IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Tommaso Torelli
02/07/2025

Outline

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

Executive Summary

Methodologies

Data were acquired from the SpaceX REST API and supplemented by web-scraping of Wikipedia tables using Python's requests and BeautifulSoup. The raw records were loaded into pandas DataFrames, missing values in PayloadMass were imputed with column means, and targeted feature engineering was applied. Exploratory analyses—including scatter plots, bar charts and time-series trends—were conducted using matplotlib and Plotly, while SQL queries against the SPACEXTABLE surfaced additional insights. Geospatial buffers and distance calculations were implemented in Folium to map launch-site proximities to coastline, rail and road networks. A Plotly Dash application was built to enable interactive filtering of launch outcomes by site and payload mass. Finally, four classification models (Logistic Regression, SVM, K-Nearest Neighbors and Decision Tree) were tuned via GridSearchCV with 10-fold cross-validation and evaluated on a held-out test set.

Results

Coastal launch sites between 25° N and 34° N latitude were confirmed to maximize safety and rotational boost. CCAFS SLC-40's location—870 m from the shoreline, 590–980 m from major transport links and over 19 km from populated areas—was shown to optimize logistical throughput and risk mitigation. Payloads in the 2 000–6 000 kg range launched on Falcon 9 FT and B5 boosters achieved the highest recovery rates, while earlier vehicle variants and sub-1 000 kg missions exhibited increased failure incidence. KSC LC-39A recorded the strongest reliability record, with a 76.9 % success rate (10 of 13). Machine-learning models attained 83.33 % test-set accuracy with zero false negatives, demonstrating the predictive power of the combined mission parameters.

Introduction

Background & Context

SpaceX advertises each Falcon 9 launch at $62 million, whereas competing providers charge upwards of $165 million, largely due to their inability to reuse the first stage. By forecasting whether the booster will land successfully, the true cost of a given launch can be estimated in advance. Such cost estimates are critical for any alternative launch provider seeking to submit competitive bids against SpaceX.

Problems to Be Addressed

1. How can the safety advantages of coastal pad locations be quantified in terms of stage drop zones and booster recovery corridors over open sea?
2. What buffer distances from shoreline, major highways and rail lines—and what exclusion zones around populated areas—strike the optimal balance between logistical efficiency and public safety?
3. What relationships exist between payload mass, booster variant and landing outcome, and which combinations maximize touchdown reliability?
4. How accurately can machine-learning classifiers predict first-stage landing success when trained on a comprehensive set of mission parameters?

Section 1

Methodology

Methodology

Executive Summary

Data were acquired via the SpaceX REST API and supplemented by web-scraping of Wikipedia tables using Python's requests and BeautifulSoup. The raw information was then loaded into pandas DataFrames (with NumPy), missing values were imputed with column mean, and targeted feature engineering was applied. Exploratory data analysis was performed using both visualizations and SQL queries to surface key patterns, and interactive dashboards were developed with Folium for geospatial mapping and Plotly Dash for dynamic charts. Finally, multiple classification models including Logistic Regression, SVM, K-Nearest Neighbors, and Decision Tree were built, with the data normalized and split via scikit-learn. Hyperparameters were tuned through GridSearchCV with 10-fold cross-validation, and performance was evaluated by accuracy scores and confusion matrices.

Data Collection

SpaceX API

- Use of request library methods get(), json(), json_normalize() to retrieve data of the Falcon 9 launches from SpaceX API

WebScraping

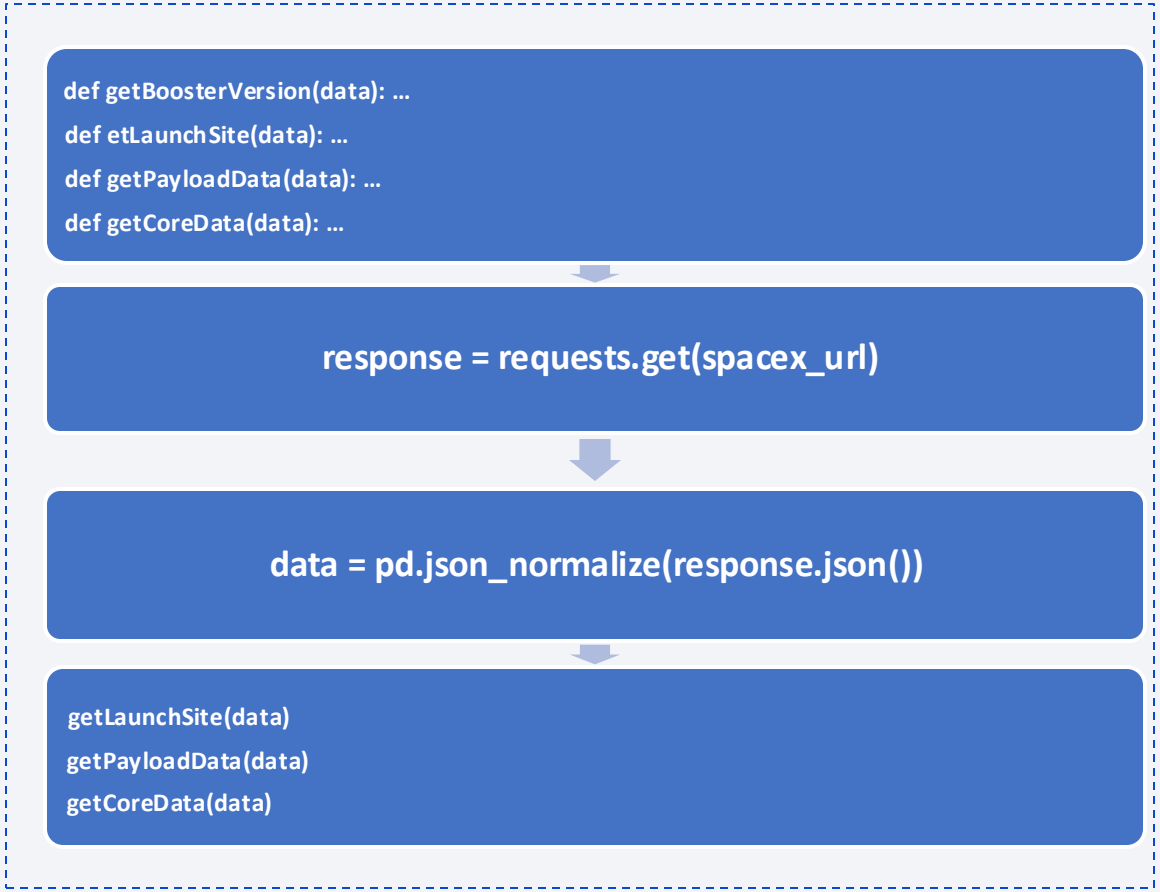
- Use of request and BeautifulSoup library methods get(), find(), find_all() to perform WebScraping and retrieve data from Falcon 9 launches Wikipedia webpage

Data Collection – SpaceX API

1. Define Auxiliary Functions to help retrieve data from the API
2. Request rocket launch data from SpaceX API (spacex_url)
3. Extract the json content as a pd dataframe
4. Use the Auxiliary Functions to get the columns of interest and create the dataframe of the Falcon 9 launches

- GitHub URL

SpaceX API Flowchart



Data Collection - Scraping

1. Define Auxiliary Functions to help retrieve data from the Wikipedia HTML table
2. Request Falcon 9 rocket launch data from Wikipidia
3. Extract the launch content of the third table in the html through BeautifulSoup
4. Use the Auxiliary Function to iterate through the `<th>` elements and extract column names
5. Use the Auxiliary Functions to iterate through the table rows and create the dataframe containing the Falcon 9 data

- GitHub URL

Scraping Flowchart

Data Wrangling

- Pandas and numpy libraries were used to retain only data from Falcon 9 launches, replace PayloadMass missing values with PayloadMass mean values and Create a Landing Outcome 0/1 “Class” label (0 = not landed 1 = landed)
- Add the GitHub

Wrangling Flowchart

1. Retain Falcon 9

```
data_falcon9 =
data[data['BoosterVersion']!='Falcon 1']
```

2. Handle Missing values

```
PayloadMean = data_falcon9['PayloadMass'].mean()

data_falcon9.replace({'PayloadMass': np.nan},
PayloadMean, inplace=True)
```

3. Feature Engineering

```
landing_class = []
for val in df.Outcome.tolist():
        if val in bad_outcomes:
         landing_class.append(0)
        else:
         landing_class.append(1)
```

EDA with Data Visualization

- In the EDA phase, we focused on identifying relationships between the dataset's variables and the success outcome (0 = not landed; 1 = landed). Accordingly, we used scatter plots and bar charts (fig1.A/1.B) to explore success-rate relationships within variables, and line chart (fig 1.C) to examine success-rate trends over the years.
- Add the GitHub URL

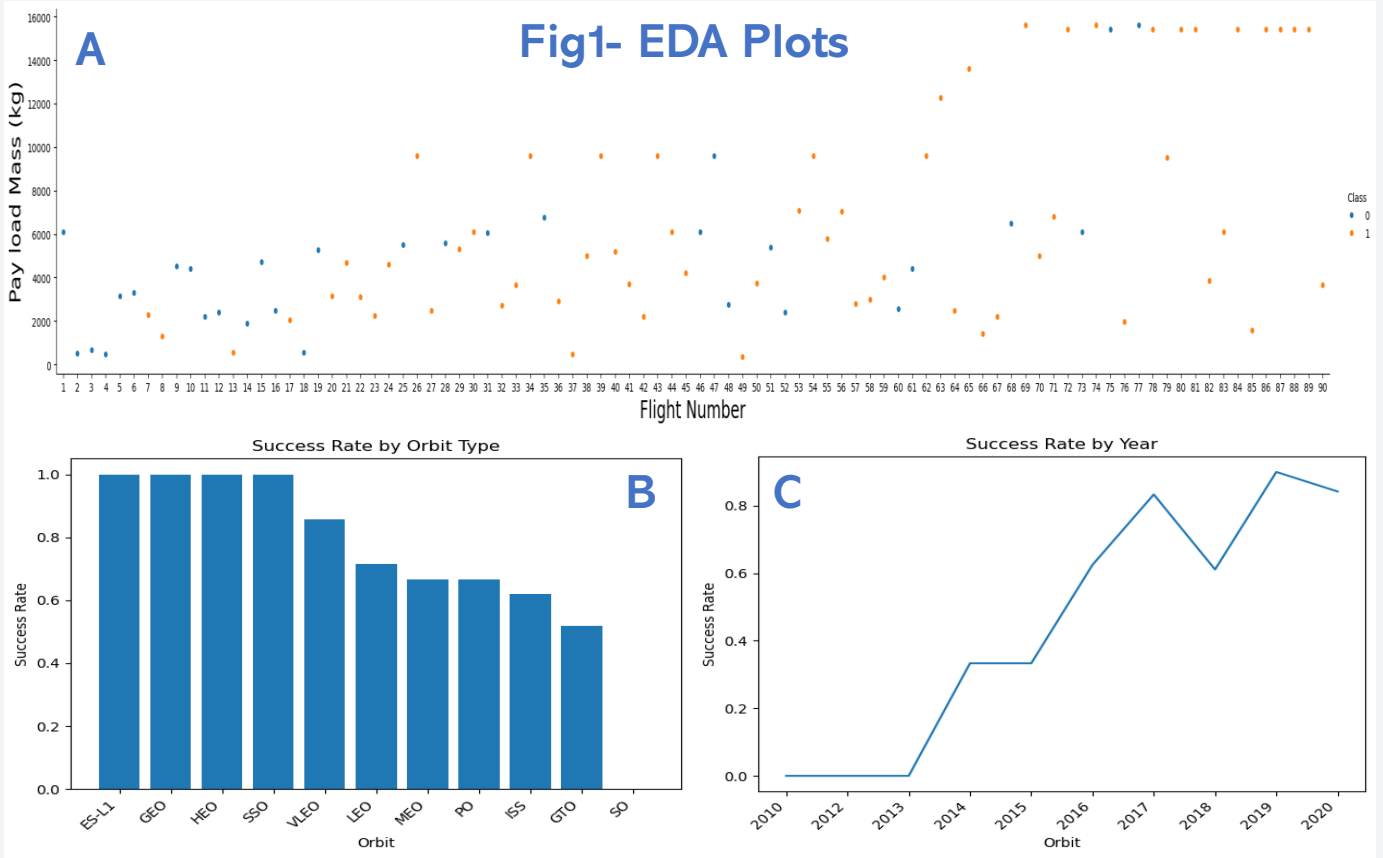


Fig1.A) Scatterplot highlighting relationships between FlightNumber, Payload Mass overlay and outcome. Fig1.B) Bar Chart depicting success-rate for each orbit type. Fig1.C) Line chart with success-rate trend over the years (2010 -2020)

EDA with SQL

Query 1 to 5

1. **Display the names of the unique launch sites in the space mission:**
   - select distinct Launch_Site from SPACEXTABLE

2. **Display 5 records where launch sites begin with the string 'CCA'**
   - select * from SPACEXTABLE where Launch_site Like "CCA%" limit 5

3. **Display the total payload mass carried by boosters launched by NASA (CRS)**
   - select sum(PAYLOAD_MASS__KG_) from SPACEXTABLE where Customer = "NASA (CRS)"

4. **Display average payload mass carried by booster version F9 v1.1**
   - select avg(PAYLOAD_MASS__KG_) from SPACEXTABLE where Booster_Version = "F9 v1.1"

5. **List the date when the first succesful landing outcome in ground pad was acheived.**
   - select min(Date) from SPACEXTABLE where Landing_Outcome = "Success (ground pad)"

EDA with SQL

Query 6 to 10

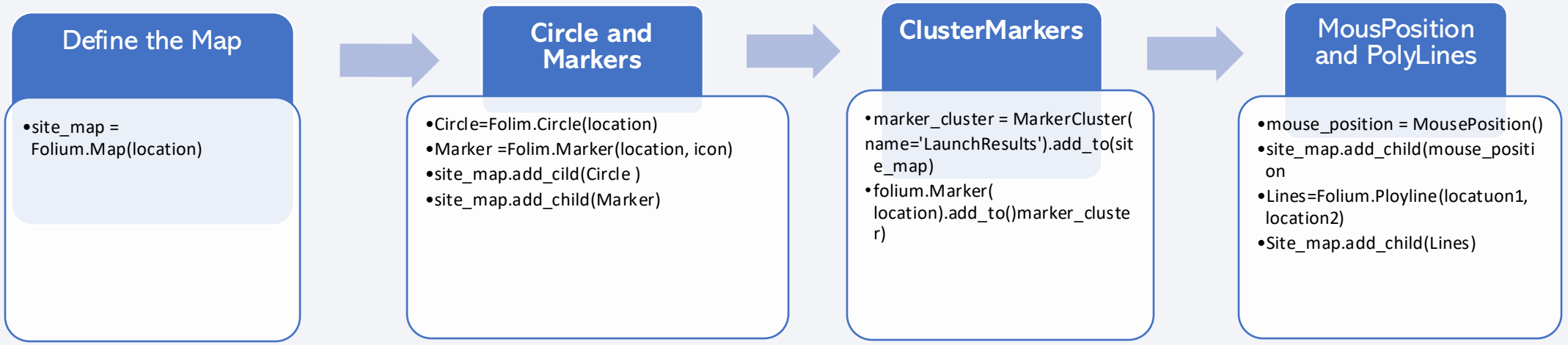
6. **List the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000**
    - SELECT Payload FROM SPACEXTABLE WHERE Landing_Outcome = 'Success (drone ship)' AND PAYLOAD_MASS__KG_ > 4000 AND PAYLOAD_MASS__KG_ < 6000
7. **List the total number of successful and failure mission outcomes**
    - SELECT Mission_Outcome, COUNT(*) AS Totale FROM SPACEXTABLE WHERE Mission_Outcome LIKE 'Success%' OR Mission_Outcome LIKE 'Failure%' GROUP BY Mission_Outcome;
8. **List all the booster_versions that have carried the maximum payload mass. Use a subquery.**
    - SELECT distinct Booster_Version FROM SPACEXTABLE WHERE PAYLOAD_MASS__KG_ = (SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTABLE);
9. **List the records which will display the month names, failure landing_outcomes in drone ship ,booster versions, launch_site for the months in year 2015.**
    - select substr(Date, 6,2) as Month,substr(Date, 0,5) as Year, Landing_Outcome, Booster_Version, Launch_Site from spacextable where landing_outcome = "Failure (drone ship)» and date Like "2015%"
10. **Rank the count of landing outcomes between the date 2010-06-04 and 2017-03-20, in descending order.**
    - SELECT Landing_Outcome, COUNT(*) AS Total FROM SPACEXTABLE WHERE Date BETWEEN '2010-06-04' AND '2017-03-20' GROUP BY Landing_Outcome Order by Total desc;

- Add the GitHub URL

Build an Interactive Map with Folium

The Folium library was used to mark launch-site locations in the USA. First, a Map object was created, and then Circle and Marker objects were added to pinpoint each site and display its name. Next, a MarkerCluster object was introduced to show how many launches (and how many successful launches) occurred at each location. Finally, a MousePosition object was added to retrieve coordinates directly from the map, and PolyLine objects were drawn to illustrate the distances from the CCAFS-SLC-40 launch site to the nearest coast, railway, highway, and city.

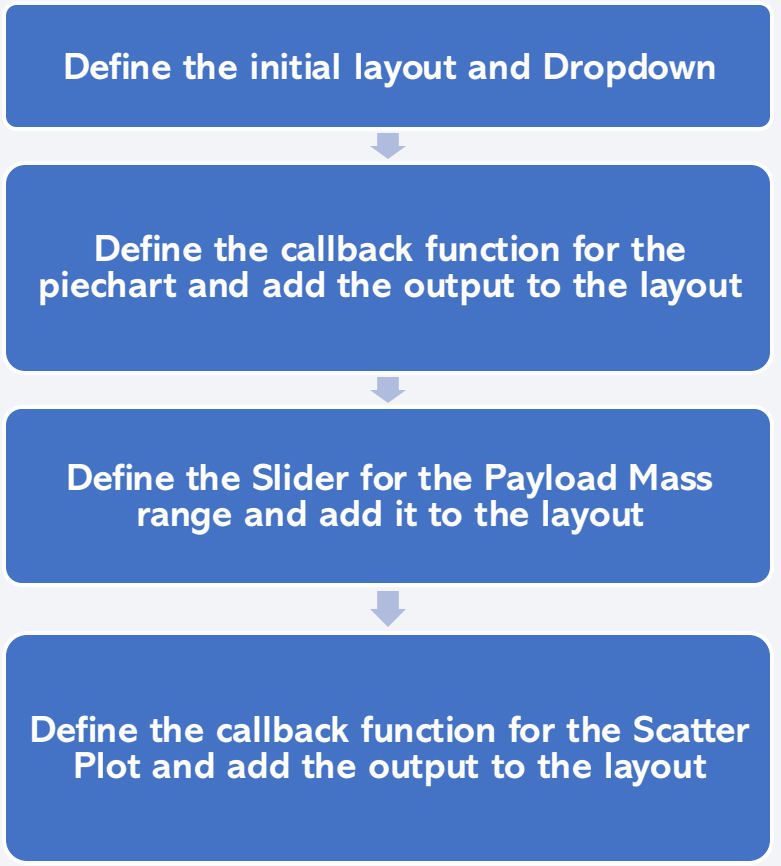
- GithubUrl



Build a Dashboard with Plotly Dash

The dashboard was built with Dash for its interactive framework and Plotly Express for charting. First, the app layout was defined with a Dropdown component to select either all launch sites or a single site, a RangeSlider to filter payload mass, and two Graph placeholders for our visualizations. Then, Plotly Express pie and scatter figures were created; one to show total successes (or success-vs-failure for a chosen site) and the other to plot payload weight against launch outcome, coloring points by booster category. Next, two Dash callbacks were wired up: the first listens to the Dropdown and updates the pie chart accordingly, while the second takes both Dropdown and RangeSlider inputs to regenerate the scatter plot based on site selection and payload range. Finally, these interactive controls let users seamlessly compare site performance, probe how payload mass affects success rates, and spot booster-specific patterns.

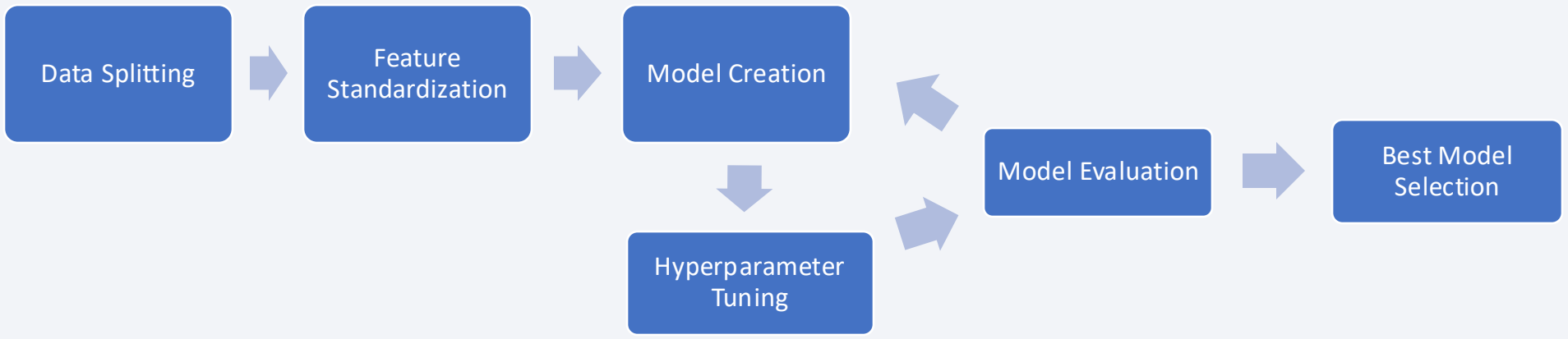
Add the GitHub URL of your completed Plotly Dash lab



Predictive Analysis (Classification)

The classification pipeline unfolded in six methodical stages. First, the full dataset was split into training (80 %) and test (20 %) subsets, and all features were standardized via scikit-learn's StandardScaler. Next, four candidate algorithms (Logistic Regression, Support Vector Machine, K-Nearest Neighbors, and Decision Tree) were instantiated, each paired with a tailored hyperparameter grid. Hyperparameter tuning was then performed using GridSearchCV with 10-fold cross-validation, producing four optimized models. These tuned models were evaluated on the held-out test set: accuracy scores were computed and confusion matrices plotted to assess both overall and per-class performance. Finally, the classifier achieving the highest test-set accuracy was declared the best performer.

Predictive Analysis Flowchart



Results

Launches to ES-L1, GEO, HEO and SSO orbits achieve the highest success rate (although only SSO has seen more than one flight attempt) and, excluding GTO missions, the probability of a successful first-stage landing increases with flight experience (i.e. higher flight numbers) and peaks for payloads of 2.000 – 4.000 kg and 4.500 – 5.500 kg. Launches employing the Falcon 9 FT booster also exhibit one of the highest success rates.

Of all launch sites, KSC LC-39A recorded the best track record, with a 76.9 % (Fig2.A) success rate (10 out of 13 launches)(Fig2.B), and overall mission success steadily improved from 2013 through 2017, with only 2014 – 2015 showing a temporary plateau.

Finally, the four machine-learning models employed (KNN, SVM, Decision Tree and Logistic Regression) performed almost identically, each achieving 83 % accuracy with three false positives.

Fig2- Dash–Folium Plots

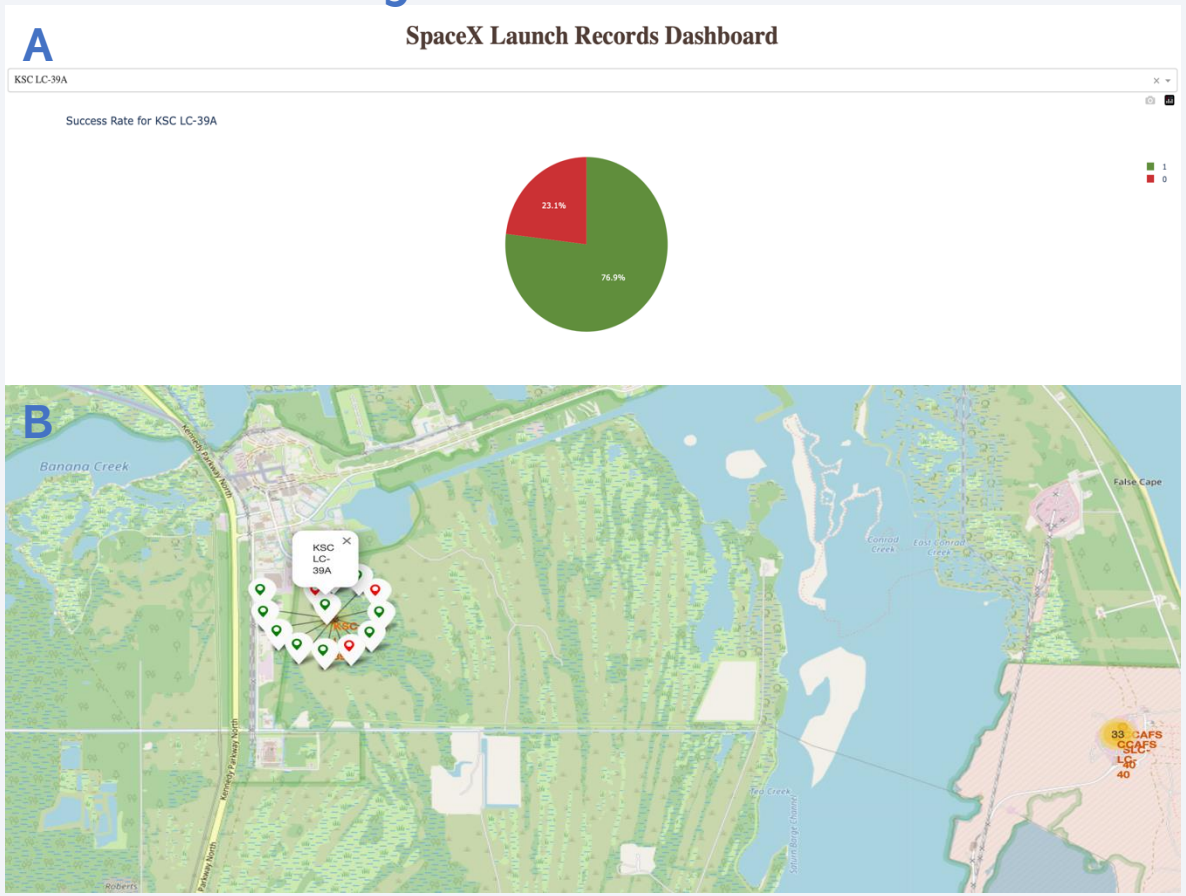
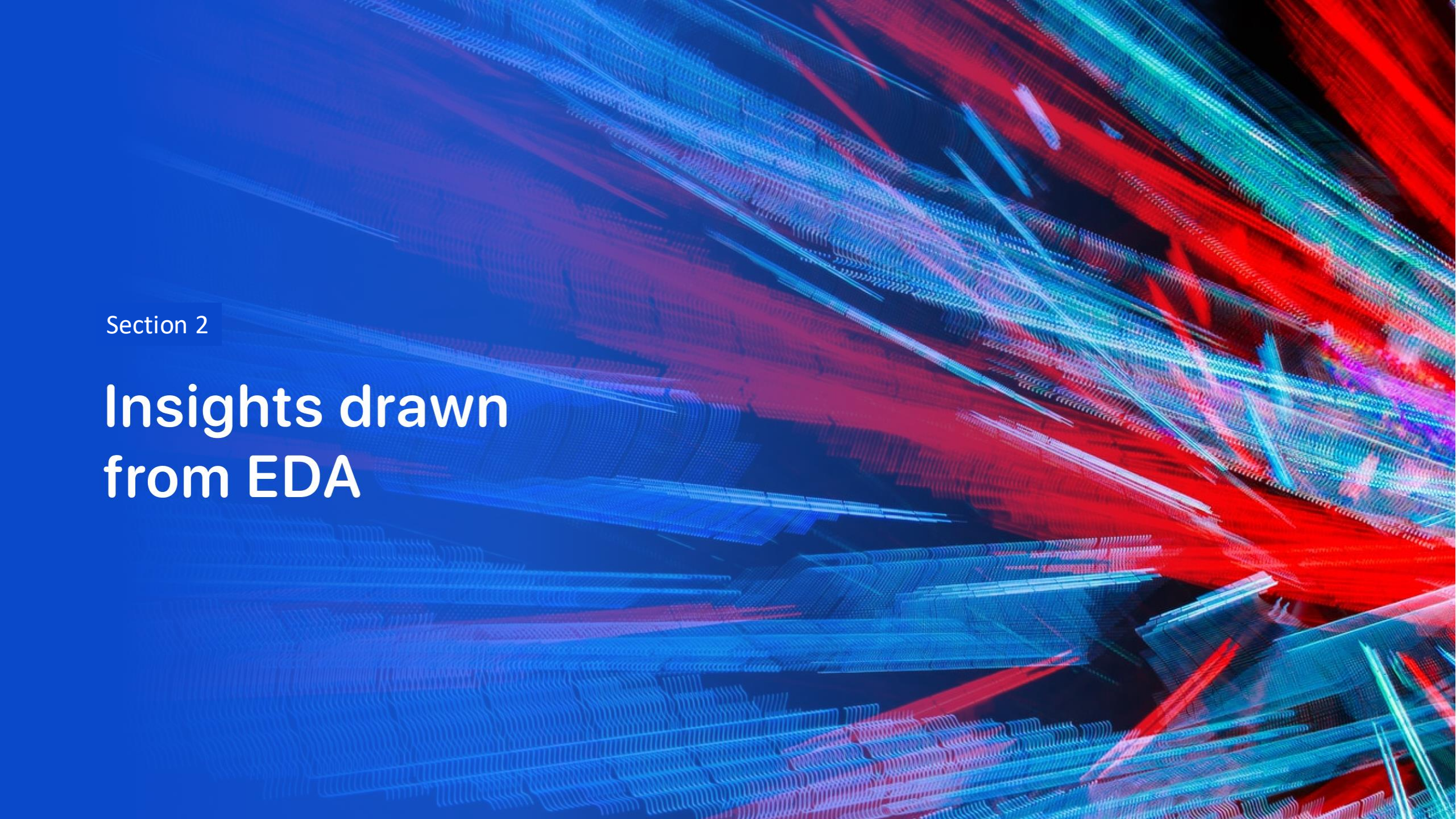
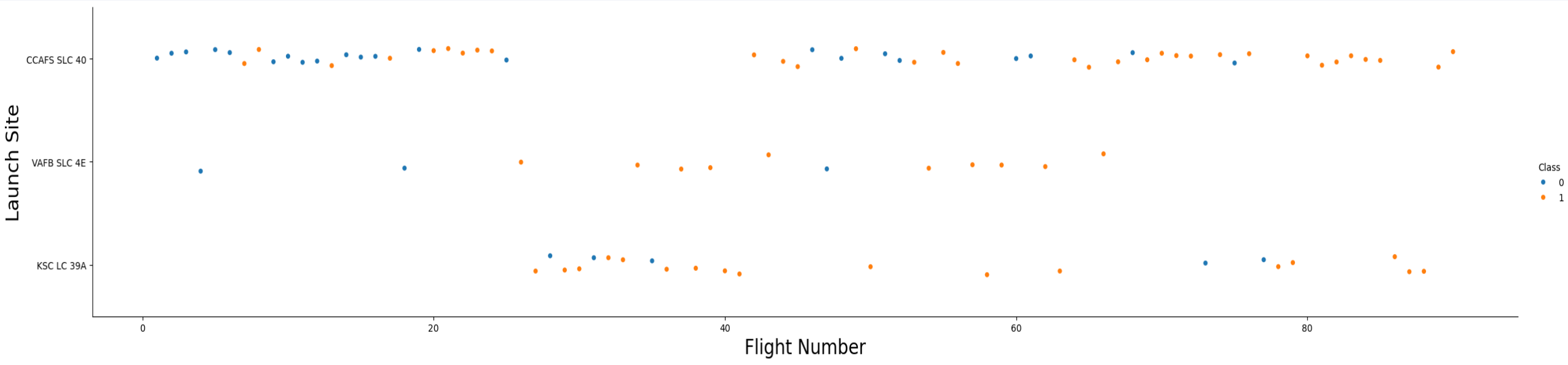


Fig2.A) interactive dash app highlighting KSC LC-39A launch site success rate . Fig2.B) Folium map showing n° of launches and success-rate for launch site.


Section 2
Insights drawn from EDA

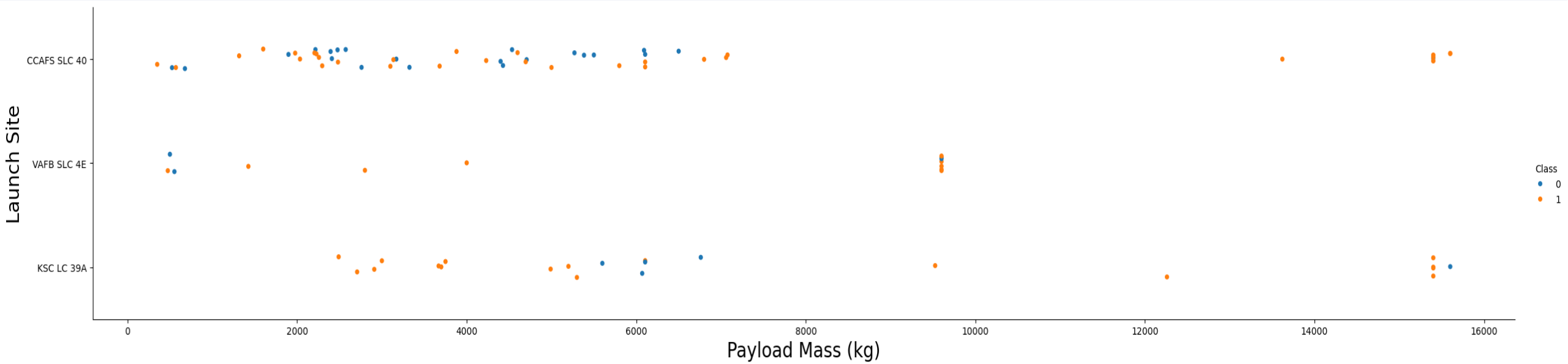
Flight Number vs. Launch Site

This scatter plot shows individual Falcon flight numbers on the x-axis and their three launch sites (KSC LC 39A, VAFB SLC 4E, CCAFS SLC 40) on the y-axis. Each point is colored by landing outcome—blue for failed landings (0) and orange for successful landings (1). Early flights across all sites exhibit a mix of failures and successes, but as flight numbers increase, orange points dominate, reflecting SpaceX's steep learning curve and the maturation of its landing procedures. The result is a clear trend toward landing success in later missions.



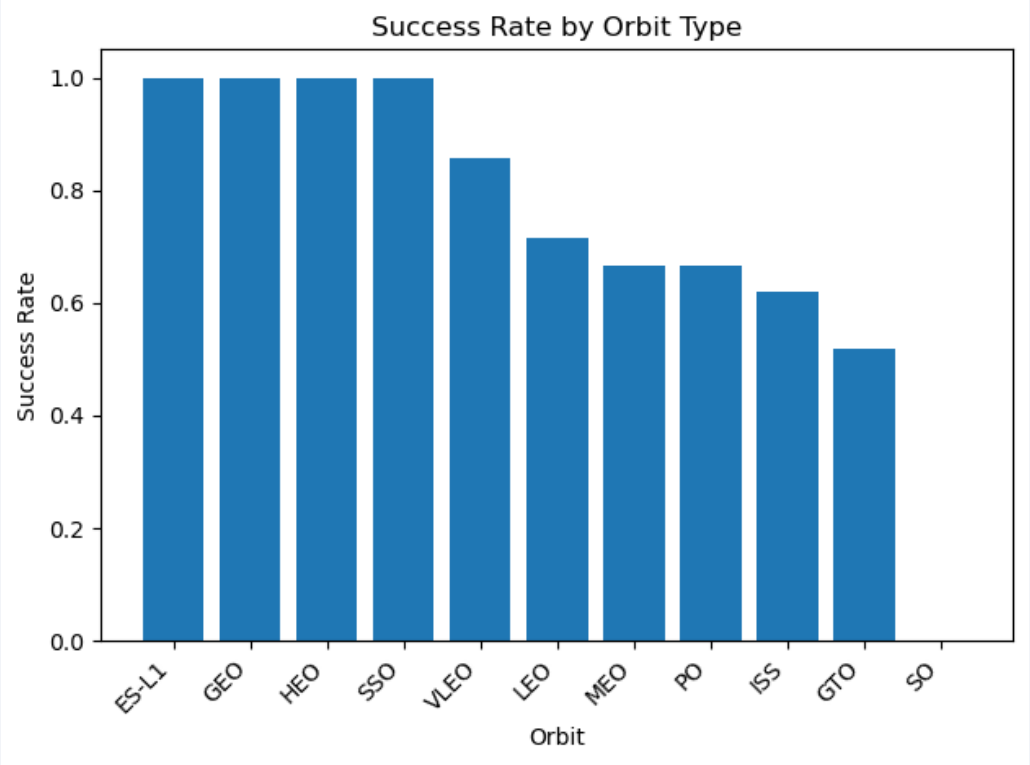
Payload vs. Launch Site

This scatter plot maps each mission's payload mass (kg) on the x-axis against its launch site on the y-axis (KSC LC 39A, VAFB SLC 4E, CCAFS SLC 40), coloring points blue for landing failures (0) and orange for successes (1). Most of the orange markers are clustered at higher masses (particularly in the 2.000–6.000kg range) yet blue dots never disappear entirely. This shows that landing outcomes, while generally more reliable at larger payloads, are not strictly determined by payload mass.



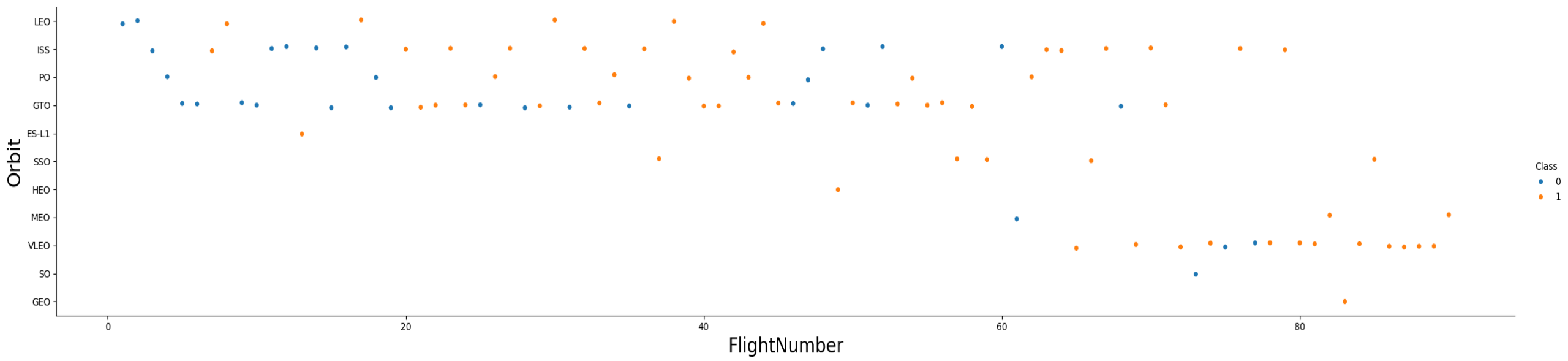
Success Rate vs. Orbit Type

This bar chart compares landing success rates across different orbit types (ES-L1, GEO, HEO, SSO, VLEO, LEO, MEO, PO, ISS, GTO, SO). Each bar's height shows the fraction of successful landings for missions targeting that orbit. However, only Sun-synchronous orbit (SSO) actually has recorded flights hence its 100% success rate is the only meaningful statistic. All other orbit types have zero launches in the dataset, so their displayed success rates merely reflect the absence of data rather than operational performance.



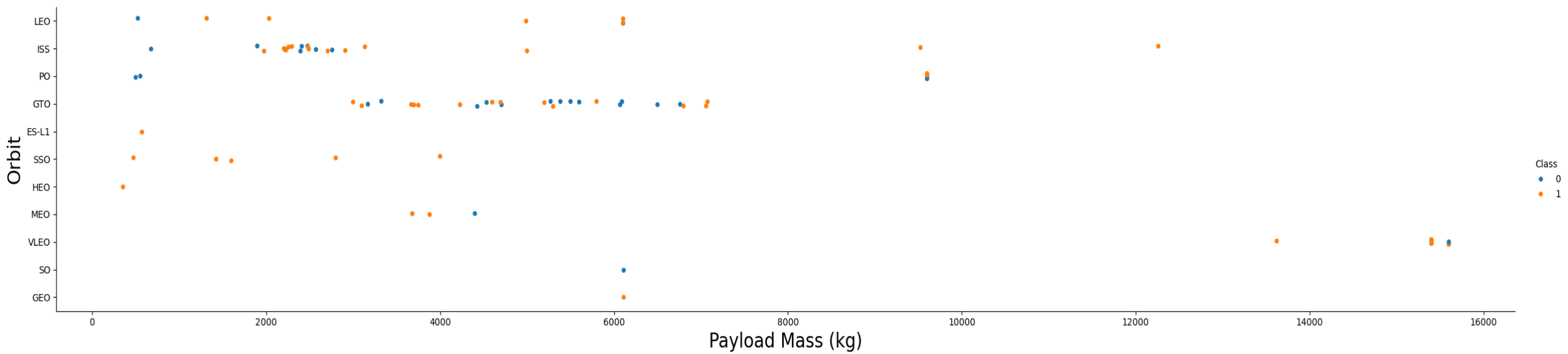
Flight Number vs. Orbit Type

This scatter plot shows each mission's flight number on the x-axis and its target orbit on the y-axis (with blue dots marking landing failures and orange dots marking successes). Early flights (below ~20) were concentrated in LEO and ISS missions and feature a mix of outcomes, but as flight numbers climb, SpaceX expands into higher and more diverse orbits (PO, GTO, ES-L1, SSO, HEO, MEO, VLEO, SO, GEO) and the orange dots begin to dominate. Yet the occasional blue marker even in late-career, complex missions underlines that recovery remains probabilistic across all orbit types (despite a clear overall improvement in landing reliability).



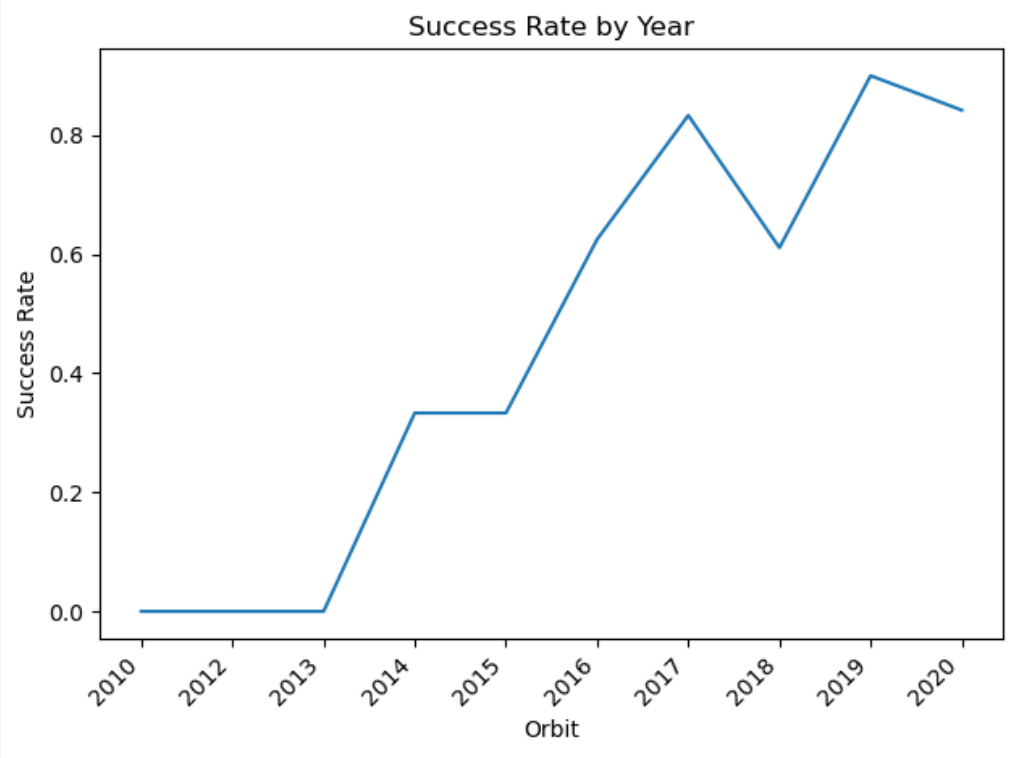
Payload vs. Orbit Type

This scatter plot maps each mission's payload mass (kg) on the x-axis and its target orbit on the y-axis (LEO, ISS, PO, GTO, ES-L1, SSO, HEO, MEO, VLEO, SO, GEO), with blue points for landing failures (0) and orange points for successes (1). Lower-mass missions (under ~2.000 kg) to LEO and ISS show a mix of blue and orange outcomes (reflecting early experimentation), while as payloads grow operations extend into higher orbits (GTO, ES-L1, SSO, VLEO, GEO) and orange markers become more common at masses above ~5.000 kg (though blue failures still occur). This suggests that heavier-lift flights tend to recover more reliably (despite no absolute payload threshold).



Launch Success Yearly Trend

This line chart shows the annual landing success rate from 2010 to 2020 (x-axis: year; y-axis: fraction of successful landings). From 2010 through 2013 the rate sits at 0 (no successful recoveries), then jumps to ~30 % in 2014–2015 (as landing attempts begin), climbs to ~60 % in 2016 and ~80 % in 2017 (reflecting rapid learning), dips back to ~60 % in 2018 (a temporary setback), peaks near 90 % in 2019 and settles around ~85 % in 2020 (indicating sustained high reliability).



All Launch Site Names

- Launch sites:
    1. CCAFS LC-40
    2. VAFB SLC-4E
    3. KSC LC-39A
    4. CCAFS SLC-40.
- They were obtained by using SELECT (to specify the Launch_Site column), DISTINCT (to remove duplicate entries), and FROM SPACEXTABLE (to designate the source table).

Launch Site Names Begin with 'CCA'

The first five records with Launch_Sit e starting with “CCA” were retrieved by using SELECT * (to return all columns), WHERE Launch_Site LIKE 'CCA%' (to filter sites beginning with “CCA”), and LIMIT 5 (to restrict the output to five rows).

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	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
2012-10-08	0:35:00	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

- The total payload mass carried by boosters launched for NASA (CRS) is 45.596 kg.
- It was obtained by using SELECT sum(PAYLOAD_MASS__KG_) (to aggregate the payload mass), FROM SPACEXTABLE (to designate the source table), and WHERE Customer = 'NASA (CRS)' (to filter only NASA CRS missions).

Average Payload Mass by F9 v1.1

- The average payload mass carried by booster version F9 v1.1 is 2.928,4 kg.
- It was obtained by using SELECT avg(PAYLOAD_MASS__KG_) (to calculate the average payload mass), FROM SPACEXTABLE (to designate the source table), and WHERE Booster_Version = 'F9 v1.1' (to filter for that specific booster version).

First Successful Ground Landing Date

- The first successful ground pad landing date is 2015-12-22.
- It was obtained by using SELECT min(Date) (to find the earliest date), FROM SPACEXTABLE (to designate the source table), and WHERE Landing_Outcome = 'Success (ground pad)' (to filter only ground pad successes).

Successful Drone Ship Landing with Payload between 4000 and 6000

- Payloads that achieved successful drone ship landings with masses between 4.000 kg and 6.000 kg:
    1. JCSAT-14
    2. JCSAT-16
    3. SES-10
    4. SES-11 / EchoStar 105.
- They were obtained by using SELECT Payload (to specify the payload names), FROM SPACEXTABLE (to designate the source table), WHERE Landing_Outcome = 'Success (drone ship)' (to filter for drone ship recoveries), AND PAYLOAD_MASS__KG_ > 4000 (to set the lower mass bound), AND PAYLOAD_MASS__KG_ < 6000 (to set the upper mass bound).

Total Number of Successful and Failure Mission Outcomes

- The total mission outcome counts:
    1. Failure (in flight) = 1
    2. Success = 99
    3. Success (payload status unclear) = 1.
- They were obtained by using SELECT Mission_Outcome (to specify the outcome column), COUNT(*) AS Totale (to count each outcome), FROM SPACEXTABLE (to designate the data source), WHERE Mission_Outcome LIKE 'Success%' OR Mission_Outcome LIKE 'Failure%' (to filter for success and failure outcomes), and GROUP BY Mission_Outcome (to aggregate by outcome).

Boosters Carried Maximum Payload

- Booster versions that have carried the maximum payload mass:
  - 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
- Were obtained by using SELECT DISTINCT Booster_Version (to specify the booster version column), FROM SPACEXTABLE (to designate the source table), and WHERE PAYLOAD_MASS__KG_ = (SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTABLE) (to filter only those records matching the maximum payload mass via a subquery).

2015 Launch Records

2015 faiiled launch records were obtained by using SELECT substr(Date,6,2) as Month and substr(Date,0,5) as Year (to extract the month and year), along with Landing_Outcome, Booster_Version and Launch_Site, FROM SPACEXTABLE (to designate the data source), WHERE Landing_Outcome = 'Failure (drone ship)' (to filter for drone-ship failures) and Date LIKE '2015%' (to restrict to the year 2015).

Month	Year	Landing_Outcome	Booster_Version	Launch_Site
01	2015	Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40
04	2015	Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40

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

The landing outcomes and their counts between 2010-06-04 and 2017-03-20 were obtained by using SELECT Landing_Outcome, COUNT(*) AS Total (to count each outcome), FROM SPACEXTABLE (to designate the source table), WHERE Date BETWEEN '2010-06-04' AND '2017-03-20' (to filter the date range), GROUP BY Landing_Outcome (to aggregate by outcome), and ORDER BY Total DESC (to rank results in descending order).

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

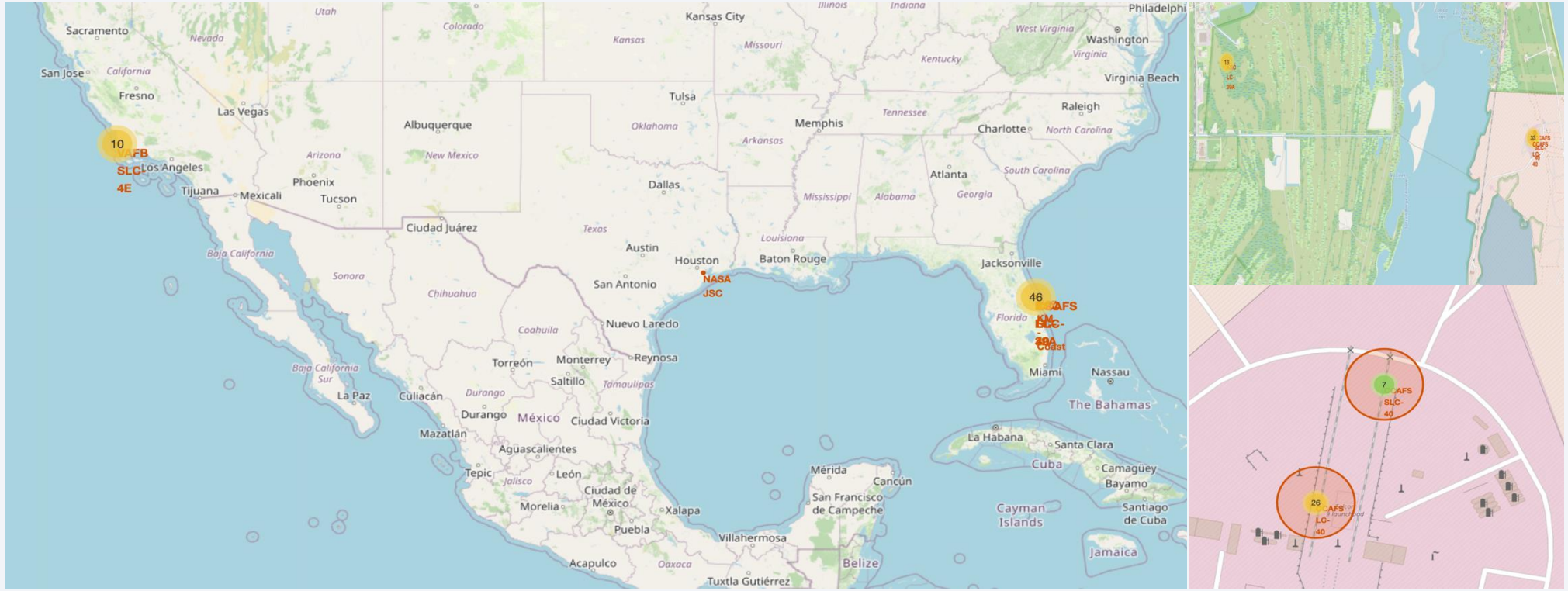
Section 3

Launch Sites Proximities Analysis

Launch sites locations

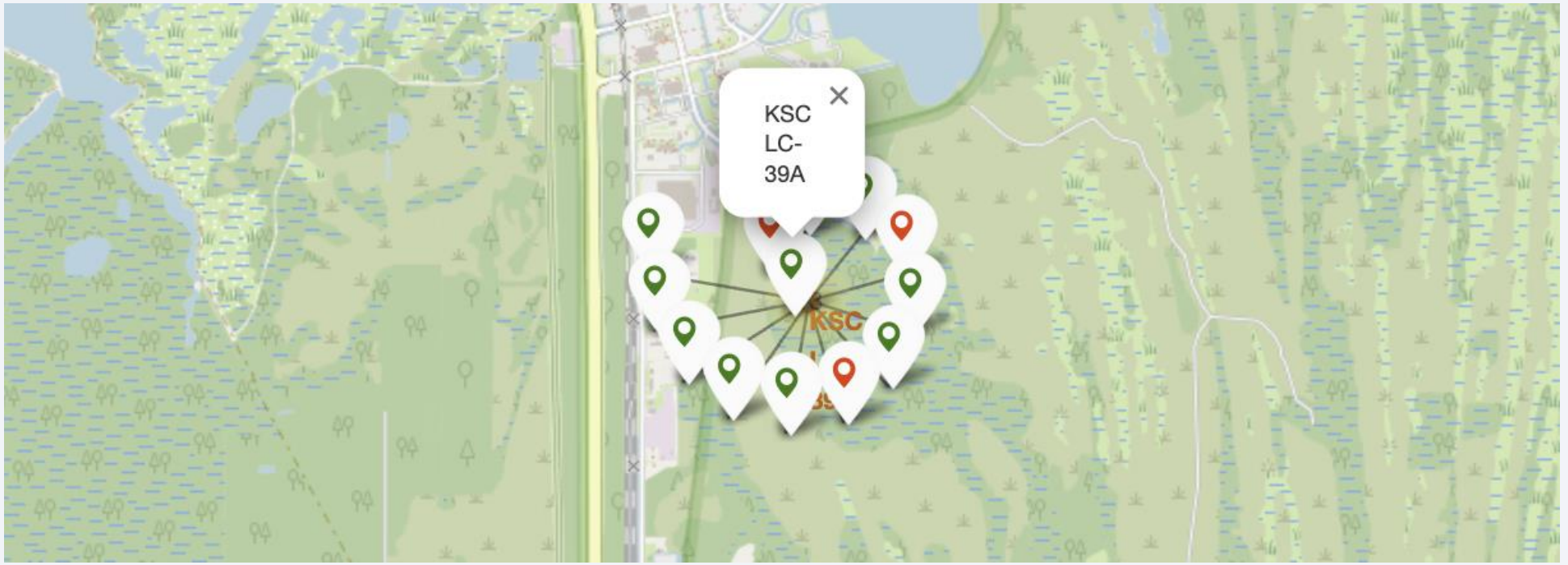
All locations of the launch sites that have been used in the Spacex mission are located well above of the equator line and along the coast of the United States. VAFB SLC-4E (10 launches) is in California while KSC LC-39A (13 launches) CCAFS LC-40 (26 launches) and CCAFS LCS-40 (7 launches) are in Florida.

Coastal launch sites are likely chosen so spent stages safely fall into the ocean (with boosters recovered at sea). Moreover, sites are as close to the equator as possible to maximize the extra launch velocity from Earth's rotation.



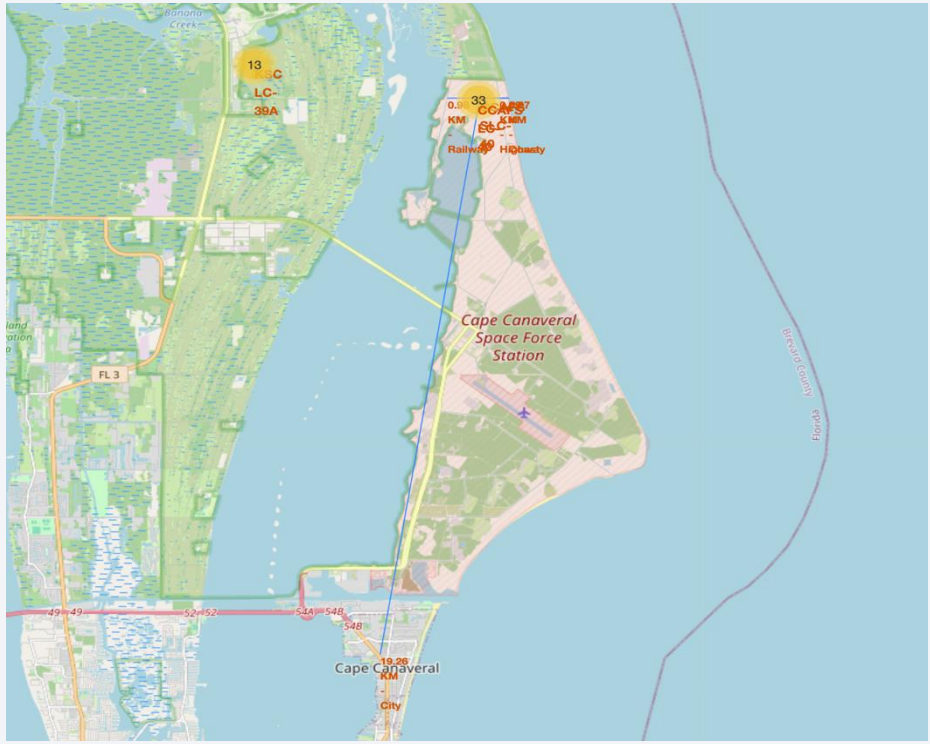
Launch Outcomes

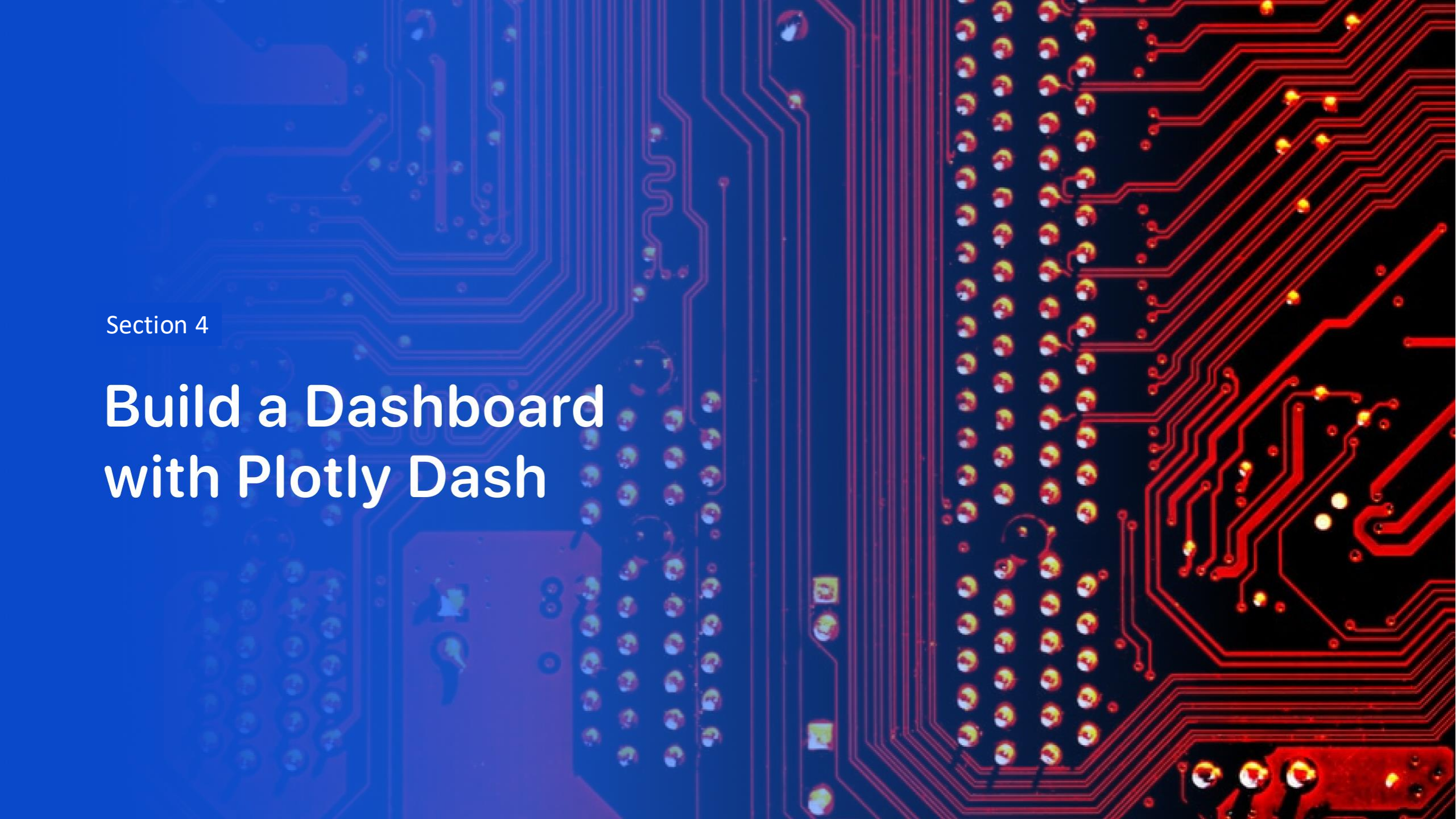
From the color coded launch outcomes marked on the map it is possible to observe both the total number of launches and the success/failure outcome (green = success, red = fail) of each specific launch site. KSC LC-39A, on Merritt Island, Florida, has the highest success rate of any SpaceX launch site, with 10 successful launches out of 13.



Launch sites proximities

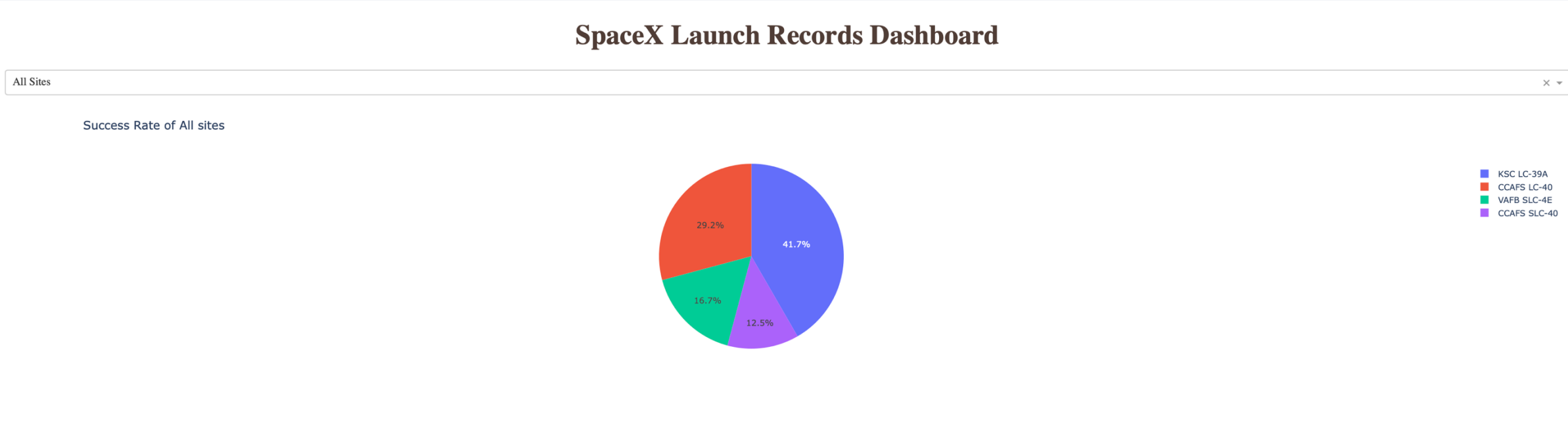
The CCAFS SLC-40 launch site, situated 870m from the coastline, 590m from the highway and 980m from the railway but more than 19.26km from the nearest urban center (Cape Canaveral), exemplifies a strategic location that facilitates logistical support (located near highway and railway) and booster recovery (located near the coast) while minimizing risk to populated areas.




Section 4
Build a Dashboard with Plotly Dash

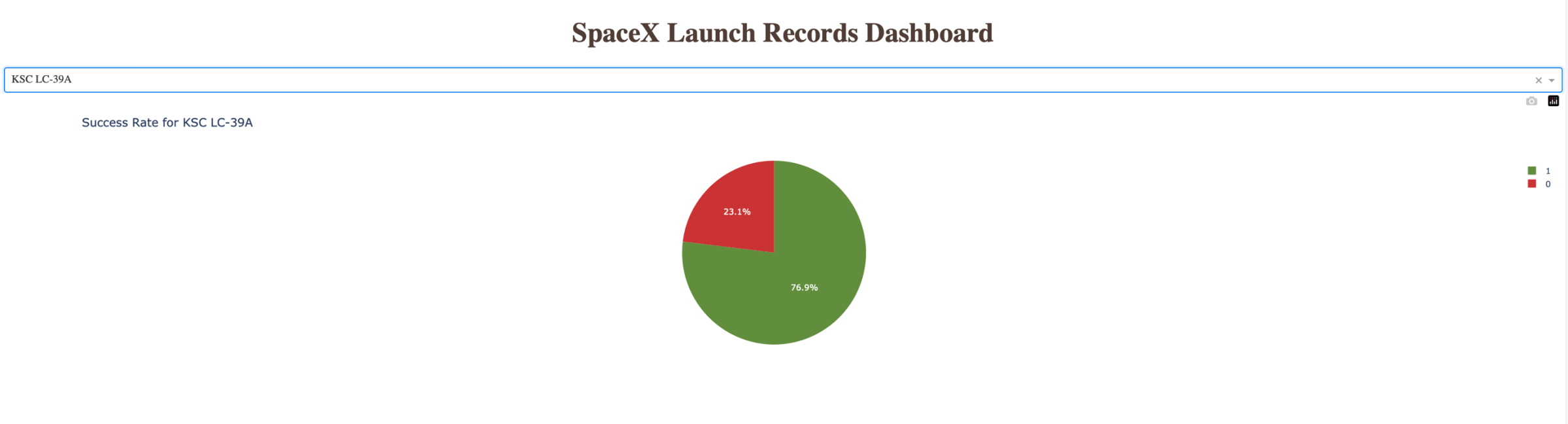
Distribution of Successful Launches by Site

The pie chart below illustrates each launch site's share of total successful missions. Kennedy Space Center's LC-39A leads with 41.7% of all successes, followed by CCAFS LC-40 (29.2%), VAFB SLC-4E (16.7%) and CCAFS SLC-40 (12.5%).



KSC LC-39A Launch Success Rate

With 10 successful missions out of 13 attempts (76.9%), KSC LC-39A demonstrates a strong reliability record, underscoring its pivotal role in SpaceX program.



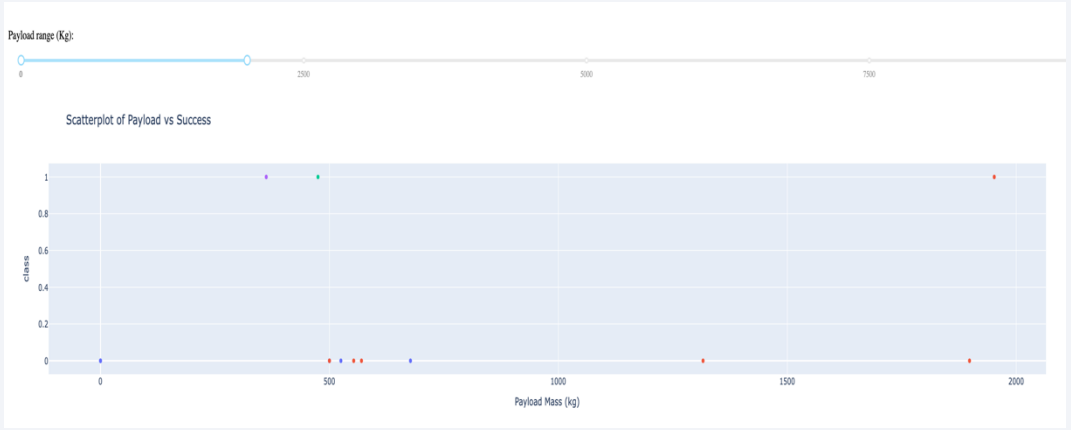
Launch Success by Payload Mass and Booster Version

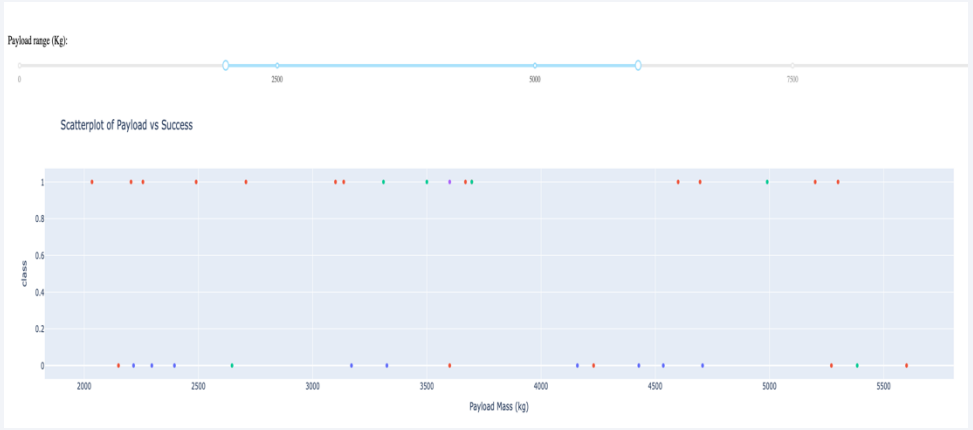
Highest Success Rates

- B5 (purple) and FT (green) boosters dominate the successful-mission cluster (class = 1) across a wide payload band (roughly 2.000–6.000 kg).
- The single heaviest payload (~10.000 kg) was also a B4 flight (light purple).

Payload-Dependent Risk

- Under ~1.000 kg there are a handful of failures across v1.1 and FT, suggesting issues with smaller payloads.
- Above ~4.000 kg, nearly every mission succeeded, indicating handling medium-to-heavy lifts reliably.



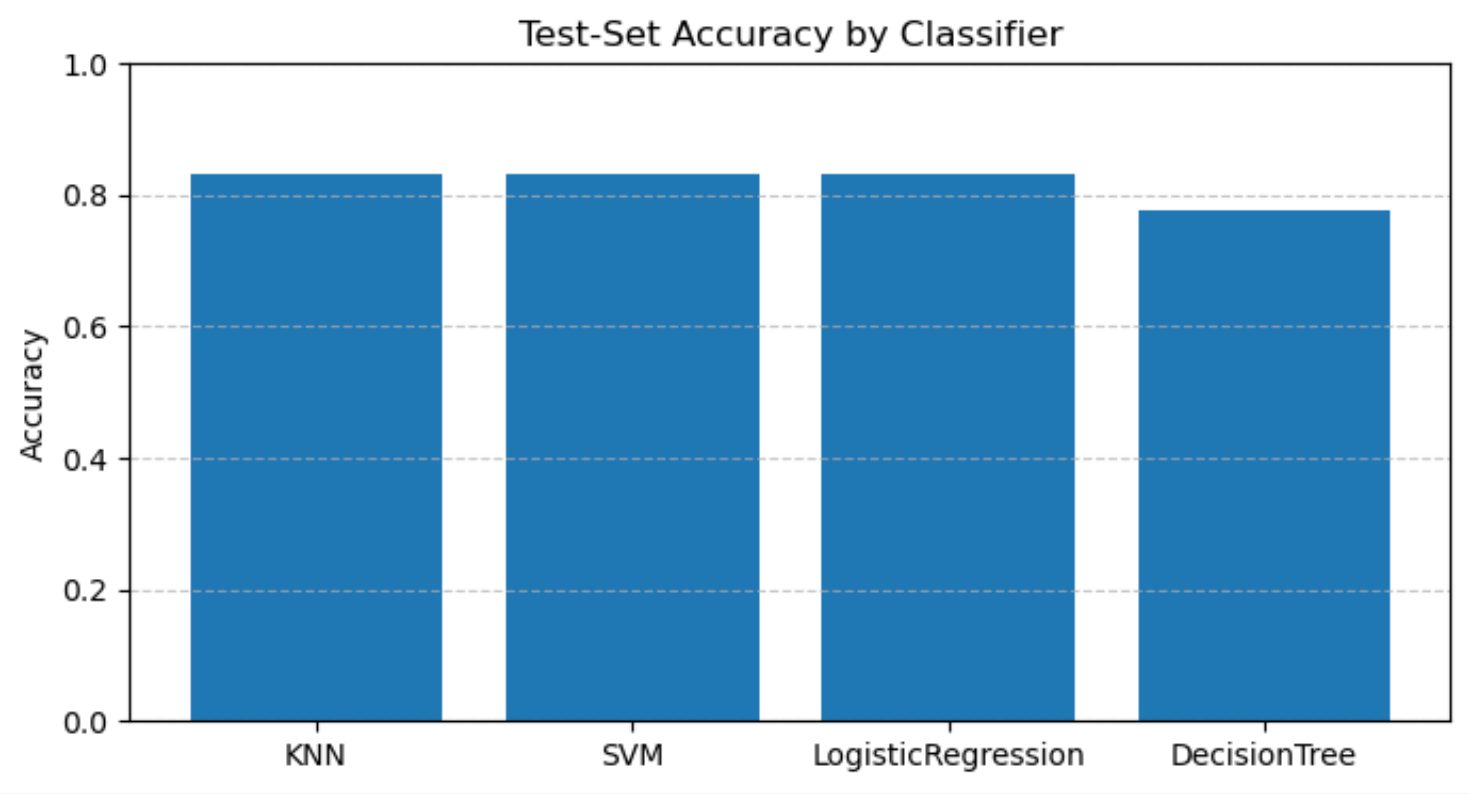


Section 5

Predictive Analysis (Classification)

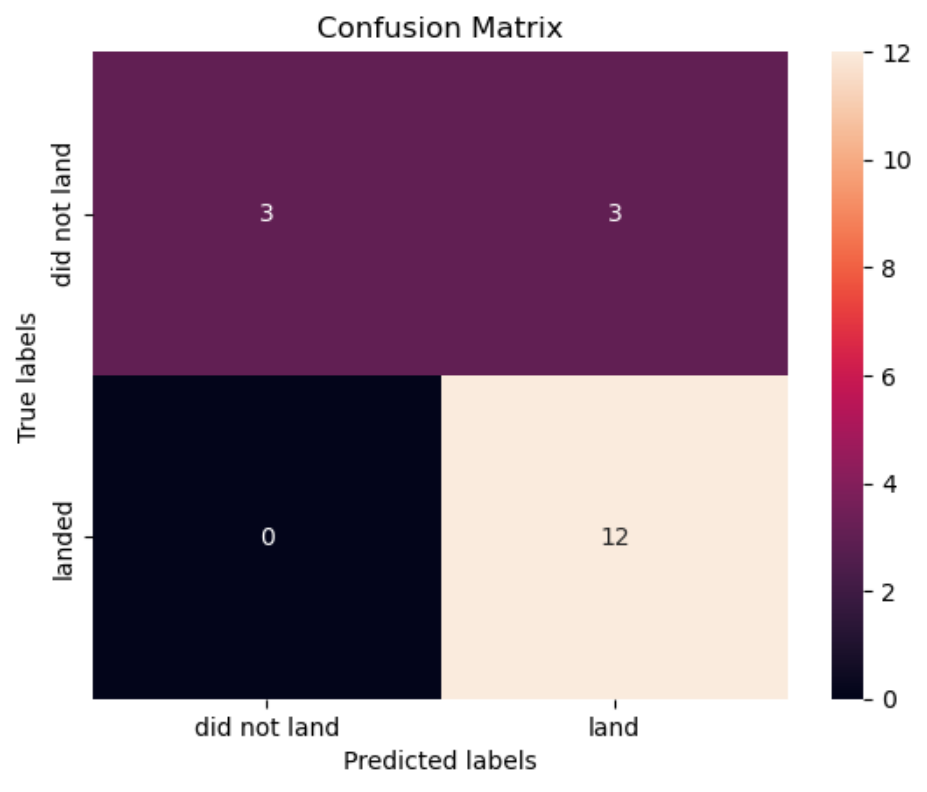
Classification Accuracy

KNN, SVM and LogisticRegression each achieve the highest classification accuracy of 83.33%, with DecisionTree trailing at 72.22%.



Confusion Matrix

This confusion matrix applies to the three top-performing classifiers (KNN, SVM and LogisticRegression), which all behave identically: they record 3 true negatives, 3 false positives, 12 true positives and 0 false negatives, yielding perfect recall and an overall precision of 80%.



Conclusions

- The choice of U.S. coastal launch sites between approximately 25° N and 34° N latitude has been shown to ensure safe stage drop-zones over open water while capturing a measurable boost from Earth's rotation.
- CCAFS SLC-40's location at 870m from the shoreline, ~600–1.000m from major transport arteries, and over 19km from populated areas, has been demonstrated to optimize both logistical throughput and range-safety margins.
- Payloads in the 2.000 – 6.000 kg band launched on later booster variants (FT, B5) have been observed to exhibit the highest landing success rates, whereas early-generation vehicles and sub-1 000 kg missions revealed a greater incidence of failures.
- KSC LC-39A has recorded a 76.9 % landing success rate (10 of 13), making it the most reliable site by share of total successful missions.
- Classifiers trained on a comprehensive feature set (including payload mass, booster version, mission sequence number and additional mission parameters) uniformly achieved 83.33 % test-set accuracy with zero false negatives, underscoring the predictive strength of the combined variables.

Appendix – Data collection Auxiliary functions

Auxiliary functions used for scarping and API requests.

```python
def date_time(table_cells):
    """
    This function returns the data and time from the HTML  table cell
    Input: the  element of a table data cell extracts extra row
    """
    return [data_time.strip() for data_time in list(table_cells.strings)][0:2]

def booster_version(table_cells):
    """
    This function returns the booster version from the HTML  table cell
    Input: the  element of a table data cell extracts extra row
    """
    out=''.join([booster_version for i,booster_version in enumerate( table_cells.strings) if i%2==0][0:-1])
    return out

def landing_status(table_cells):
    """
    This function returns the landing status from the HTML table cell
    Input: the  element of a table data cell extracts extra row
    """
    out=[i for i in table_cells.strings][0]
    return out


def get_mass(table_cells):
    mass=unicodedata.normalize("NFKD", table_cells.text).strip()
    if mass:
        mass.find("kg")
        new_mass=mass[0:mass.find("kg")+2]
    else:
        new_mass=0
    return new_mass


def extract_column_from_header(row):
    """
    This function returns the landing status from the HTML table cell
    Input: the  element of a table data cell extracts extra row
    """
    if (row.br):
        row.br.extract()
    if row.a:
        row.a.extract()
    if row.sup:
        row.sup.extract()

    colunm_name = ' '.join(row.contents)

    # Filter the digit and empty names
    if not(colunm_name.strip().isdigit()):
        colunm_name = colunm_name.strip()
        return colunm_name

```

From the `rocket` column we would like to learn the booster name.

```python
# Takes the dataset and uses the rocket column to call the API and append the data to the list
def getBoosterVersion(data):
    for x in data['rocket']:
        if x:
         response = requests.get("https://api.spacexdata.com/v4/rockets/"+str(x)).json()
         BoosterVersion.append(response['name'])
```

From the `launchpad` we would like to know the name of the launch site being used, the logitude, and the latitude.

```python
# Takes the dataset and uses the launchpad column to call the API and append the data to the list
def getLaunchSite(data):
    for x in data['launchpad']:
        if x:
          response = requests.get("https://api.spacexdata.com/v4/launchpads/"+str(x)).json()
          Longitude.append(response['longitude'])
          Latitude.append(response['latitude'])
          LaunchSite.append(response['name'])
```

From the `payload` we would like to learn the mass of the payload and the orbit that it is going to.

```python
# Takes the dataset and uses the payloads column to call the API and append the data to the lists
def getPayloadData(data):
    for load in data['payloads']:
        if load:
         response = requests.get("https://api.spacexdata.com/v4/payloads/"+load).json()
         PayloadMass.append(response['mass_kg'])
         Orbit.append(response['orbit'])
```

From `cores` we would like to learn the outcome of the landing, the type of the landing, number of flights with that core, whether the core is reused, the number of times it was reused, whether legs were used, the landing pad used, the block of the core which is a number used to seperate version of cores, the number of times this specific core has been reused, and the serial of the core.

```python
# Takes the dataset and uses the cores column to call the API and append the data to the lists
def getCoreData(data):
    for core in data['cores']:
            if core['core'] != None:
                response = requests.get("https://api.spacexdata.com/v4/cores/"+core['core']).json()
                Block.append(response['block'])
                ReusedCount.append(response['reuse_count'])
                Serial.append(response['serial'])
            else:
                Block.append(None)
                ReusedCount.append(None)
                Serial.append(None)
            Outcome.append(str(core['landing_success'])+' '+str(core['landing_type']))
            Flights.append(core['flight'])
            GridFins.append(core['gridfins'])
            Reused.append(core['reused'])
            Legs.append(core['legs'])
            LandingPad.append(core['landpad'])
```

Appendix – Folium

Code snippets defining Folium MarkerCluster, PolyLine and Mouse Position

```
import folium
from folium.plugins import MarkerCluster

# 1) Inizializza la mappa
#site_map = folium.Map(location=[20, 0], zoom_start=2)

# 2) Crea il MarkerCluster e aggancialo subito alla mappa
marker_cluster = MarkerCluster(
    name='Launch Results',
    #options={'disableClusteringAtZoom': 6}  # opzionale, scioglie i cluster oltre lo zoom 6
).add_to(site_map)

# 3) Cicla sul DataFrame spacex_df
for _, row in spacex_df.iterrows():
    coord = [row['Lat'], row['Long']]
    label = row['Launch Site']
    color = row['marker_color']  # 'red' o 'green'

    # create & add marker direttamente al cluster
    folium.Marker(
        location=coord,
        popup=label,
        icon=folium.Icon(
            color='white',       # bordo esterno
            icon_color=color,    # pallino interno
            icon='map-marker'
        )
    ).add_to(marker_cluster)

# 4) (Opzionale) layer control per attivare/disattivare il cluster
site_map.add_child(folium.LayerControl())
```

```
distance_coastline = calculate_distance(28.56319718, -80.57682003,28.56319718,-80.58686)
distance_marker = folium.Marker(
   location= [28.56319718,-80.58686],
  icon=DivIcon(
       icon_size=(20,20),
       icon_anchor=(0,0),
         html='<div style="font-size: 12; color:#d35400;"><b>%s</b></div>' % "{:10.2f} KM - Railway".format(distance_coastline),
       ))
site_map.add_child(distance_marker )
lines=folium.PolyLine(locations=[[28.56319718, -80.57682003],[28.56319718,-80.58686]], weight=1)
site_map.add_child(lines)
```

```
# Add Mouse Position to get the coordinate (Lat, Long) for a mouse over on the map
formatter = "function(num) {return L.Util.formatNum(num, 5);};"
mouse_position = MousePosition(
    position='topright',
    separator=' Long: ',
    empty_string='NaN',
    lng_first=False,
    num_digits=20,
    prefix='Lat:',
    lat_formatter=formatter,
    lng_formatter=formatter,
)

site_map.add_child(mouse_position)
site_map
```

Appendix – Dashboard Layout

Layout of the Dash app

```
# Create an app layout
app.layout = html.Div(children=[html.H1('SpaceX Launch Records Dashboard',
                                        style={'textAlign': 'center', 'color': '#503D36',
                                               'font-size': 40}),
                                # TASK 1: Add a dropdown list to enable Launch Site selection
                                # The default select value is for ALL sites
                                dcc.Dropdown(id='site-dropdown',
                                    options= options,
                                    value='ALL',
                                    placeholder="place holder here",
                                    searchable=True
                                    ),

                                # TASK 2: Add a pie chart to show the total successful launches count for all sites
                                # If a specific launch site was selected, show the Success vs. Failed counts for the site
                                html.Div(dcc.Graph(id='success-pie-chart')),
                                html.Br(),

                                html.P("Payload range (Kg):"),
                                # TASK 3: Add a slider to select payload range
                                dcc.RangeSlider(id='payload-slider',
                                                min=0, max=10000, step=1000,
                                                marks= marks,
                                                value=[min_payload, max_payload]),

                                # TASK 4: Add a scatter chart to show the correlation between payload and launch success
                                html.Div(dcc.Graph(id='success-payload-scatter-chart')),
                                ])
```

Appendix – Dashboard Callback Functions

Callback functions defining the pie chart and scatter plots of the Dash app

```python
# TASK 2:
# Add a callback function for `site-dropdown` as input, `success-pie-chart` as output
# Function decorator to specify function input and output
@app.callback(Output(component_id='success-pie-chart', component_property='figure'),
              Input(component_id='site-dropdown', component_property='value')) #component id must be the same of the dropdown
def get_pie_chart(entered_site):
    filtered_df = spacex_df
    if entered_site == 'ALL':
        fig = px.pie(filtered_df, values='class', #values needs to be an existing column
        names='Launch Site',
        title='Success Rate of All sites')
    else:
        # filtro per il sito selezionato
        df_site = spacex_df[spacex_df['Launch Site'] == entered_site]
        # conto quante volte compare 0 e quante volte 1
        counts = df_site['class'].value_counts().reset_index()
        counts.columns = ['class', 'count']

        # costruisco la torta su count vs class
        fig = px.pie(
            counts,
            values='count',     # colonna con i numeri
            names='class',      # colonna con le etichette 0 e 1
            color='class',
            color_discrete_map={0:'#CB3234', 1:'#618E3C'},
            title=f"Success Rate for {entered_site}"
        )
    return fig
```

```python
# TASK 4:
# Add a callback function for `site-dropdown` and `payload-slider` as inputs, `success-payload-scatter-chart` as output
@app.callback(Output(component_id='success-payload-scatter-chart', component_property='figure'),
              [Input(component_id='site-dropdown', component_property='value'), Input(component_id="payload-slider", component_property="value")]) #
def get_scatter(entered_site, slider_range):
    low, high = slider_range
    filtered_df = spacex_df[
        (spacex_df['Payload Mass (kg)'] >= low) &
        (spacex_df['Payload Mass (kg)'] <= high)]
    if entered_site == 'ALL':
        fig = px.scatter(filtered_df,
        x='Payload Mass (kg)', #Payload
        y='class',
        color= 'Booster Version Category',
        title='Scatterplot of Payload vs Success')
    else:
        # filtro per il sito selezionato
        filtered_df = filtered_df[filtered_df['Launch Site'] == entered_site]

        # costruisco la torta su count vs class
        fig = px.scatter(filtered_df,
        x='Payload Mass (kg)', #Payload
        y='class',
        color= 'Booster Version Category',
        title=f'Scatterplot of Payload vs Success: {entered_site}')
    return fig
```

Appendix – Predictive Analysis

Dataset for Model prediction:

- Y = pd.read_csv("https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/dataset_part_2.csv")["Class"].to_numpy()
- X = pd.read_csv('https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/dataset_part_3.csv')

Thank you!