- Calculate and visualize the distance between launch sites and some certain kinds of their proximities.



Map Objects	Code	Result
Map Marker	folium.Marker()	Map object to a mark on the map.
Icon Marker	folium.Icon()	Create an icon on the map.
Circle Marker	folium.Circle()	Create a circle on the map.
PolyLine	folium.PolyLine()	Create a line between two coordinates.
Marker Cluster Object	plugins.MarkerCluster()	Automatically aggregate multiple markers with the same coordinates into a cluster marker on a certain zoom level.

Link to Github

Build a Dashboard with Plotly Dash



Pie Chart – shows the successful landing rate for each launch site or overall sites, according to the dropdown menu selected.

Scatter Plot – shows the correlation between two variables. The following has been explored:

- Flight Number and Payload
- Flight Number and Launch Site
- Payload and Launch Site
- Flight Number and Orbit type
- Payload and Orbit type

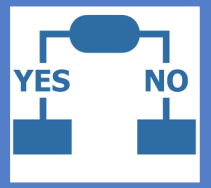
Objects	Codes	Result
Dash and its components	import dash import dash_html_components as html import dash_core_components as dcc from dash.dependencies import Input, Output	Plotly Dash is simple enough that you can bind a user interface to Python, R, Julia, or F# code in less than 10 minutes. With Dash Open Source, Dash apps can run on a local laptop or server. The Dash Core Component library contains a set of higher-level components such as sliders, graphs, dropdowns, tables and more. Dash provides all of the available HTML tags as user-friendly Python classes.
Plotly	import plotly.express as px	Plots graphs with interactive plotly library.
Dropdown	dcc.Dropdown()	Create a dropdown menu for launch sites.
RangeSlider	dcc.RangeSlider()	Create a range slider for PayLoadMass range selection.
Pie Chart	px.pie()	Create pie charts for successful landing rate of each launch site.
Scatter Plot	px.scatter()	Create scatter plot for correlation among features.

Link to Github

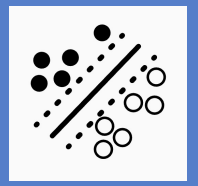
Predictive Analysis (Classification)

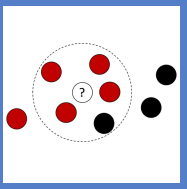
Classification Model Candidates


Logistic Regression



Decision Tree


SVM


KNN

1 Build

- Load data into NumPy & Pandas
- Transform and standardize data
- Split data into training and testing set
- Check testing sample size
- List down suitable machine learning models
- Wrap hyperparameters into GridSearchCV object
- Fit training data to machine learning models

2 Evaluate

- Check accuracy for each model
- Get tuned hyperparameters for each type of algorithm
- Plot confusion metrics

3 Improve

- Feature engineering
- Algorithm tuning

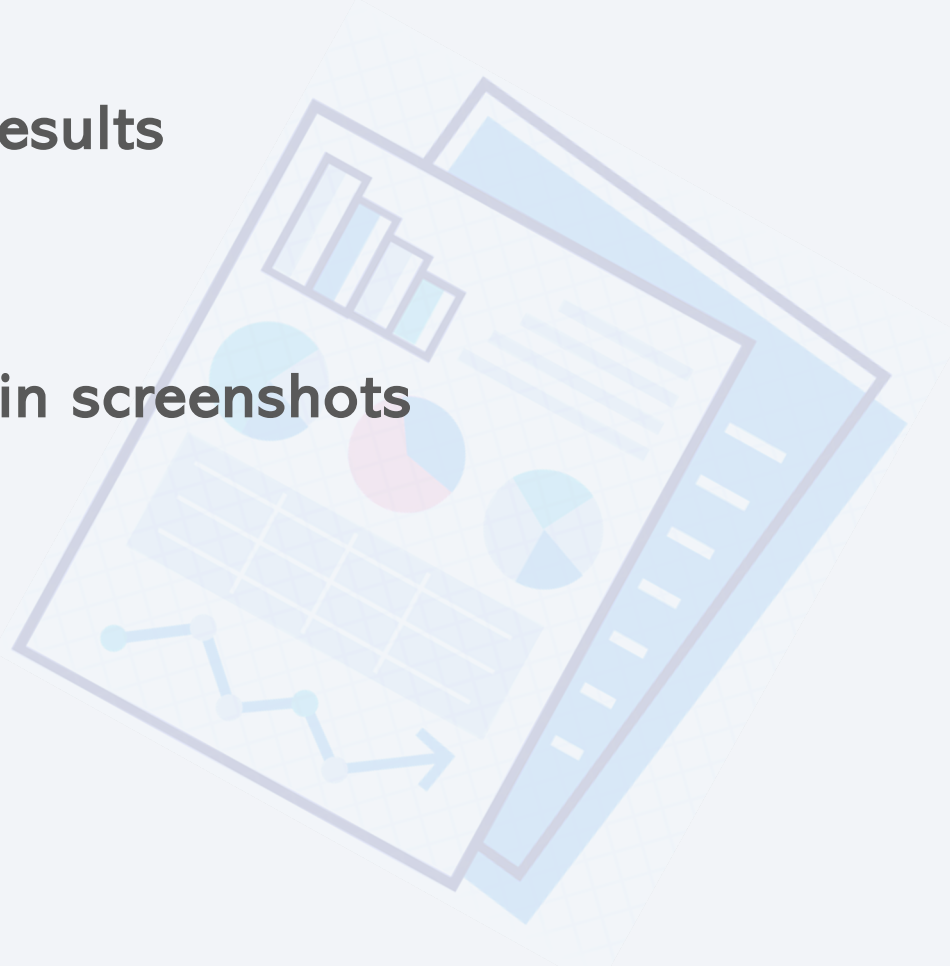
4 Select

- The model with the best accuracy score on testing data wins the best performing classification model

Link to Github

Results

1. Exploratory data analysis results
2. Interactive analytics demo in screenshots
3. Predictive analysis results

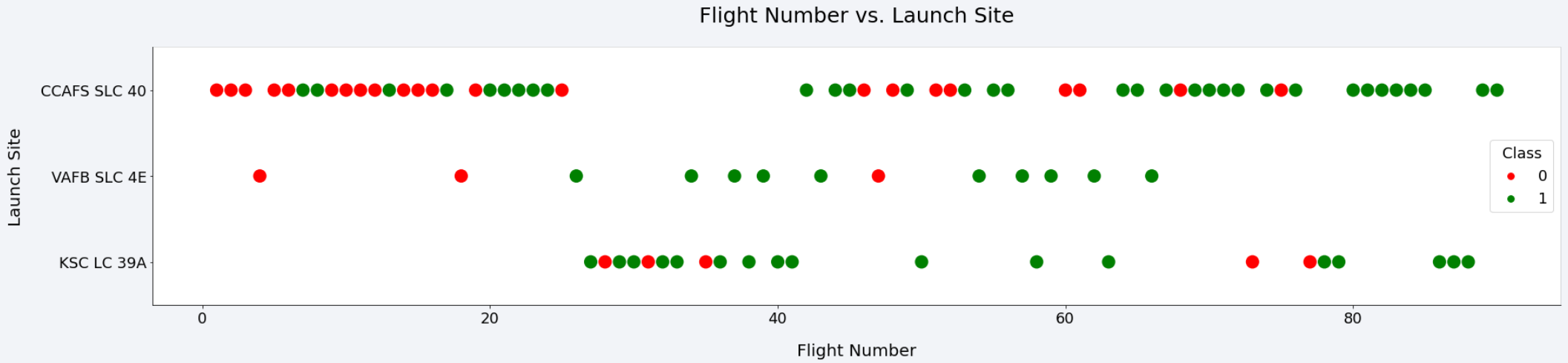


Section 2

EDA with Visualization

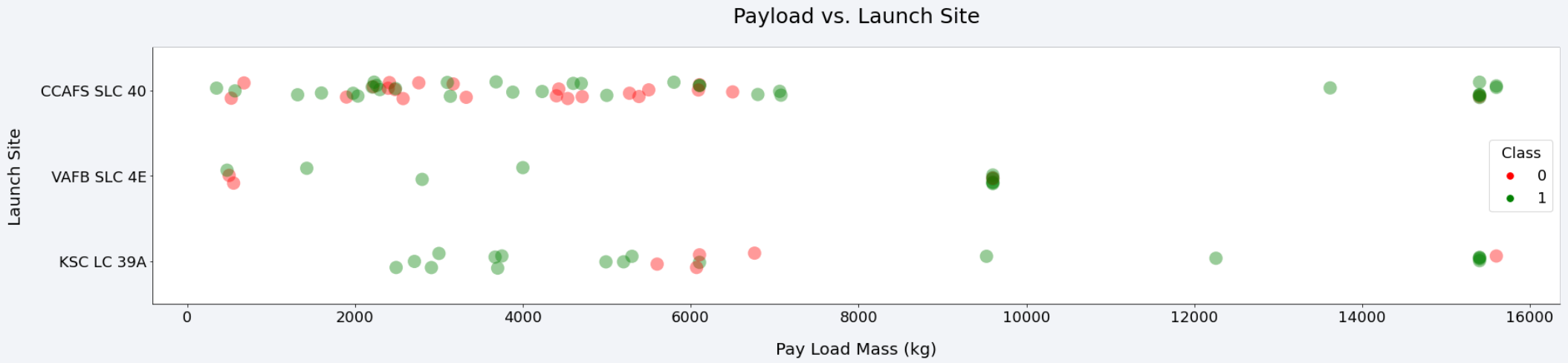
Flight Number vs. Launch Site

The higher the flight number (around 20+) is, the more likely the landing outcome would be a success.



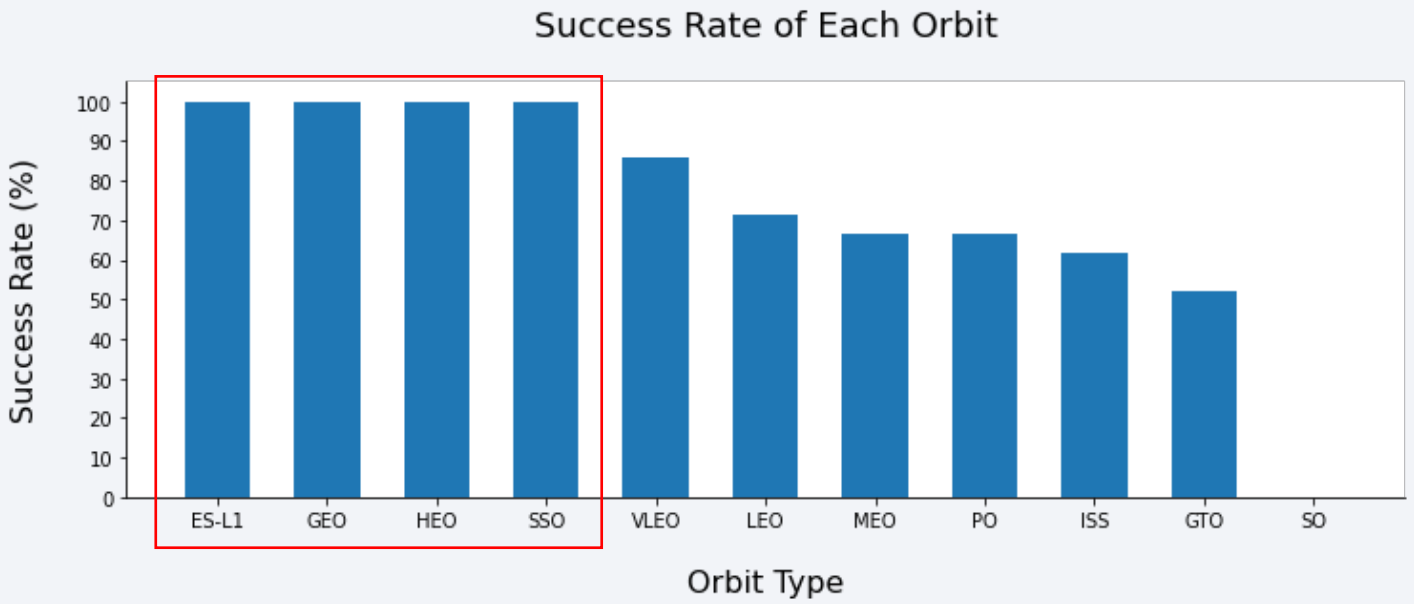
Payload vs. Launch Site

To some extent, when the payload is above 7,000 kg, the more likely a successful landing would appear. However, there is no obvious evidence to determine if the relationship between launch site and payload would lead a successful landing outcome.



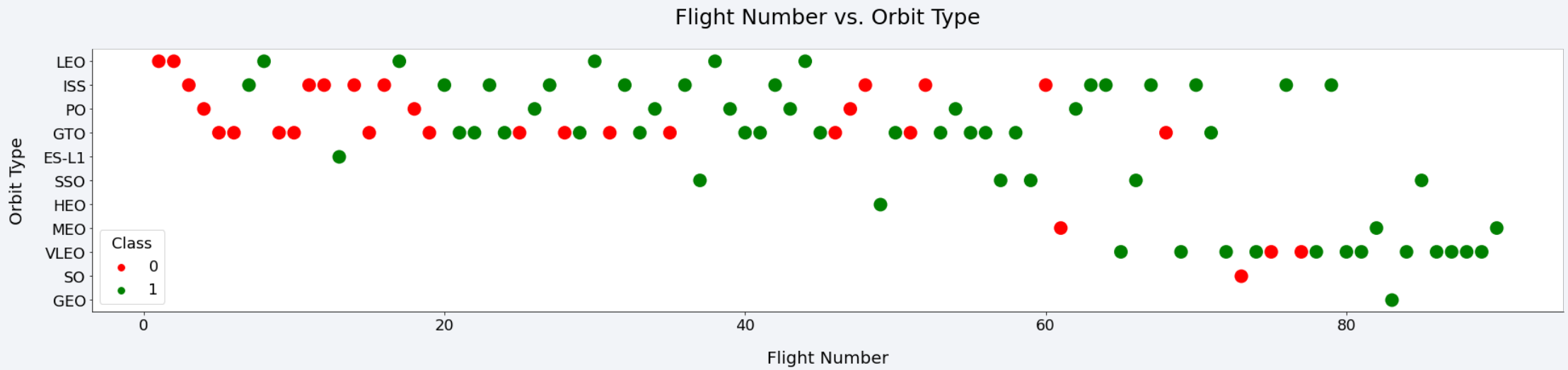
Success Rate vs. Orbit Type

Obviously, orbit type in *ES-L1, GEO, HEO, SSO* has the highest successful landing rate.



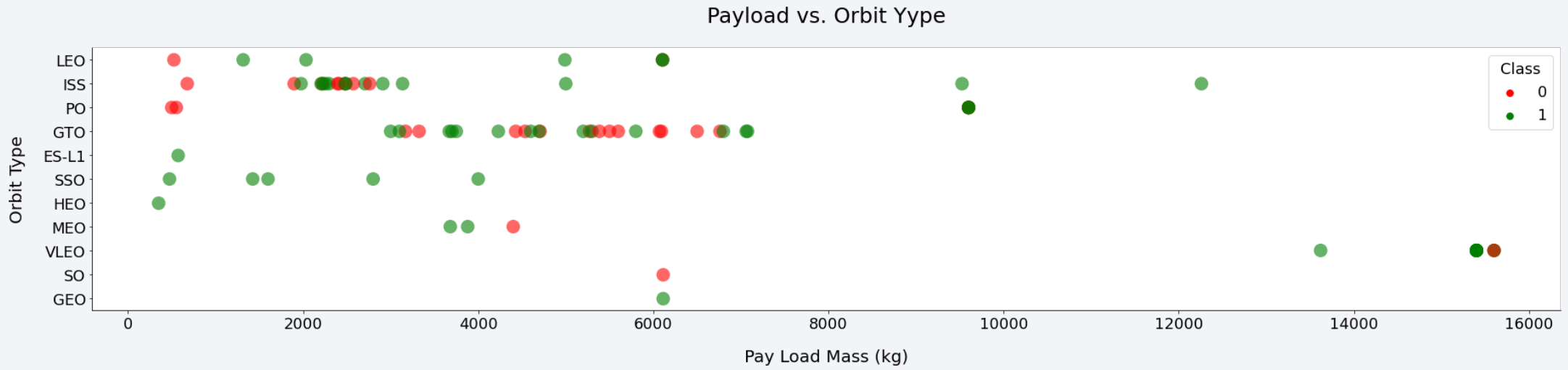
Flight Number vs. Orbit Type

- **The success for LEO orbit appears related to the number of flights;**
- **On the other hand, there seems to be no relationship between flight number when in orbits such as GTO, ISS ...**



Payload vs. Orbit Type

- Heavy payloads have a negative influence on GTO orbits.
- Heavy payloads have a positive on GTO and Polar LEO, ISS orbits.



Launch Success Yearly Trend

In general trend, the success rate began to climb up since 2013, although with a slight dip in 2018 and 2020.

Section 3

EDA with SQL

All Launch Site Names

Task 1: Find the names of the **unique** launch sites.

Using the keyword DISTINCT in the query we pulled the unique values in launch_site column from the table.

```
%sql SELECT DISTINCT(launch_site) FROM spacextbl
```

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

Link to Github

Launch Site Names Begin with 'CCA'

Task 2: Find 5 records where launch sites **begin with `CCA`**

- Using wildcard `CCA%` on launch_site column we can filter only the records with launch_site name starting with `CCA`
- Using the keyword ‘LIMIT 5’ in the query we can extract only 5 records from the above result.

```
%sql SELECT * FROM spacextbl WHERE launch_site LIKE 'CCA%' LIMIT 5
```

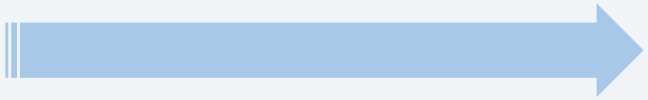
DATE	time__utc_	booster_version	launch_site	payload	payload_mass__kg_	orbit	customer	mission_outcome	landing__outcome
2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
2010-12-08	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of Brouere cheese	0	LEO (ISS)	NASA (COTS) NRO	Success	Failure (parachute)
2012-05-22	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
2012-10-08	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
2013-03-01	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

Total Payload Mass

Task 3: Calculate the total payload carried by boosters from NASA (CRS).

- Using function *SUM()* on the `payload_mass_kg` column we can get the summation of all payload weights;
- Using *WHERE* clause, we can get only the summation matching the condition *customer = `NASA (CRS)`*

```
%%sql
SELECT sum(payload_mass__kg_) AS "total_payload_mass_by_NASA (CRS)"
FROM spacextbl
WHERE customer = 'NASA (CRS)';
```



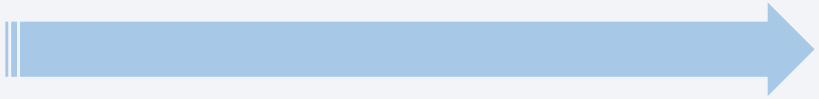
total_payload_mass_by_NASA (CRS)
45596

Average Payload Mass by F9 v1.1

Task 4: Calculate the average payload mass carried by booster version F9 v1.1

- Using function *AVG()* on the `payload_mass_kg` column we can get the average weights of all payload;
- Using *WHERE* clause, we can get only the record matching the condition *booster_version = `F9 v1.1`*

```
%%sql
SELECT AVG(payload_mass__kg_)
FROM spacextbl
WHERE booster_version = 'F9 v1.1'
```



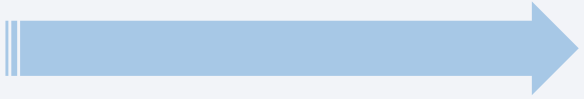
Average Payload Mass by Booster Version F9 v1.1
2928.400000

First Successful Ground Landing Date

Task 5: Find the dates of the first successful landing outcome on ground pad.

- Using function *MIN()* on the `DATE` column we can get the earliest record of all;
- Using *WHERE* clause, we can get only the record matching the condition *landing_outcome= `Success (ground pad)`*

```
%%sql
SELECT MIN(DATE) AS "First Successful Landing Outcome on Ground Pad"
FROM spacextbl
WHERE landing__outcome='Success (ground pad)';
```



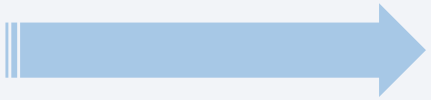
First Successful Landing Outcome on Ground Pad
2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

Task 6: List the names of boosters which have successfully landed on drone ship and had payload mass greater than 4000 but less than 6000.

- *WHERE* clause can be used with logical operators, so we can get records matching with multiple conditions. Here we have limitations in both `payload_mass_kg` and `landing_outcome` columns.

```
%%sql
SELECT booster_version
FROM spacextbl
WHERE (payload_mass__kg_>4000 and payload_mass__kg_<6000) and landing__outcome='Success (drone ship)'
```



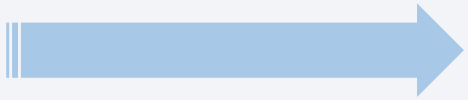
booster_version
F9 FT B1022
F9 FT B1026
F9 FT B1021.2
F9 FT B1031.2

Total Number of Successful and Failure Mission Outcomes

Task 7: Calculate the total number of successful and failure mission outcome.

- Firstly we count the successful mission by using function `*COUNT()*` with *WHERE* clause by setting *mission_outcome LIKE 'Success%'* , then take the result as a temporary table;
- Secondly we count failed mission by using function `*COUNT()*` with *WHERE* clause by setting *mission_outcome LIKE `Failure %'* , then take the result as a temporary table;
- Finally we put the two results together by combining the two temporary tables above.

```
%%sql
SELECT * FROM
  (SELECT COUNT(*) AS "Success Count"
    FROM spacextbl
    WHERE mission_outcome LIKE 'Success%') tb1,

  (SELECT COUNT(*) AS "Failure Count"
    FROM spacextbl
    WHERE mission_outcome LIKE 'Failure%') tb2;
```



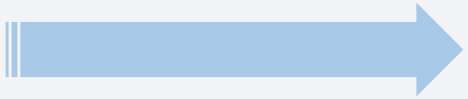
Success Count	Failure Count
100	1

Boosters Carried Maximum Payload

Task 8: List the names of the booster which have carried the maximum payload mass.

- We use function `*MAX()*` to work out the maximum payload in `payload_mass_kg` column as sub-query.
- Then we select the booster by setting the above sub-query as *WHERE* clause condition.

```
%%sql
SELECT booster_version
FROM spacextbl
WHERE payload_mass__kg_=
    (SELECT MAX(payload_mass__kg_)
    FROM spacextbl)
```



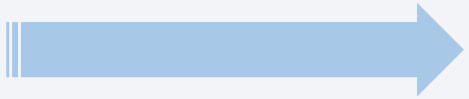
booster_version
F9 B5 B1048.4
F9 B5 B1049.4
F9 B5 B1051.3
F9 B5 B1056.4
F9 B5 B1048.5
F9 B5 B1051.4
F9 B5 B1049.5
F9 B5 B1060.2
F9 B5 B1058.3
F9 B5 B1051.6
F9 B5 B1060.3
F9 B5 B1049.7

2015 Launch Records

Task 9: List the failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015

- Logical operator was used in *WHERE* clause to combine multiple conditions.
- In the first condition, we limit the year to be 2015 using function `*EXTRACT()*`on DATE column.
- In the second condition, we limit the outcome to be `Failure (drone ship)` in `landing__outcome` column

```
%%sql
SELECT DATE, landing__outcome, booster_version, launch_site
FROM spacextbl
WHERE (EXTRACT(YEAR FROM DATE) = 2015) AND (landing__outcome='Failure (drone ship)')
```



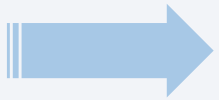
DATE	landing__outcome	booster_version	launch_site
2015-01-10	Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40
2015-04-14	Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40

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

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.

- Use *WHERE* clause to limit the date range between 2010-06-04 and 2017-03-20;
- CTE technique was used to wrap the query result above as a common table expression (CTE);
- From the CTE, we count the number in each type of `*landing__outcome*` by using `*GROUP BY*` KEYWORD;
- Then we sort the results in descending order by using `*ORDER BY DESC*` KEYWORD.

```
%%sql
WITH tmp AS
  (SELECT *
    FROM spacextbl
    WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20' )

SELECT landing__outcome, COUNT(*) AS counts
FROM tmp
GROUP BY landing__outcome
ORDER BY 2 DESC
```



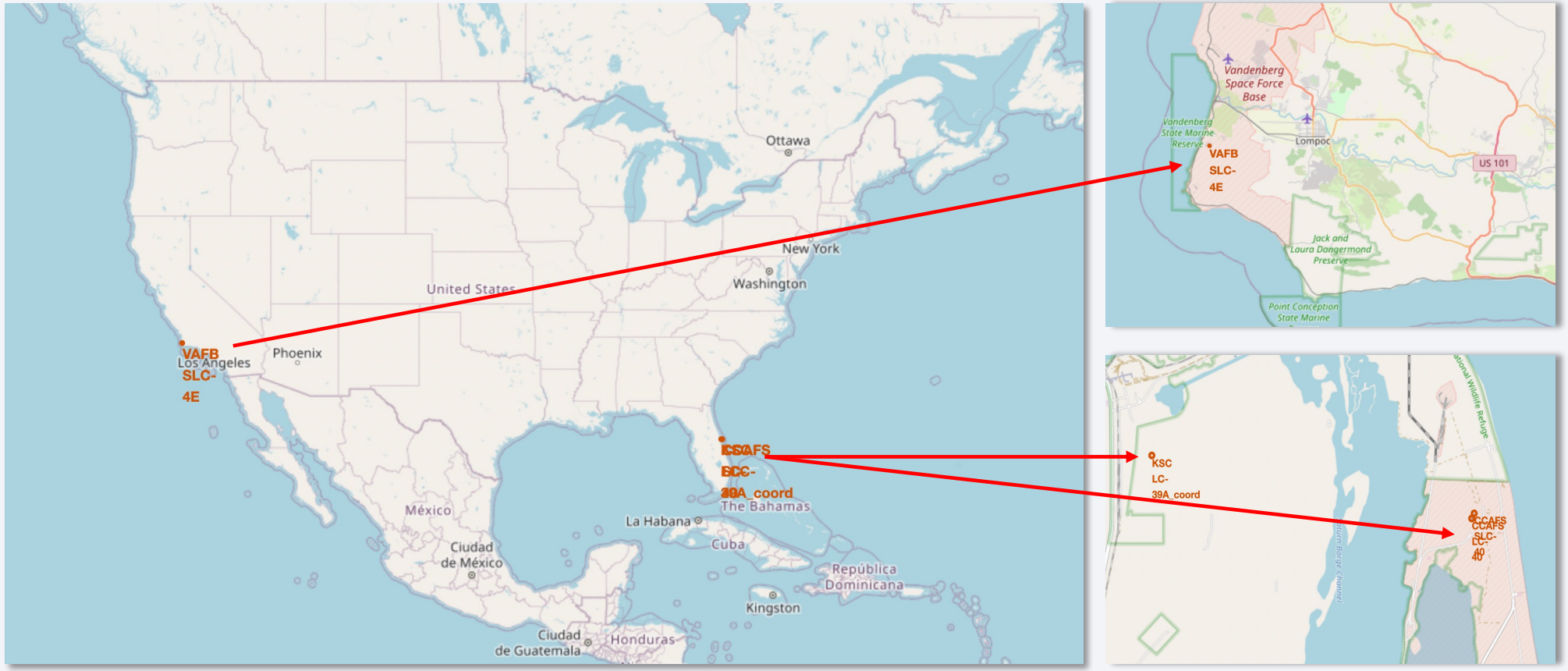
landing__outcome	counts
No attempt	10
Failure (drone ship)	5
Success (drone ship)	5
Controlled (ocean)	3
Success (ground pad)	3
Failure (parachute)	2
Uncontrolled (ocean)	2
Precluded (drone ship)	1

Section 4

Launch Sites Proximities Analysis

All Launch Sites on the Map

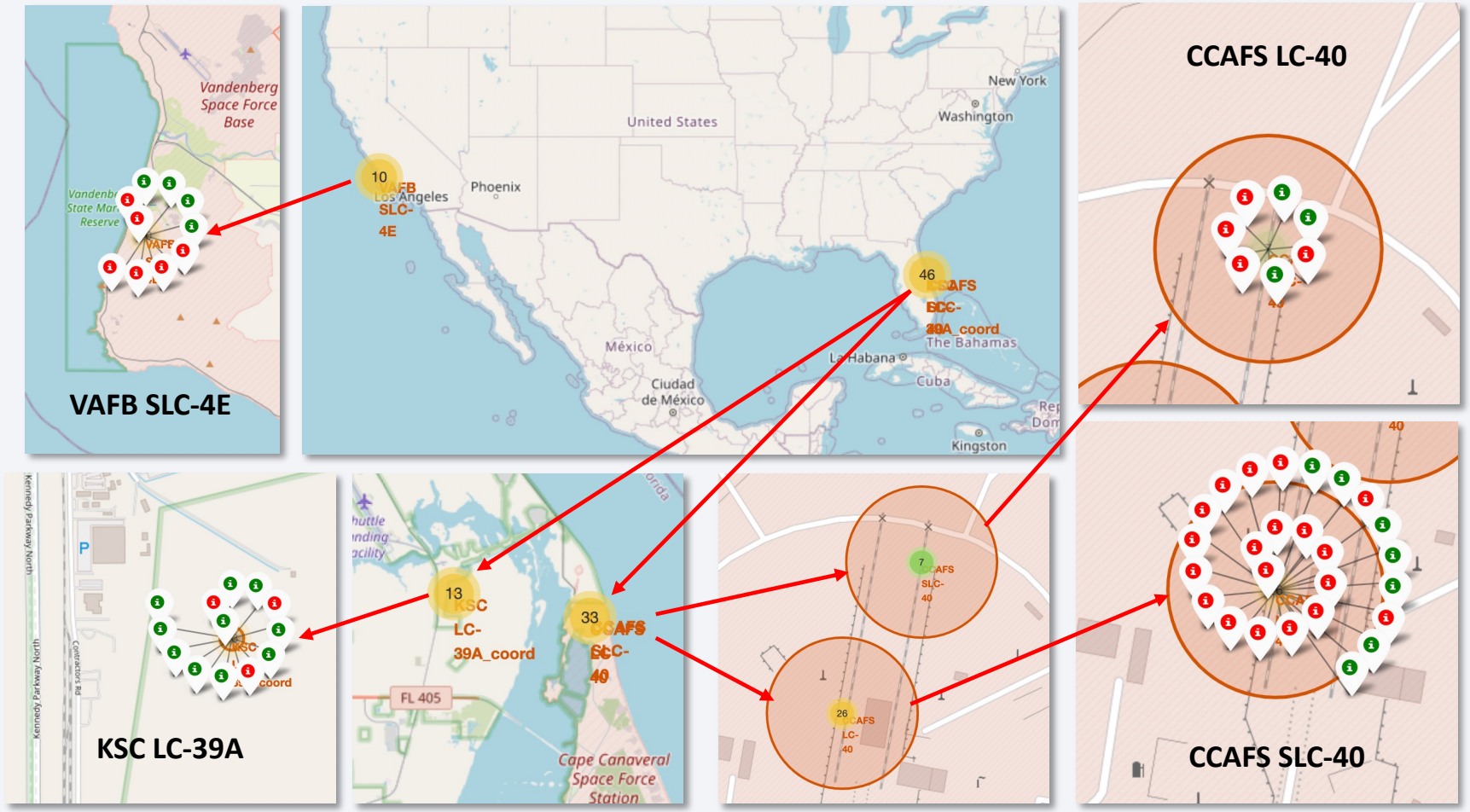
We can see that SpaceX launch sites are all very close to the coastlines (i.e. California & Florida)



Color Labeled Markers

- Green label represent successful landing outcome
- Red label represents unsuccessful landing outcome;

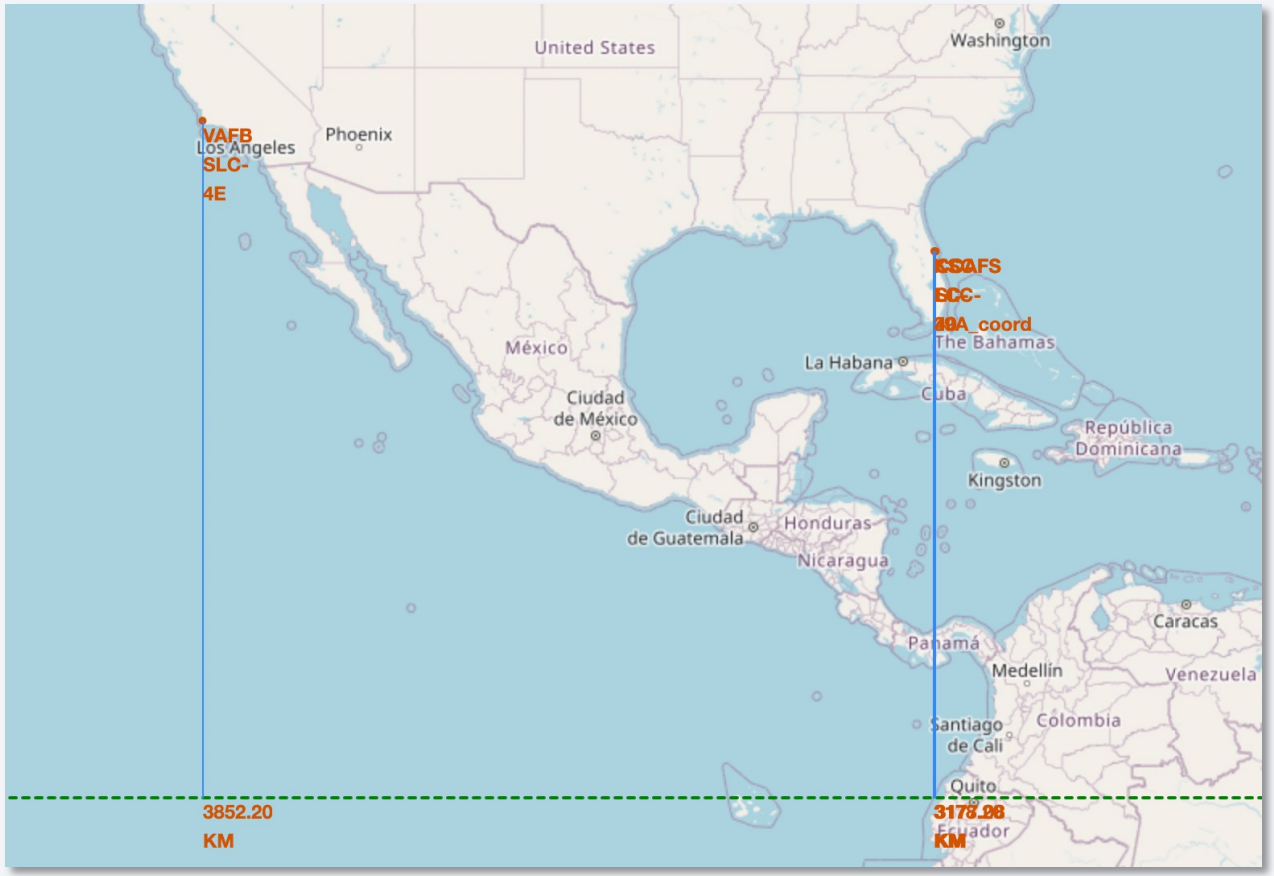
- KSC LC-39A has the **largest** successful landing ratio;
- CCAFS SLC-40 has the **largest** unsuccessful landing ratio;

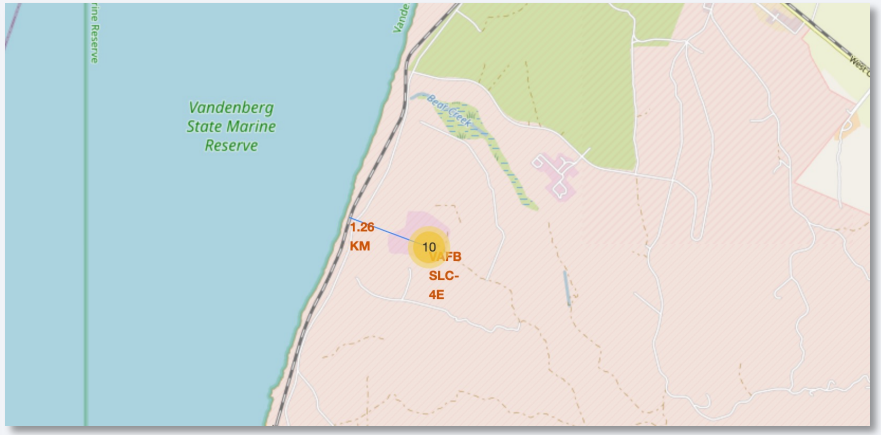


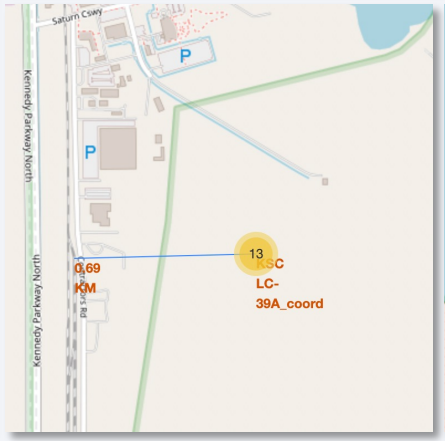
Launch Site Distance from Equator & Railways

All launch sites are far away from the **equator,** greater than 3,000 km.

Distance to railway ranges between 0.5 km to 1.5 km. All launch sites are in close proximity to **railways**.

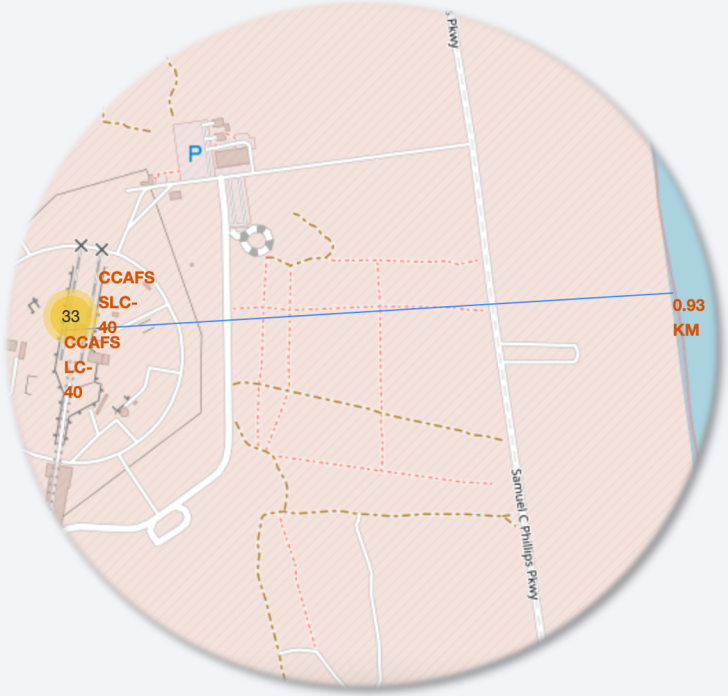




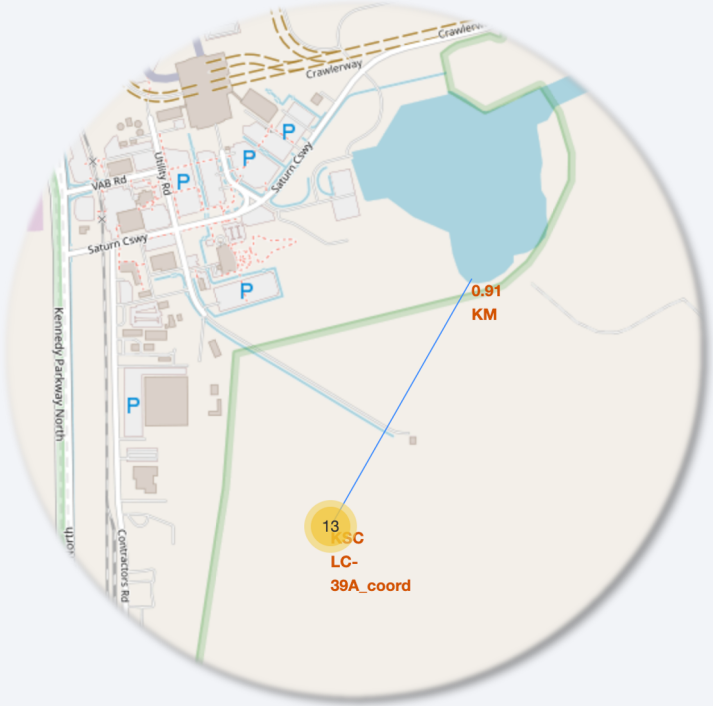


Launch Site Distance from Coastlines

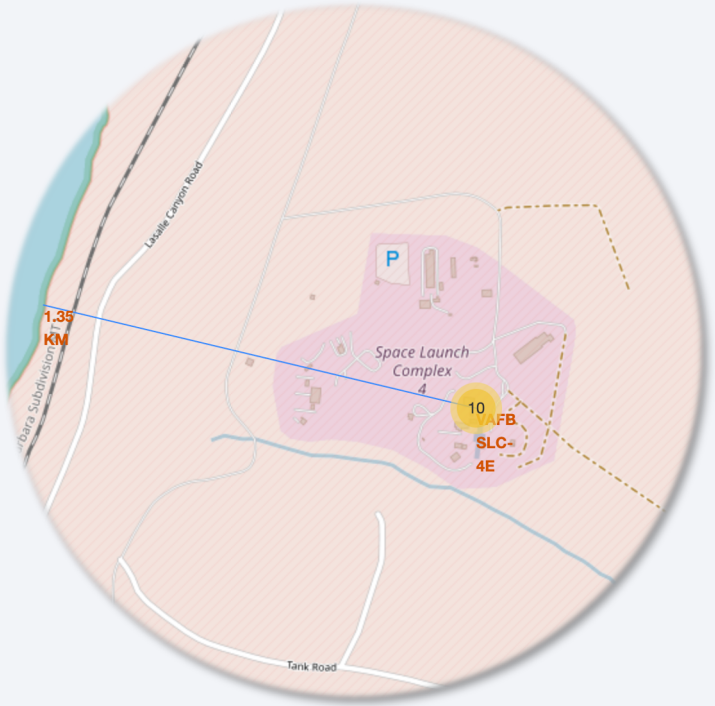
All launch sites are in close proximity to **coastlines,** less than 1.5 km.



CCAFS SLC-40 / CCAFS LC-40



KSC LC-39A

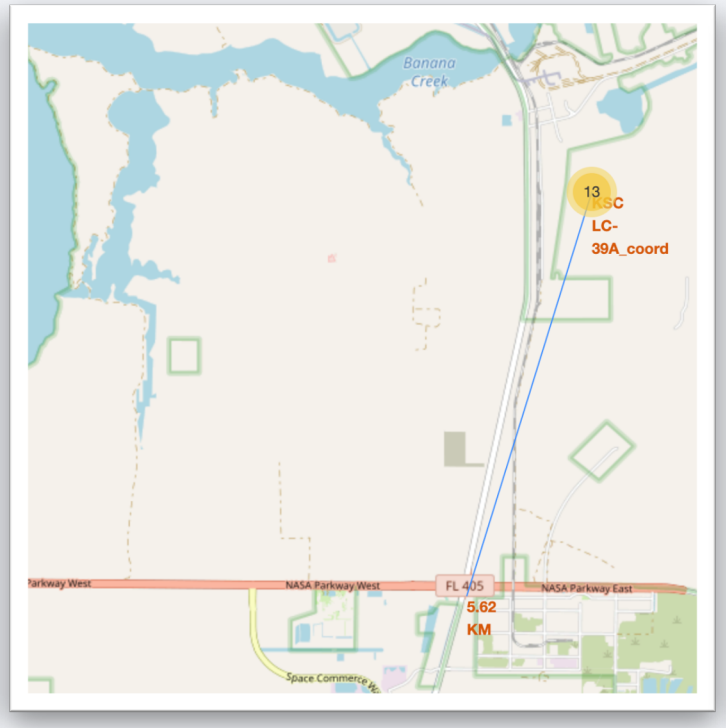


VAFB SLC-4E

Launch Site Distance from Highways

All launch sites are relatively far away from **highways,** greater than 5 km.

CCAFS SLC-40 / CCAFS LC-40



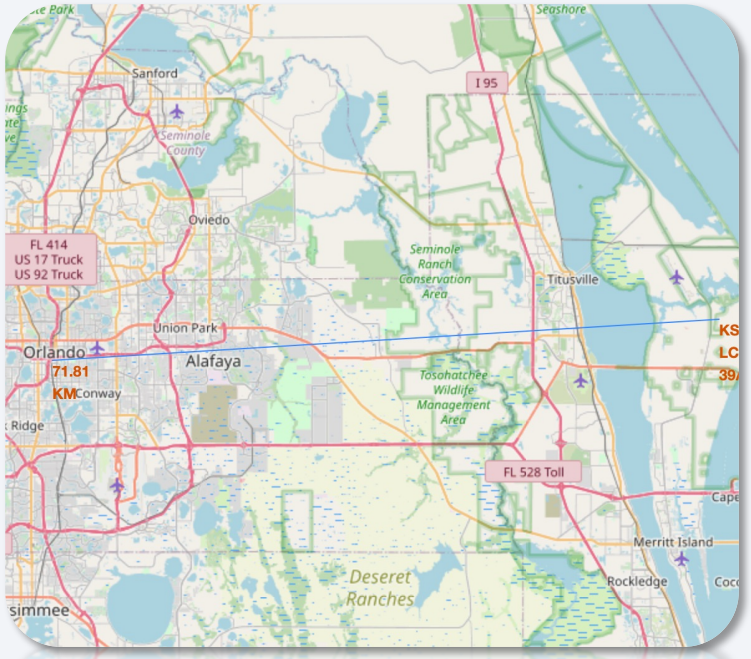
KSC LC-39A

VAFB SLC-4E

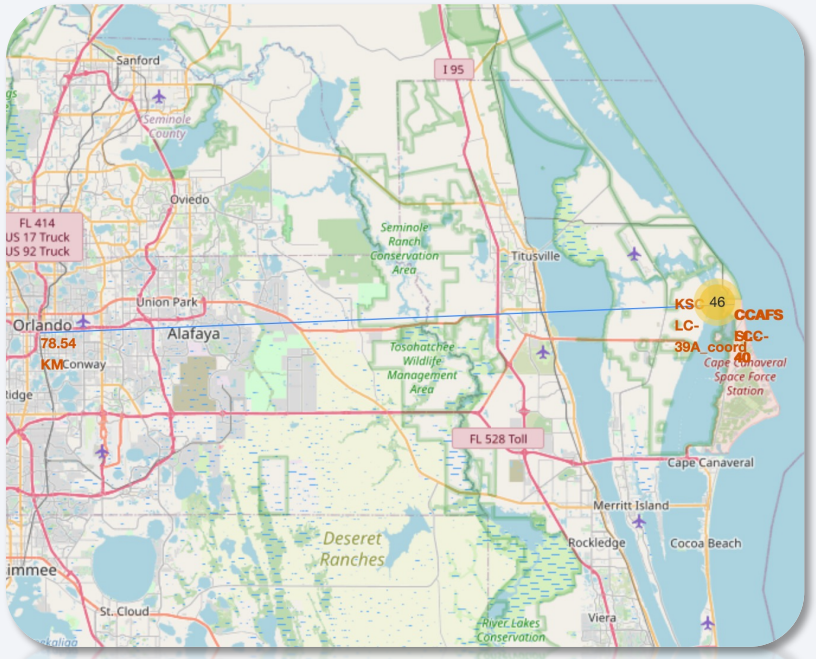
Launch Site Distance from Cities

All launch sites are far away from their **neighboring cities,** greater than 75 km.

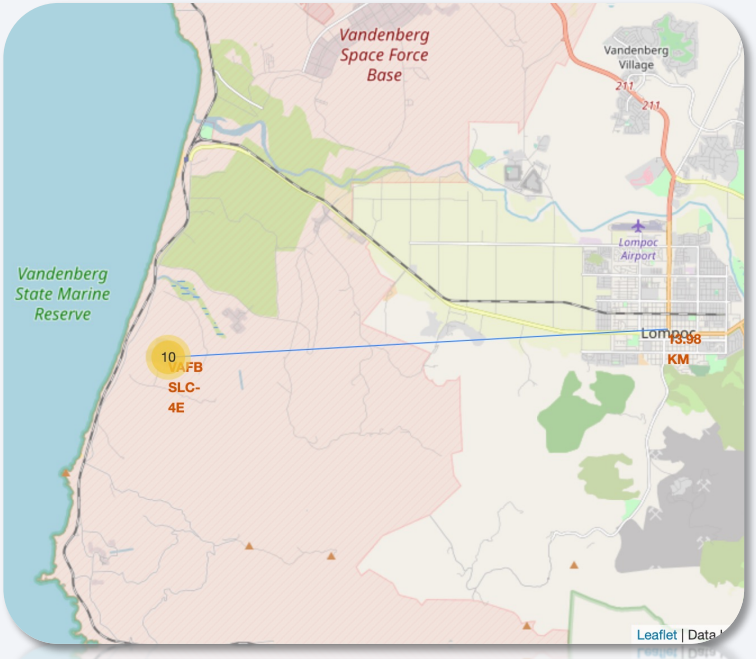
KSC LC-39A

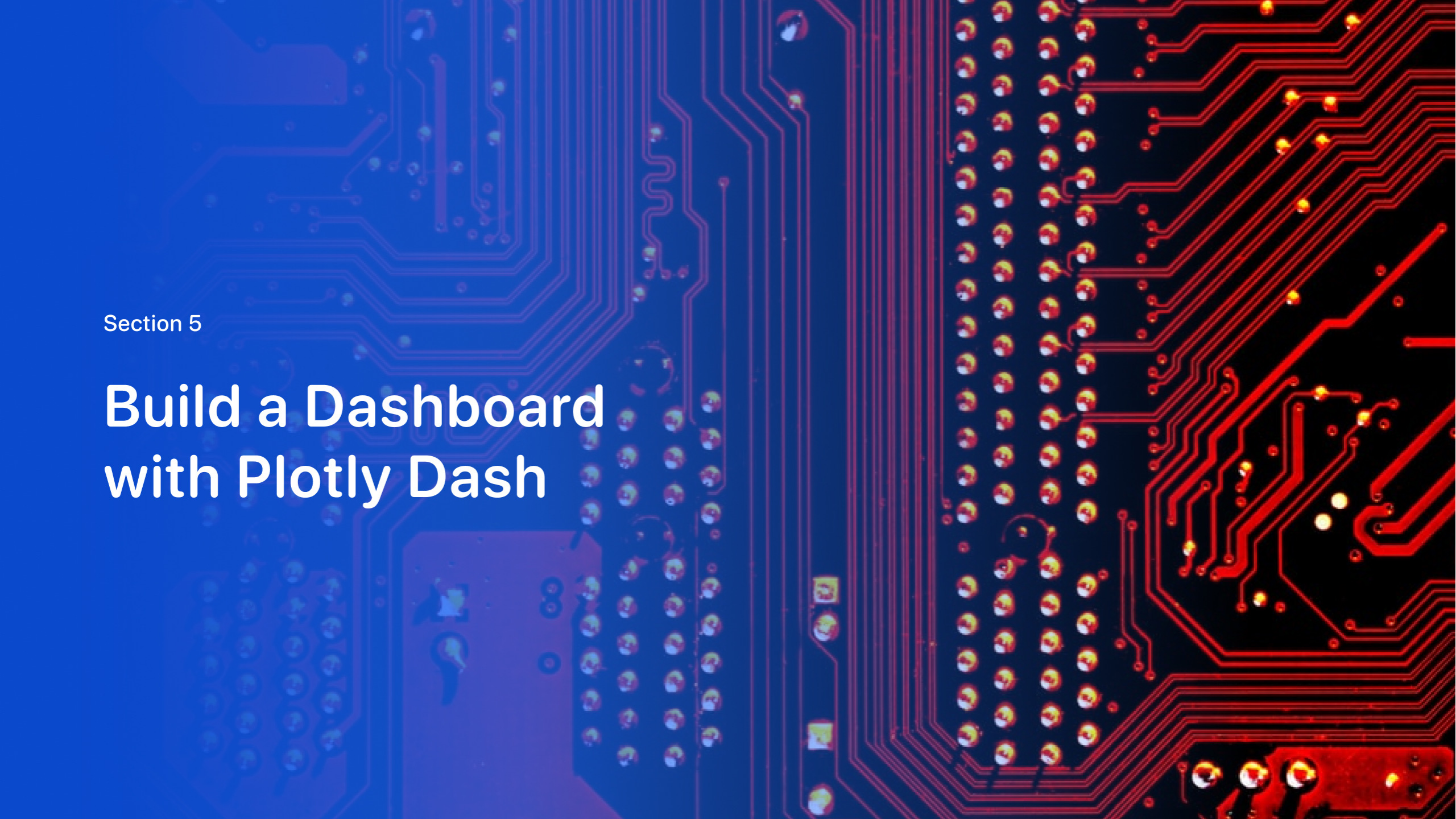


CCAFS SLC-40 / CCAFS LC-40



VAFB SLC-4E



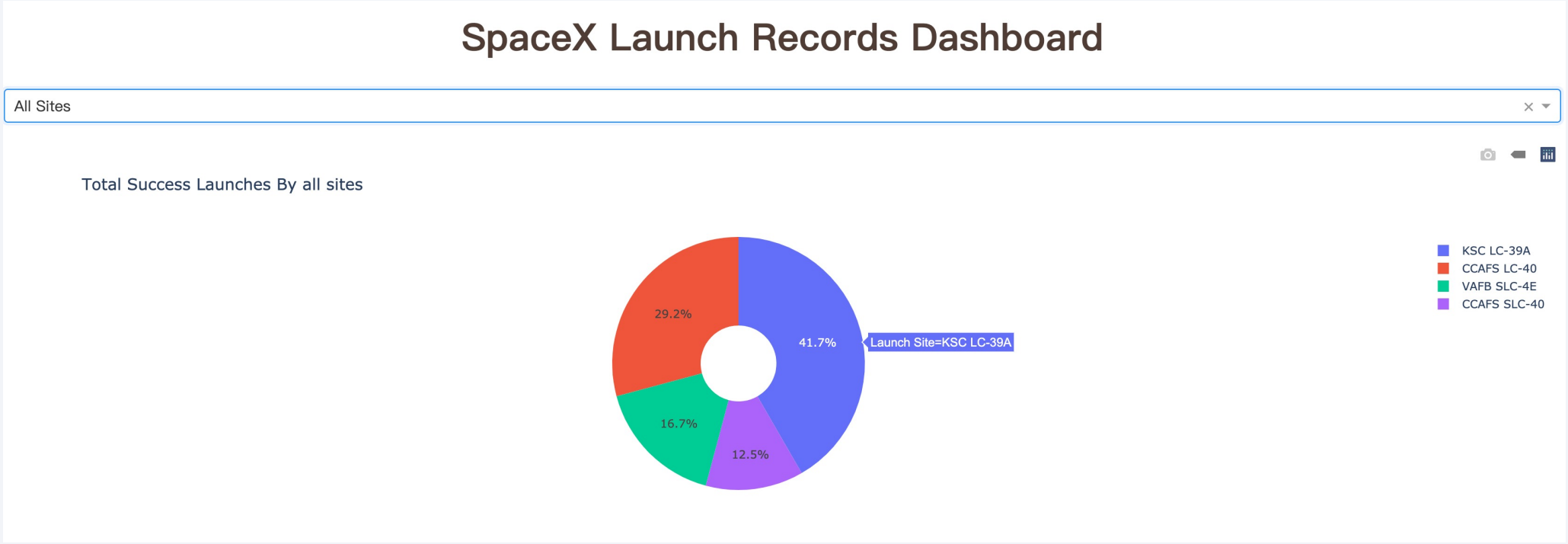


Section 5

Build a Dashboard with Plotly Dash

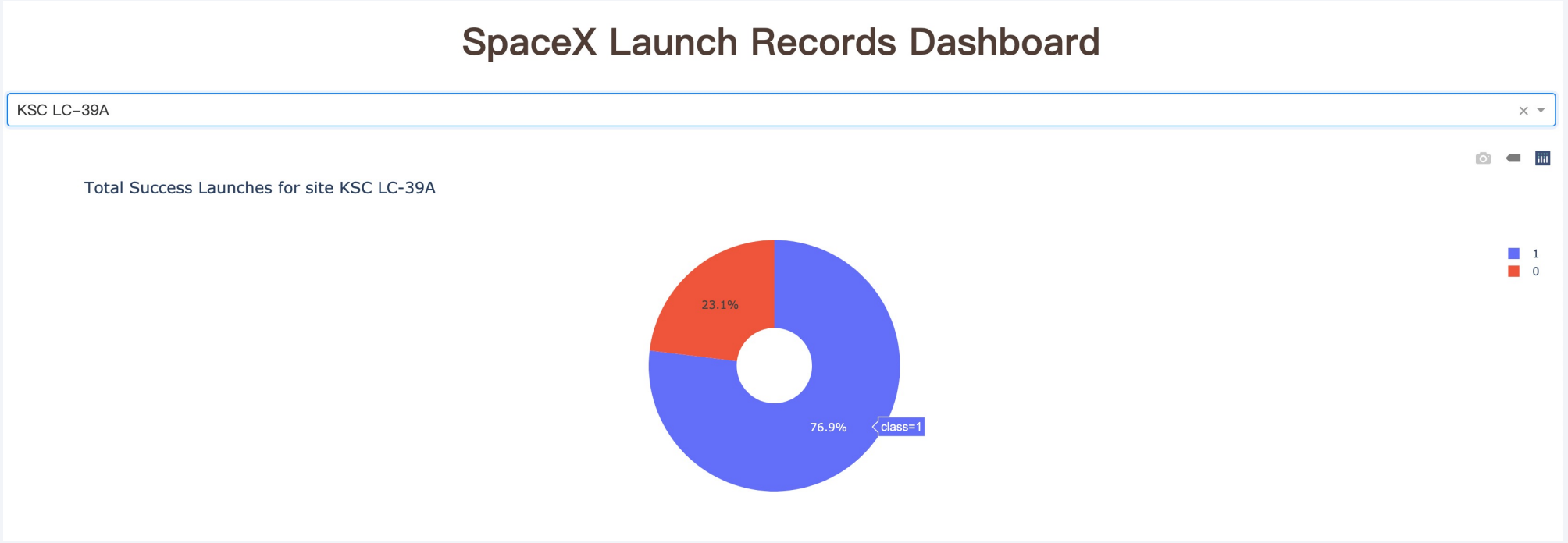
Launch Success Ratio for All Successful Launch Records

We can see that **KSC LC-39A (41.7%)** has the ***MOST*** successful launches of all successful launches.



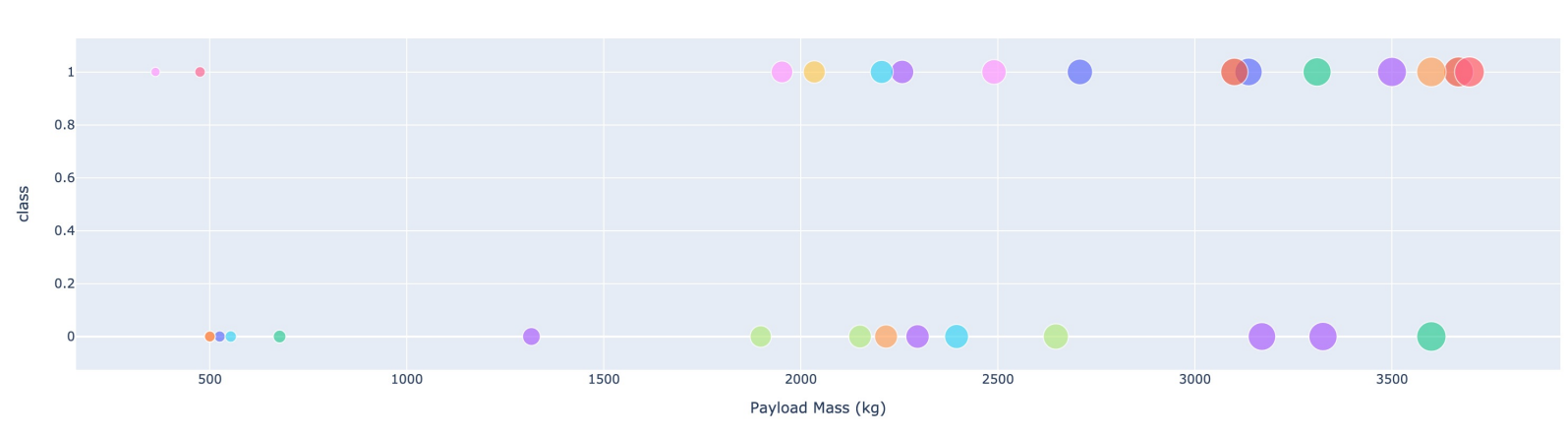
Success Rate for the Launch Site with Most Success Launch Count Percentage

In the last slide, we know that **KSC LC-39A** has the ***MOST*** successful launch coverage of all successful launches. Furthermore from the pie chart below, we can see that **KSC LC-39A** has as high as **76.9%** of success rate.



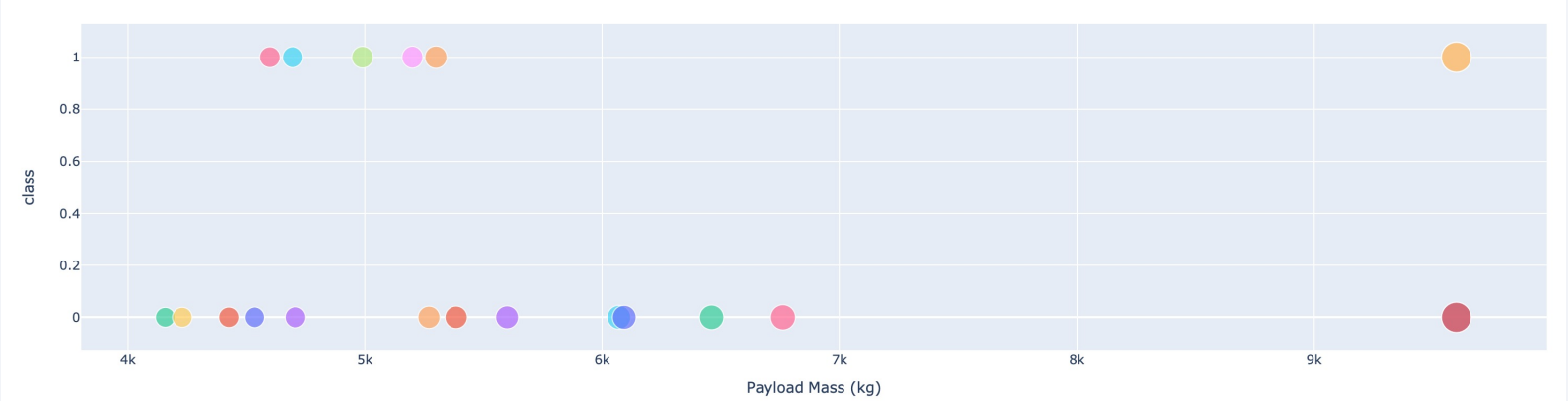
Effect of Payload Mass on Successful Launches

Low weighted payload between **0 and 4,000 kg**



We can see that the successful launches' count is likely larger when payload mass is below 4,000 kg.

Heavy weighted payload between **above 4,000 kg**

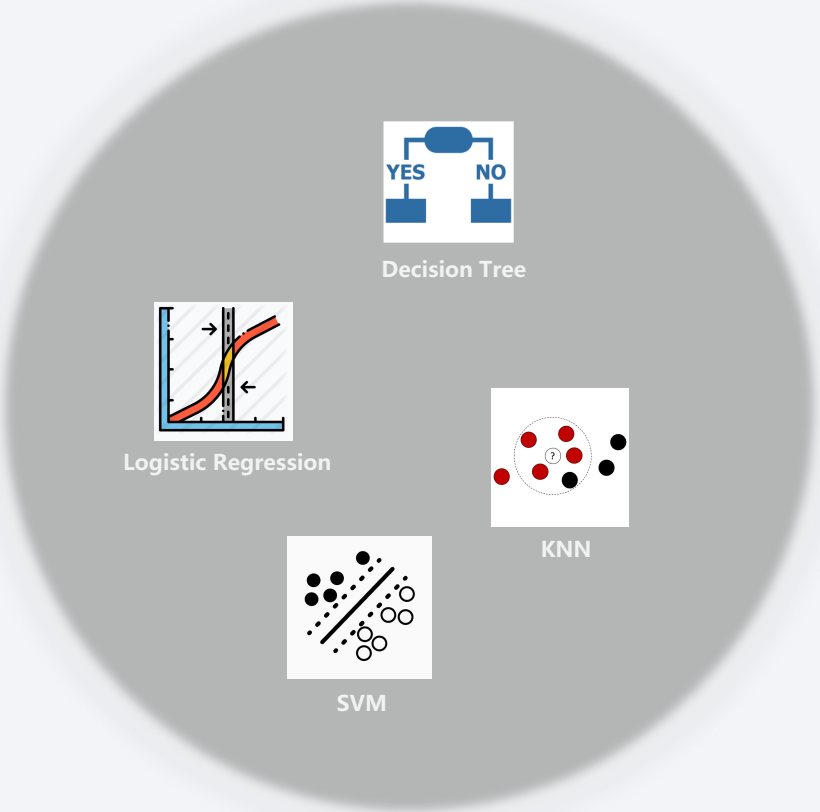


Section 6

Predictive Analysis (Classification)

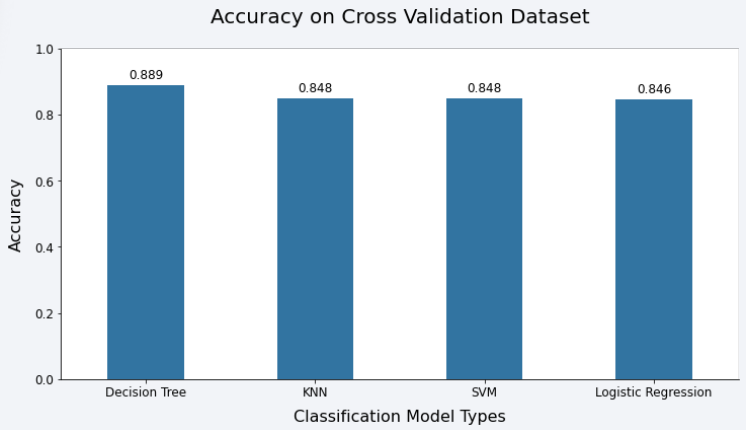
Classification Accuracy

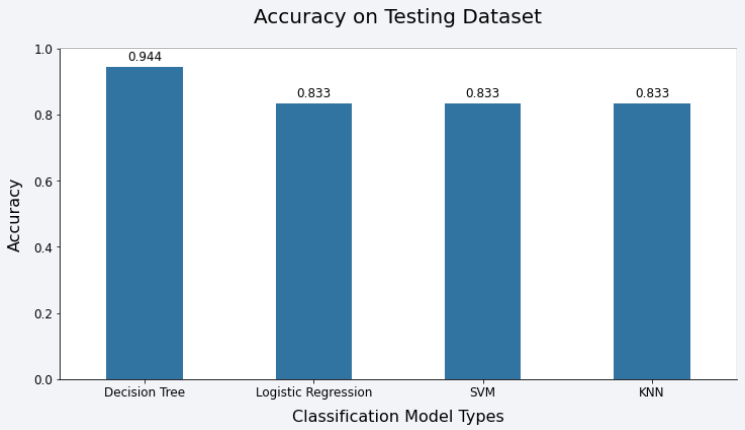
As we can see, among the four classification models, **Decision Tree** stands out in both cross validation score (**0.889**) and testing score (**0.944**). So we can use this trained model to predict the landing outcome of SpaceX Rockets in future.



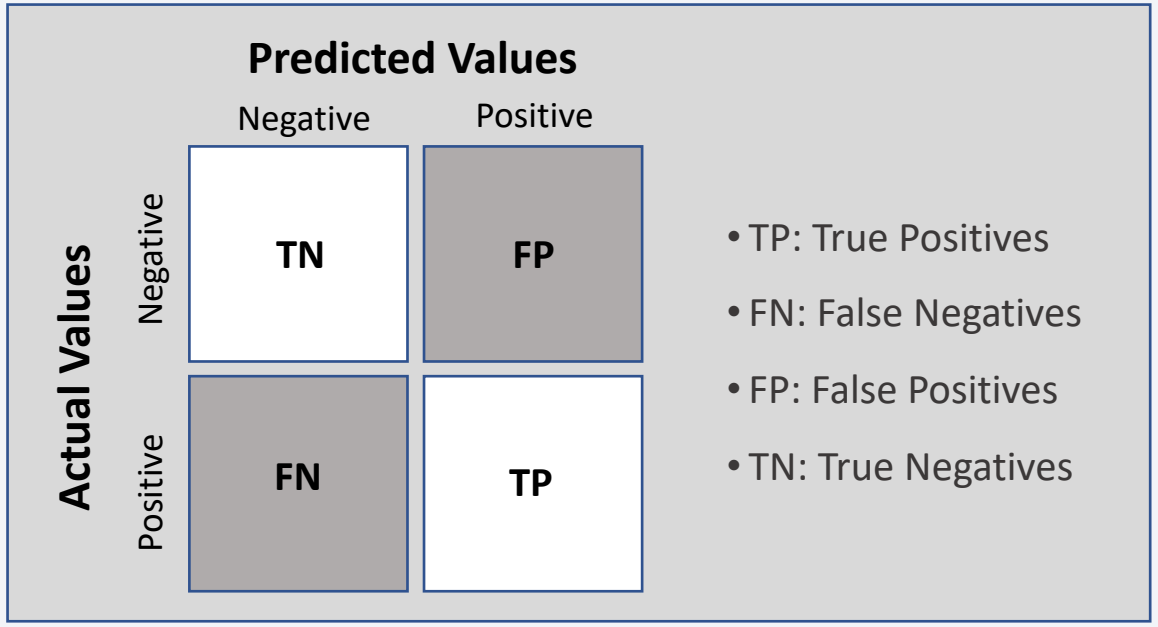
	CV Score	Test Score
Decision Tree	0.889286	0.944444
KNN	0.848214	0.833333
SVM	0.848214	0.833333
Logistic Regression	0.846429	0.833333

```
Decision Tree cv score: 0.889
           test score: 0.944

Model Hyperparameters:
         criterion: entropy
         max_depth: 14
      max_features: sqrt
  min_samples_leaf: 2
 min_samples_split: 10
          splitter: random
```





Confusion Matrix

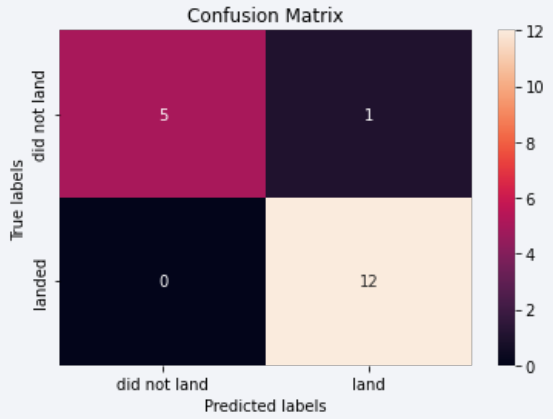


Decision Tree Model

```
acc = accuracy_score(Y_test, yhat)
precision = precision_score(Y_test, yhat)
recall = recall_score(Y_test, yhat)
f1 = f1_score(Y_test, yhat)
```

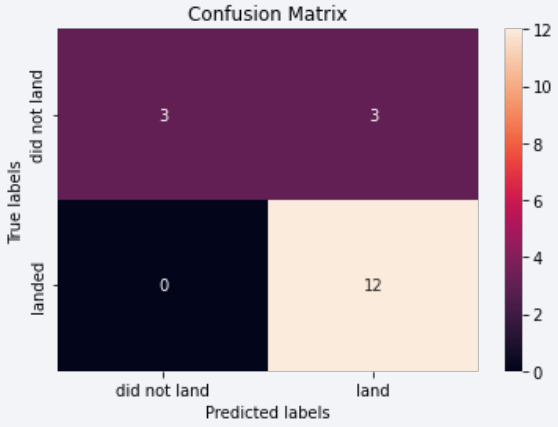
```
Accuracy Score: 0.944
Precision Score: 0.923
Recall Score: 1.000
f1 Score: 0.960
```

Decision Tree



Logistic Regression

SVM



KNN

Conclusions

- All launch sites are *far away from* their neighboring **cities**; but in *close proximity* to **railways** and **coastlines**; relatively *far from* **highways**.
- Orbit type **ES-L1, GEO, HEO, SSO** has the highest first stage successful landing rate.
- **Low weighted** payloads perform **better than** those heavier ones in terms of successful launches.
- Launch site **KSC LC-39A** had the most successful launch counts of all.
- The launch success rate is **increasing by years** since 2013.
- The **Decision Tree classifier** is the best performing model for predicting the first stage landing outcome of SpaceX Rocket.

Thank you!