IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Charles Bergin
August 14, 2023

Contents

Introduction

- **Objective**
  - Determine launch prices and predict first stage reuse for Space Y, a new rocket booster company targeting satellite delivery and space services
- **Target Market**
  - The key market segment is the satellite market, a $14bn. p.a. market that is expected to grow to $40bn. p.a. by 2030
- **Competition**
  - SpaceX is the market leader in rocket booster thanks the decisive cost savings of reusable Falcon 9 rocket boosters
- **Feasibility**
  - Rocket boosters cost $70 million so the feasibility of SpaceY competing with SpaceX relies upon identifying the optimum launch site and developing models to predict rocket reuse rates
- **Key Challenge**
  - We should predict if a rocket's first stage will land successfully. The launch success rate primarily depend on payload mass and orbit type, but also depends the location of a launch site.
- **Launch Success Rate**
  - We identify and analyze correlations between the launch site and historical success rates from SpaceX launch sites
- **Rocket Reusability Success Rate**
  - We will not rely on traditional rocket science but machine learning models to predict success rates for rocket reusability using SpaceX data on landing outcomes, orbits and payload

Executive Summary

The primary goal is to competitively bid against SpaceX by leveraging information from descriptive results from EDA and predictive insights with Machine Learning to calculate the profitability of potential contracts.

Summary of methodologies

- Data Collection, Wrangling & Exploratory Data Analysis
  - Comprehensive data was gathered and organized
  - Data quality was enhanced by cleaning and structuring
  - In-depth exploration of data was conducted to uncover patterns and trends.
- Visualization & Interactive Dashboard
  - Meaningful visualizations were created to present data-driven insights effectively.
  - An interactive dashboard was developed, enabling dynamic exploration of data findings.
- Predictive Analysis with Machine Learning
  - Advanced predictive modelling techniques were applied to forecast Falcon 9 first stage landings.

Summary of all results

- **Revenue Drivers:** Results from descriptive EDA indicate the market segments with the greatest commercial potential and potential profitability:
  - Optimum Payload: 3000kg-7000kg payload
  - Optimum Orbits: GTO, SSO, HEO. ES-L1 & VLEO
- **Costs Drivers:** Results from EDA/predictive analysis show:
  - Optimum Launch Site: Kennedy Space Center
  - Model Prediction Success Rate: >83%
- **Profitability:** By analyszng potential revenue sources and cost drivers, SpaceY's Business Development and Finance teams have the data to prepare financial models showing the profitability (revenues v. costs) of contracts in target market segments where they will compete with SpaceX.

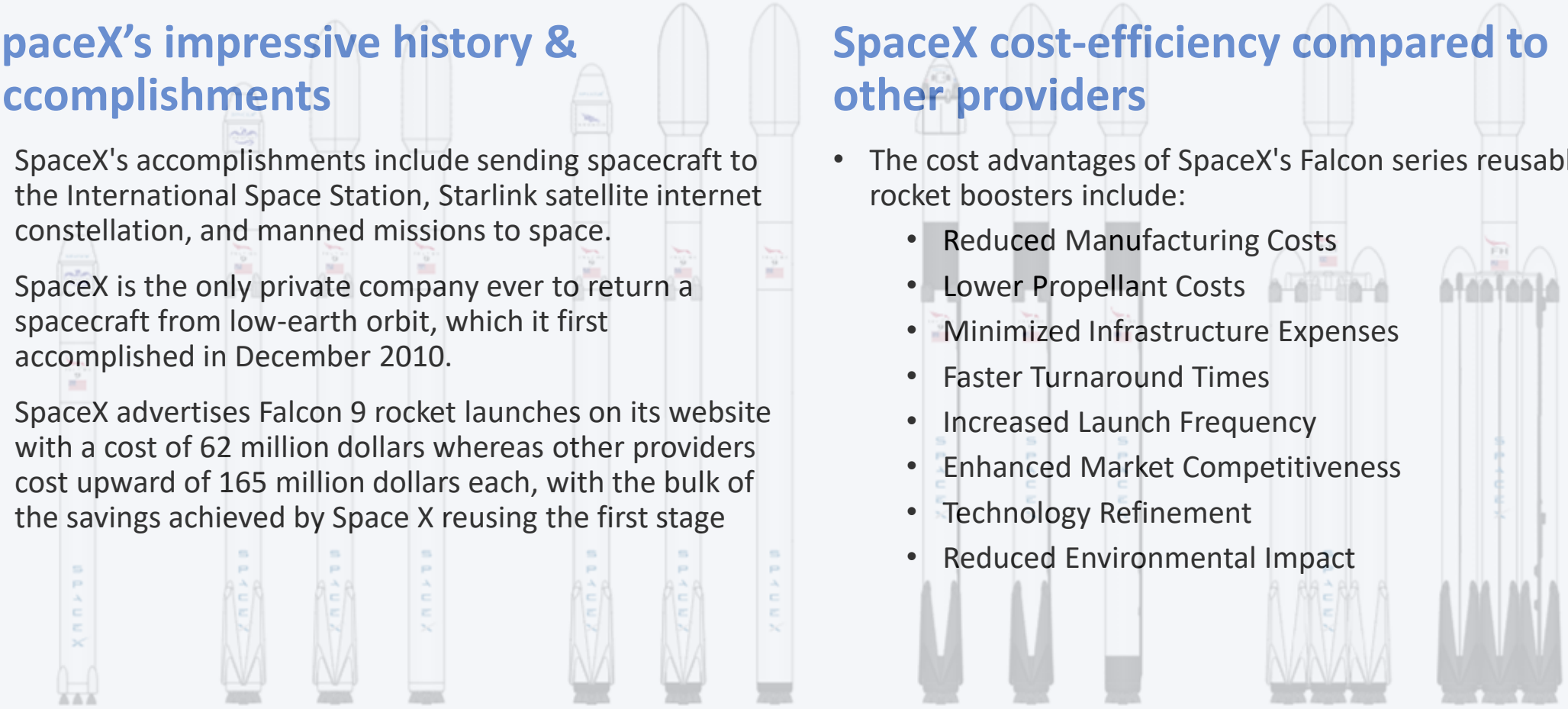
SpaceX History & Competitive Advantage

SpaceX's impressive history & accomplishments

- SpaceX's accomplishments include sending spacecraft to the International Space Station, Starlink satellite internet constellation, and manned missions to space.
- SpaceX is the only private company ever to return a spacecraft from low-earth orbit, which it first accomplished in December 2010.
- SpaceX advertises Falcon 9 rocket launches on its website with a cost of 62 million dollars whereas other providers cost upward of 165 million dollars each, with the bulk of the savings achieved by Space X reusing the first stage

SpaceX cost-efficiency compared to other providers

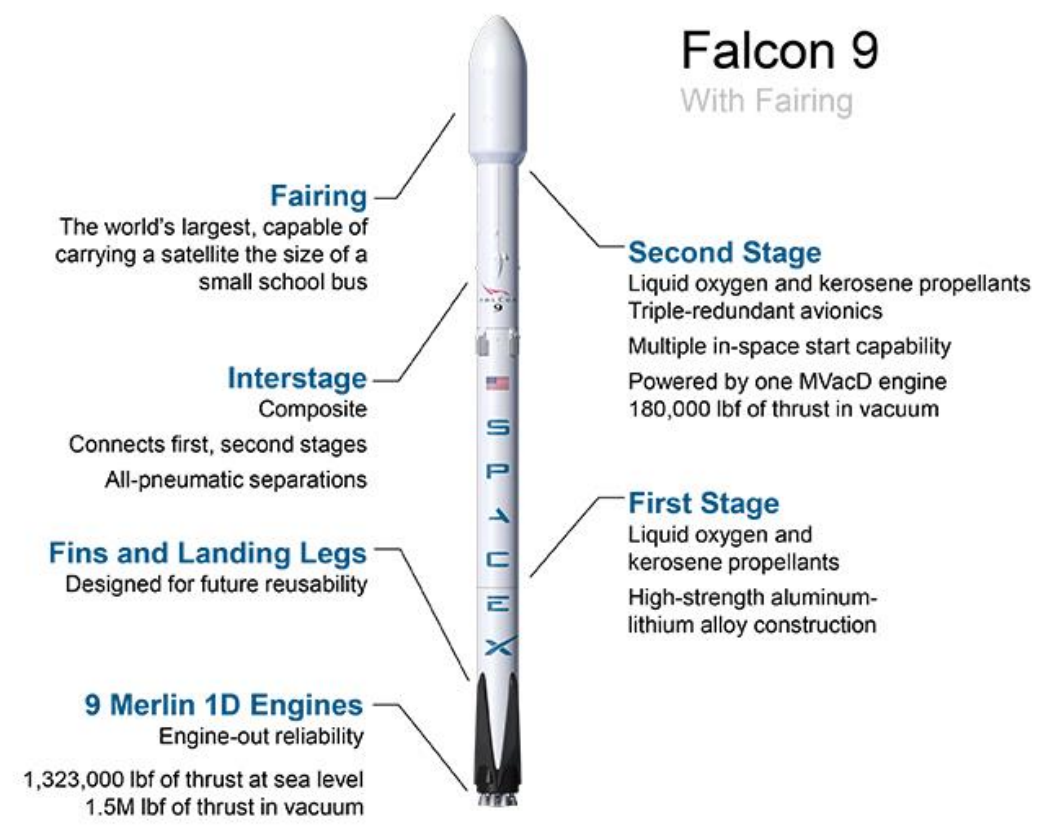
- The cost advantages of SpaceX's Falcon series reusable rocket boosters include:
  - Reduced Manufacturing Costs
  - Lower Propellant Costs
  - Minimized Infrastructure Expenses
  - Faster Turnaround Times
  - Increased Launch Frequency
  - Enhanced Market Competitiveness
  - Technology Refinement
  - Reduced Environmental Impact

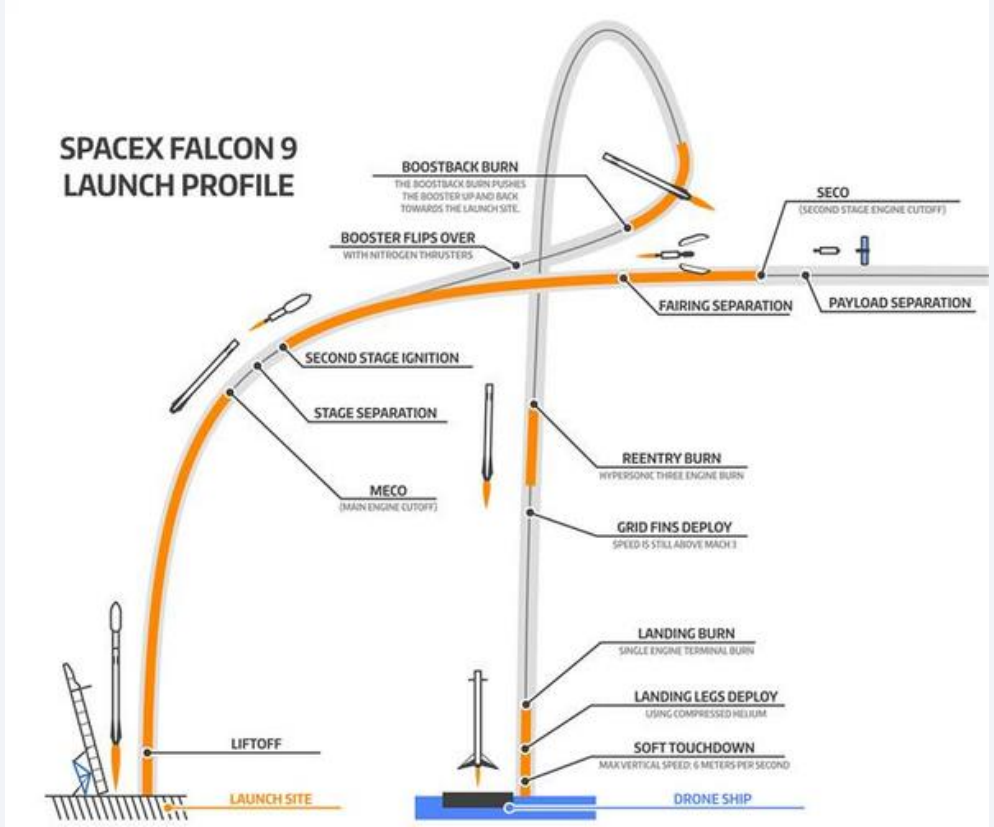


Source: Wikipedia

Falcon 9 series

SpaceX's Falcon 9 and Falcon Heavy boosters are designed for reusability and reduce space travel costs. Falcon 9's reusability transformed the space industry, reduced costs and increased launch frequency.





Source: Wikipedia

Technical Features & Competitors

SpaceX's Falcon 9 enjoys a decisive launch cost at $62 million vs. competitors' $165 million

Falcon 9 Technical Features

- Boosters have landing legs and grid fins for controlled descent and landing.
- Boostback burn initiated after primary stage separates from the second stage.
- Booster engines reignited for controlled descent and landing accuracy.
- Two landing methods: ground landings near launch site and drone ship landings at sea.
- Boosters inspected, refurbished, and repaired after landing for flight readiness.
- Refurbished boosters integrated with payload and second stage for new missions.
- Reused boosters launched again, maximizing hardware utilization and lowering costs.
- Continuous improvements based on data collected from missions.

SpaceX Competitors

- While there are an increasing number of competitors in the launch market, no competitor has emerged to rival SpaceX's re-use of the first stage booster

ULA America's Ride to Space

Virgin GALACTIC

BLUE ORIGIN

arianespace arianeGROUP

ROCKETLAB

NORTHROP GRUMMAN

Section 1

Methodology

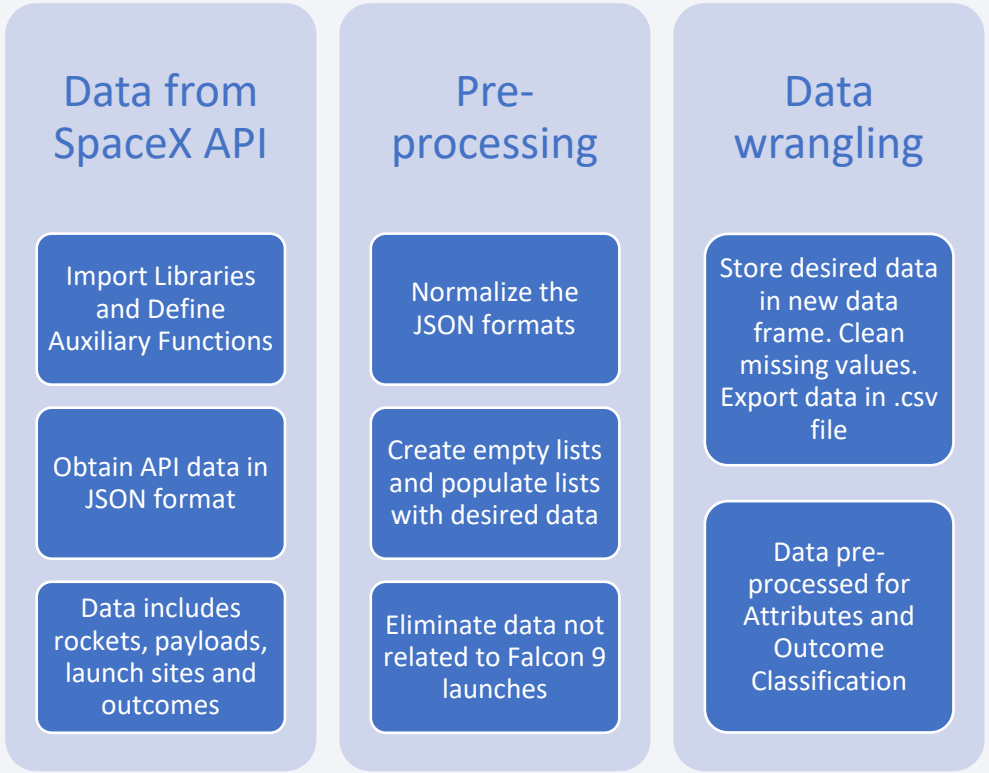
Methodology

- Data collection methodology
  - Data collected from SpaceX API, Wikipedia using Beautiful Soup
- Perform data wrangling
  - Data pre-processing, landing outcomes converted to binary classes
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - Evaluation with linear regression, SVM, KNN and Decision Tree classification models using scoring metrics and confusion matrices

Data Collection Methodology

SpaceX API

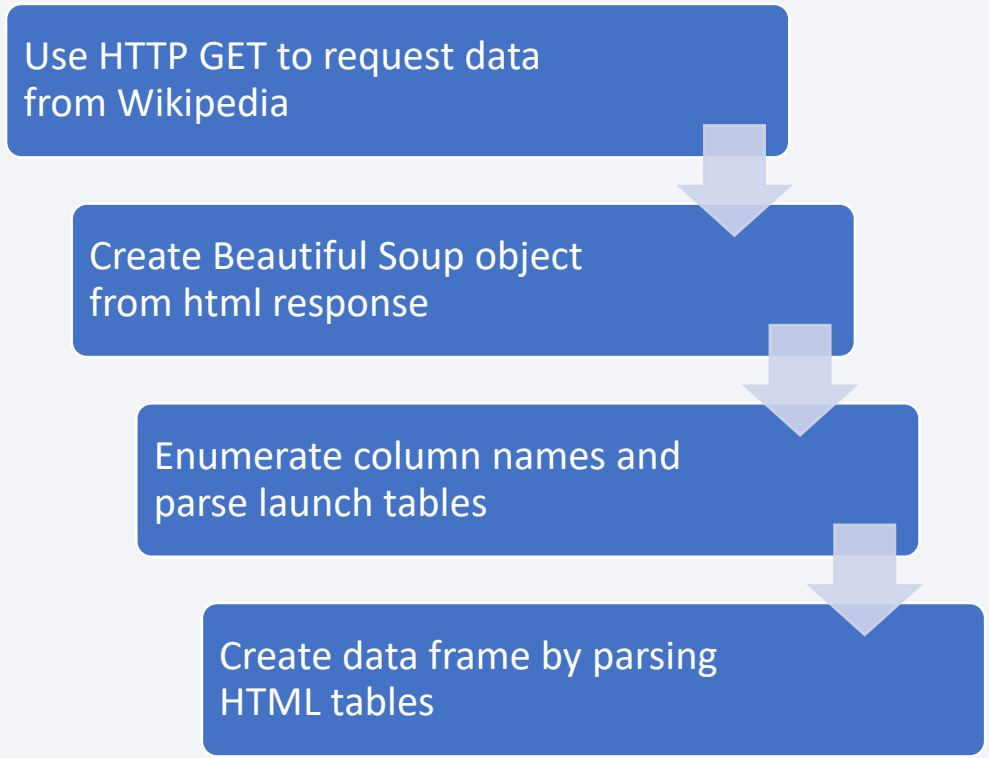
- Request to the SpaceX API
- Clean the requested data



SpaceX API Notebook

Webscraping

- Extract Falcon 9 launch records as HTML table
- Parse the table and convert it into a Pandas data frame



Webscraping Notebook

Data Wrangling & EDA Visualization

Data Wrangling

- Exploratory Data Analysis to determine labels for training supervised models

Libraries: numpy, pandas

Categorical outcome labels:

1=‘Booster successfully landed’,

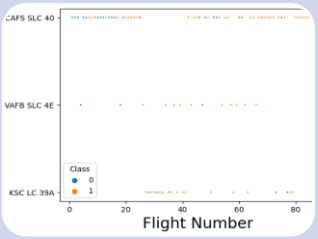
0=‘Booster failed to land ‘

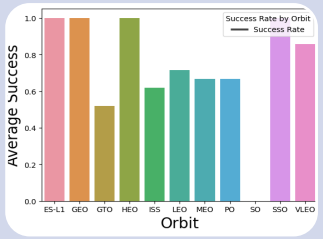
Data Analysis

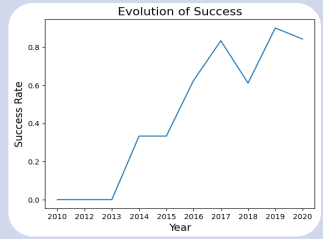
- Calculate Launches per site, Missions by Orbits and Results by Orbits
- Create landing outcomes labels for EDA and predictive analysis

EDA with Visualization

- Exploratory Data Analysis with Charts
- Preparing Data Feature Engineering







Success rate by launch site

Success rate by Orbit

Success rate over time

[Data Wrangling Notebook]

[EDA with Visualization Notebook]

EDA with SQL & Site Analysis with Folium

EDA with SQL

- Understanding the SpaceX dataset using SQL to query Db2 database

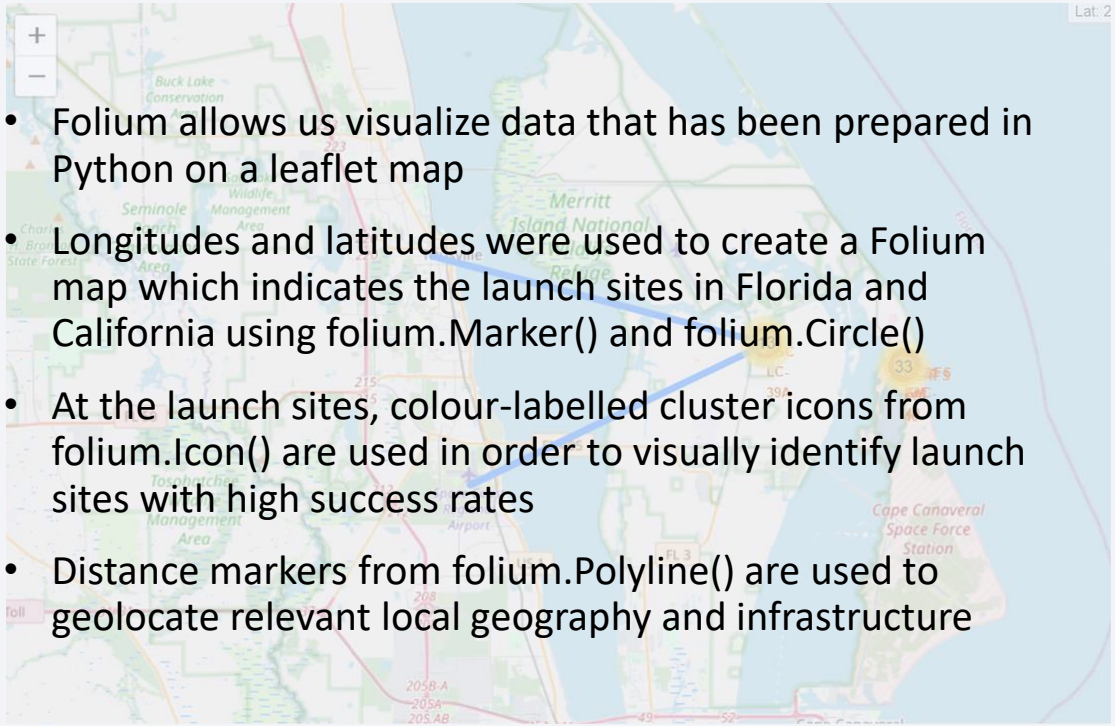
Queries

- Names of the unique launch sites in the space mission
- Total payload mass carried by boosters launched by NASA
- Average payload mass carried by booster version F9 v1.1
- Date of first successful landing outcome in ground pad
- Names of boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000
- Total number of successful and failure mission outcomes
- Names of the boosters which carried the maximum payload
- Iterate launch records for the year 2015
- Rank landing outcomes for 2010-06-04 and 2017-03-20

EDA with SQL Notebook

Interactive Map with Folium

- Identify geographical pattens around launch sites by locating sites on a map



- Folium allows us visualize data that has been prepared in Python on a leaflet map
- Longitudes and latitudes were used to create a Folium map which indicates the launch sites in Florida and California using folium.Marker() and folium.Circle()
- At the launch sites, colour-labelled cluster icons from folium.Icon() are used in order to visually identify launch sites with high success rates
- Distance markers from folium.Polyline() are used to geolocate relevant local geography and infrastructure

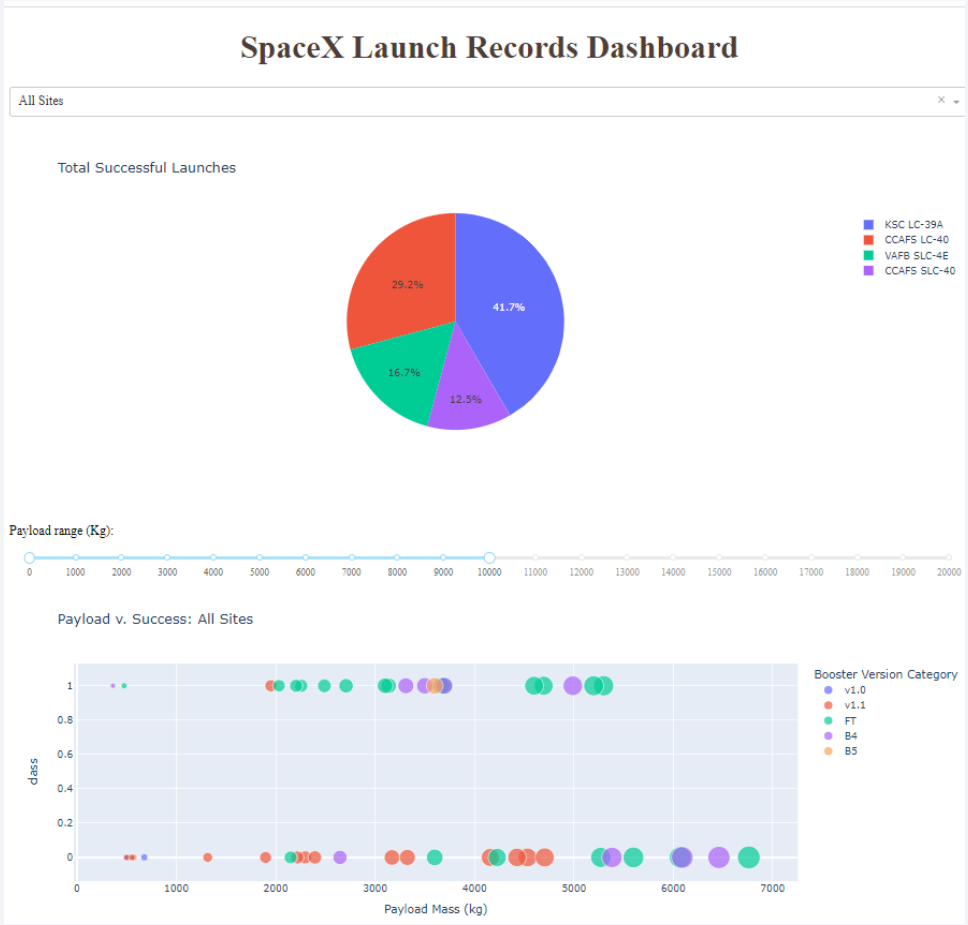
Launch Site Analysis with Folium Notebook

Interactive Dashboard

- The Dash application contains html components and includes an input dropdown list and a range slider to interact with two output components, a pie chart and a scatter point chart generated using Plotly and Seaborn

- Data frames created us pandas were used to:
  - Adding a Launch Site Drop-down Input Component
  - Adding a callback function to render a success-pie-chart based on selected site dropdown
  - Adding a Range Slider to Select Payload
  - Adding a callback function to render a success-payload-scatter-chart scatter plot

- Using the interactive dashboard to explore difference inputs allows us to find insights from the SpaceX dataset more easily than with static graphs



Dashboard App

Predictive Analysis (Classification)

- The financial feasibility of competing with SpaceX relies upon re-using the first stage rocket booster and predicting if the booster will land is foundational for SpaceY.
- The models selected to predict if the first stage of the rocket booster will land include linear regression, SVM (Support Vector Machines), K Nearest Neighbor and Decision Tree models
- As well as using the usual libraries (pandas, numpy, matplotlib), we use the scikit-learn, a python library to implement machine learning models and statistical modelling.
- Using scikit-learn, we implement various machine learning models for regression, classification, clustering, and statistical tools for analyzing the models.
- After developing the models using training data, we tune the model hyperparameters by fitting and evaluate their accuracy. Finally, we use test data and, using confusion matrices and accuracy scoring, identify the best performing model

Model Development

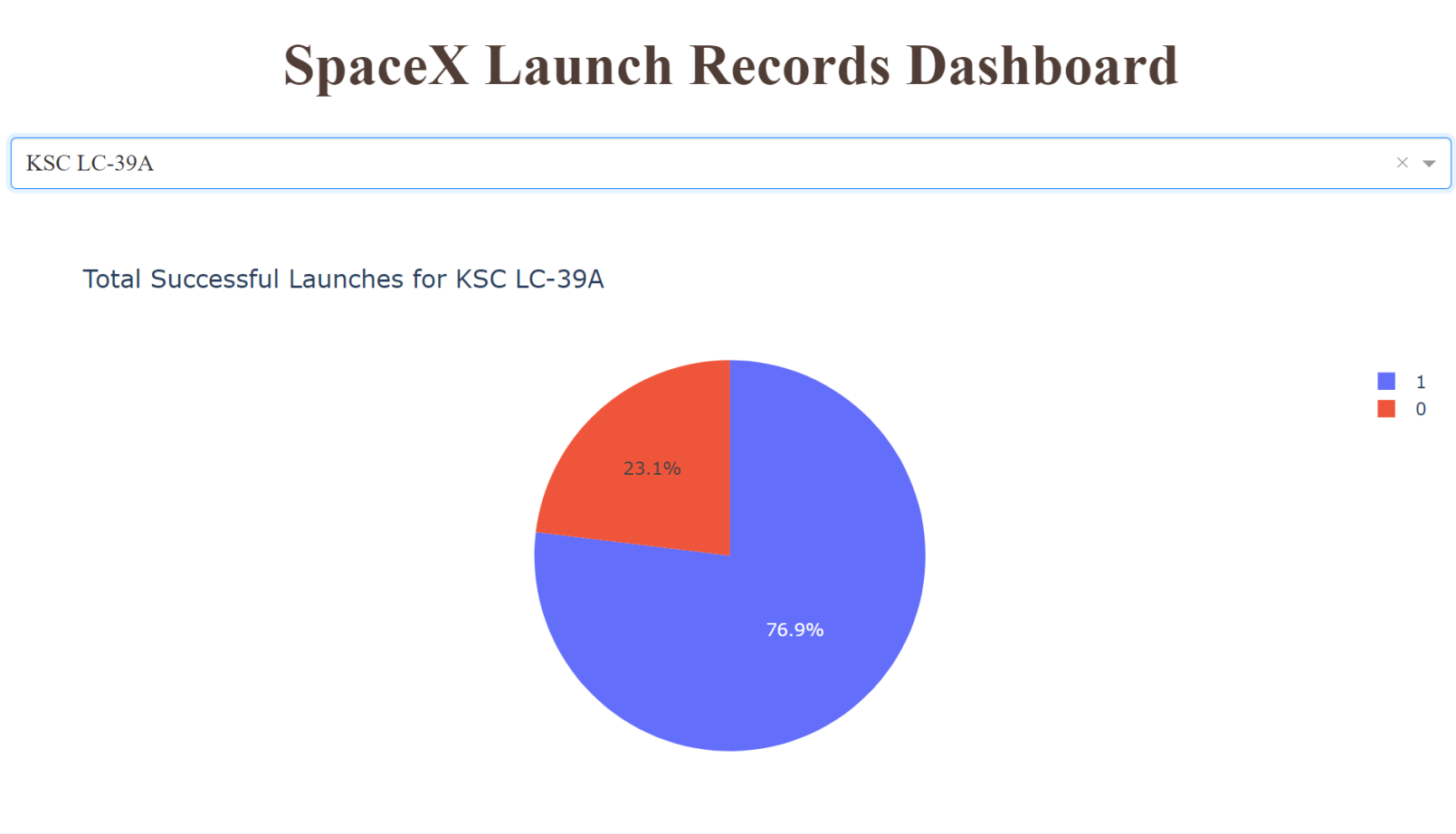
Predictive Analysis with Machine Learning Notebook

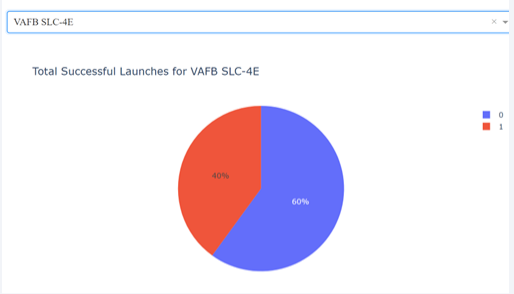
Results: Exploratory Data Analysis

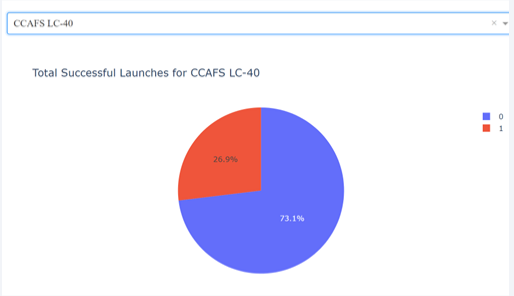
EDA Insights for SpaceX Competitors

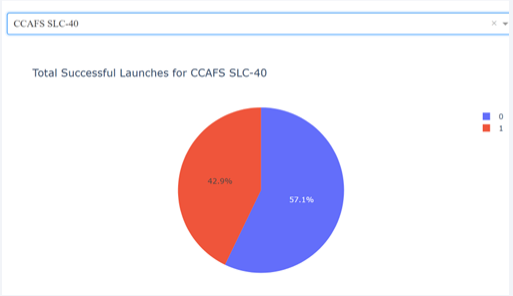
- Predicted Success Rate
  - The prediction model generated a 0.83 score for predicting the success of a rocket booster landing.
- Optimum Payload
  - The 3000kg-7000kg payload range to GTO is the historically most frequent for SpaceX
- Optimum Orbits
  - SpaceX Falcon flights to the GTO, SSO, HEO and ES-L1 orbits are the most successful
  - VLEO recent flights have been successful and is the most active segment of SpaceX flights
- Optimum Launch Site
  - Kennedy Space center has the highest score for successful launches but Cape Canaveral is the busiest launch site in recent years

Results: Launch Success by Site



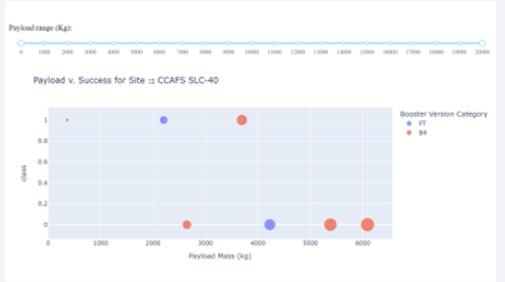


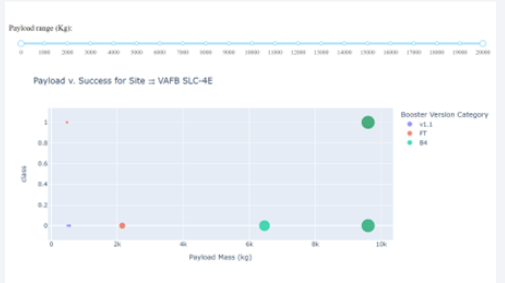




To date, KSC has a success rate of over 76% for launches compared to 73% for CC-LC 40, 60% for VAFB and 57% for CC-SLC-40

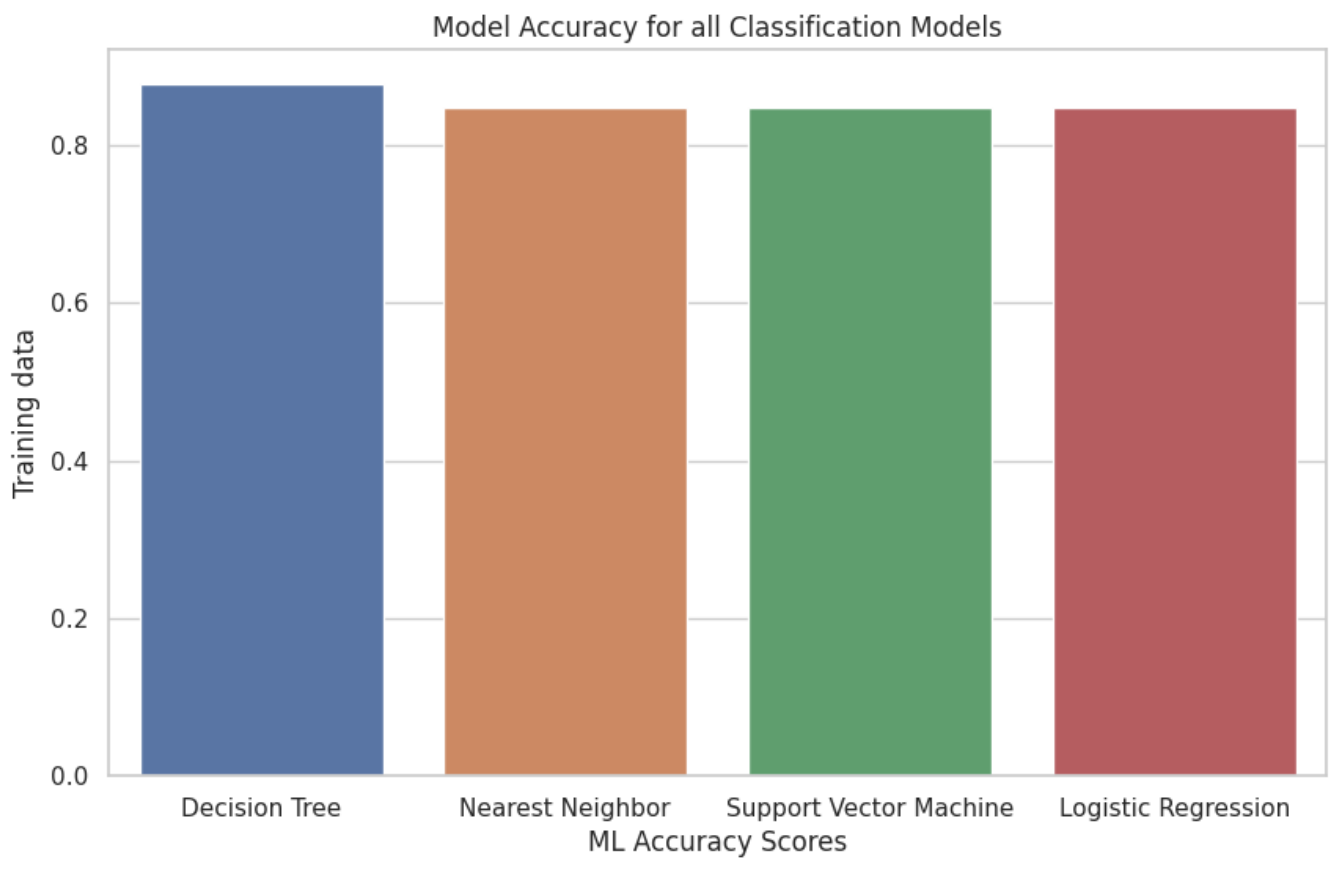
Results: Launch Success by Payload & by Site





Results: Predictive Analytics

Four classification models were evaluated using cross-validation and testing data: Logistic Regression, Support Vector Machine, Decision Tree, and Nearest Neighbour (KNN).

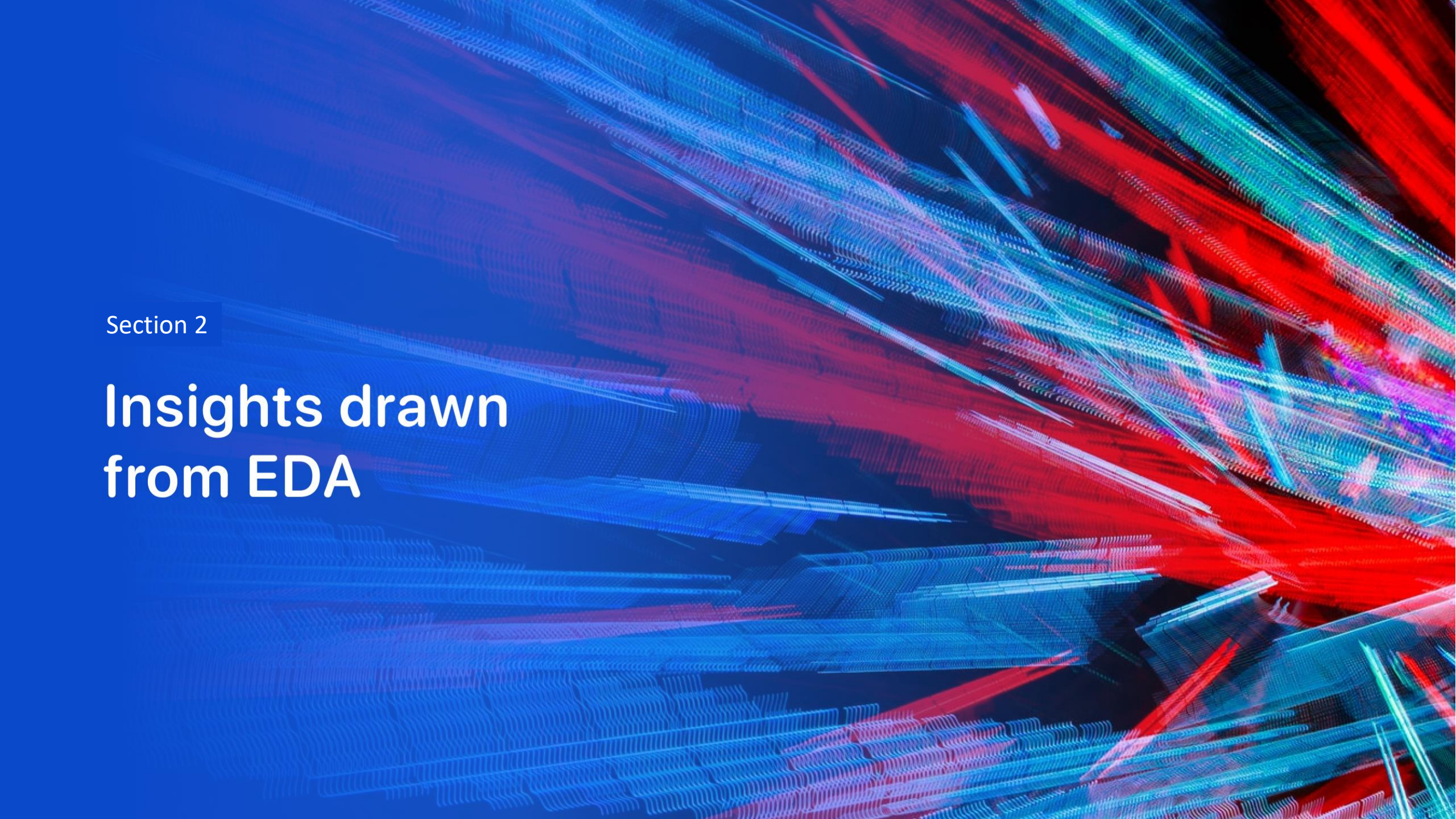


The best method is Decision Tree

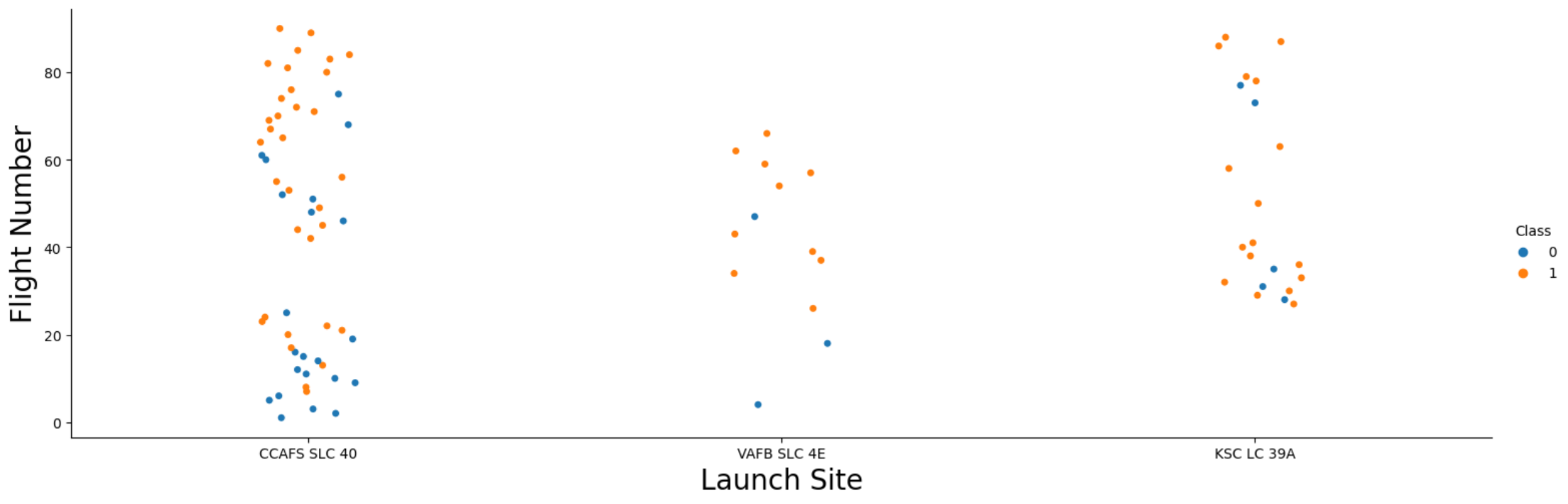
	ML Accuracy Scores	Training data	Testing data
2	Decision Tree	0.876786	0.833333
3	Nearest Neighbor	0.848214	0.833333
1	Support Vector Machine	0.848214	0.833333
0	Logistic Regression	0.846429	0.833333

The best performing model was the decision tree model.

The decision tree model gave a 0.87 result on training data compared to just under 0.85 for SVM, KKN and logistic regression models.

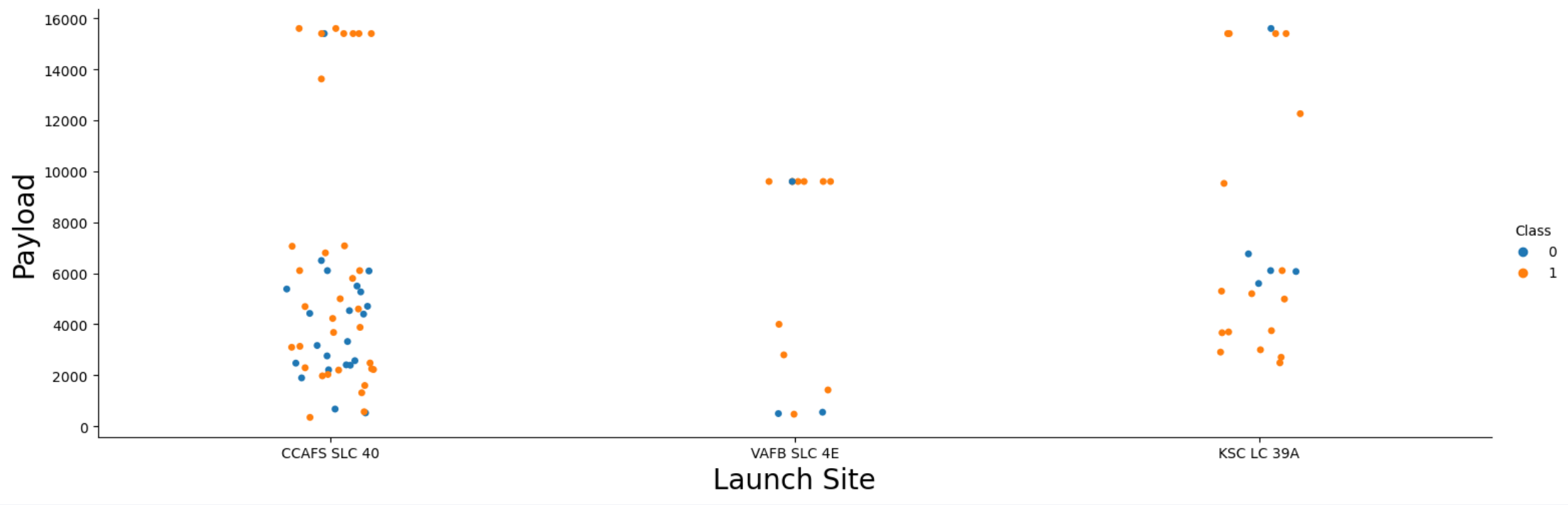

Section 2
Insights drawn from EDA

Flight Number vs. Launch Site



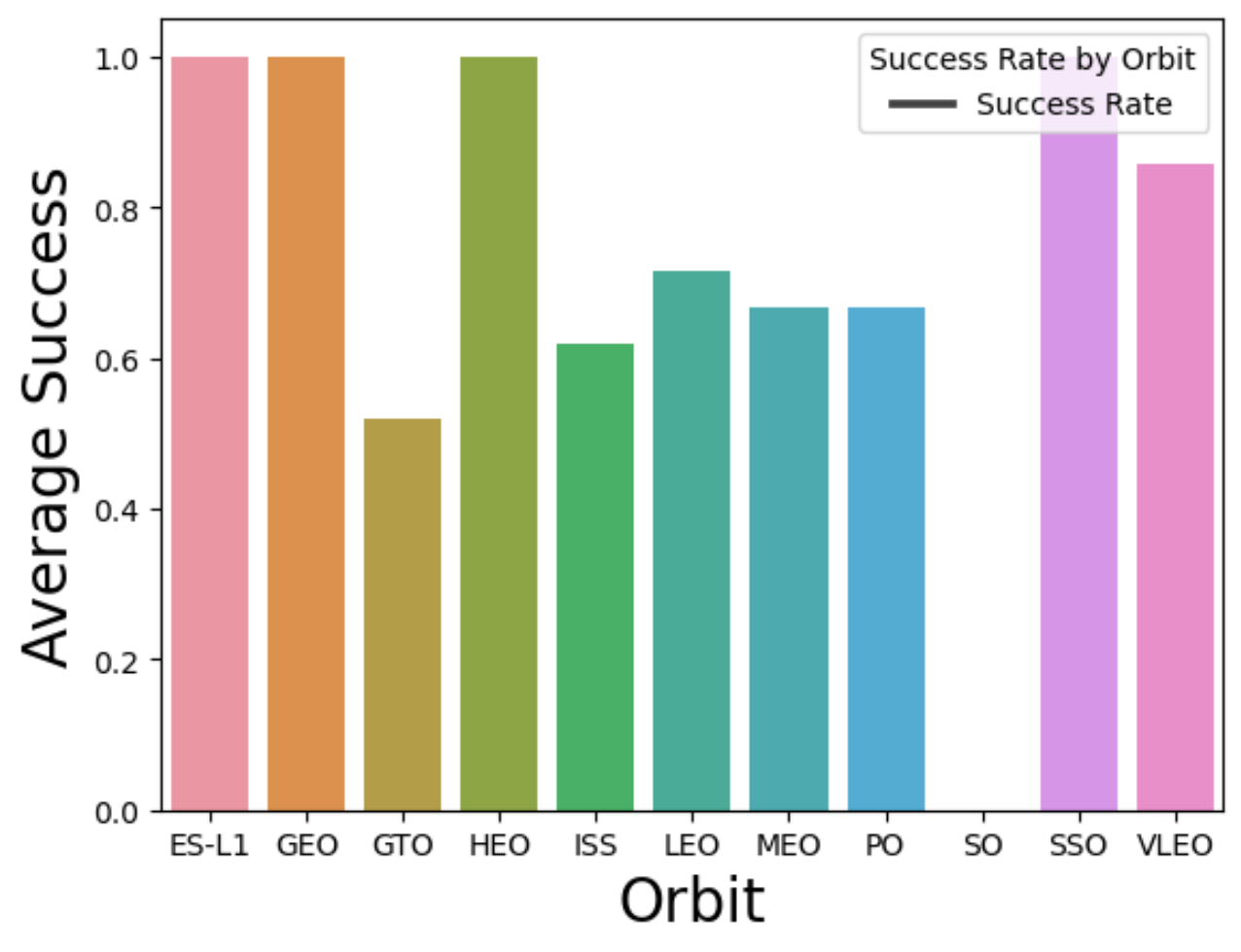
The scatterplots demonstrate that the highest number of launches occurred from Florida at Cape Canaveral, and the least number of launches occurred at Vandenberg Air Force base. Of recent flights, there have been no launches from Vandenberg, while **the highest number of recent launches has been from Cape Canaveral and success rates are high.**

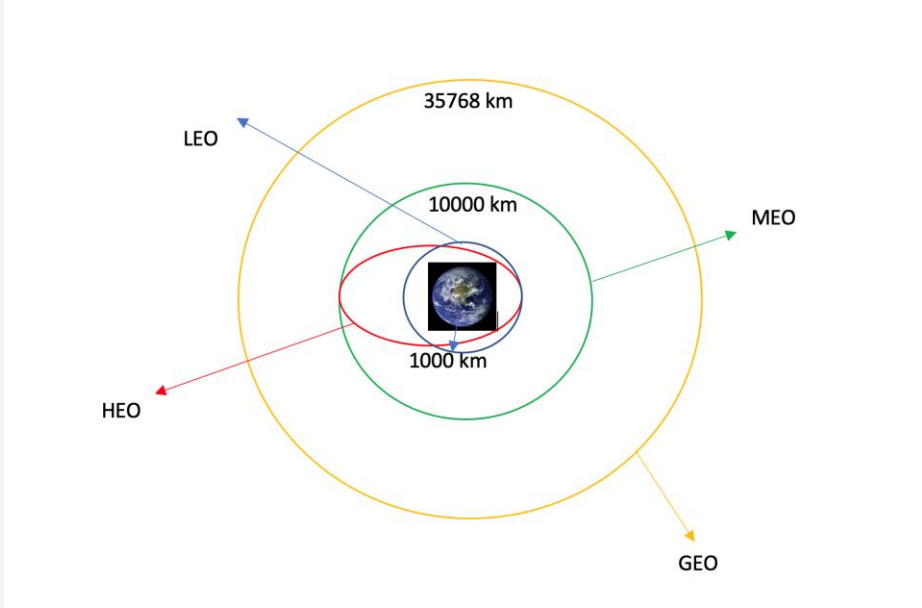
Payload vs. Launch Site



The Payload v. Launch Site scatterplots indicate that launches from Vandenberg Air Force did not include heavier payloads, while a majority of heavier payloads were launched from Cape Canaveral, and with a good success rate.

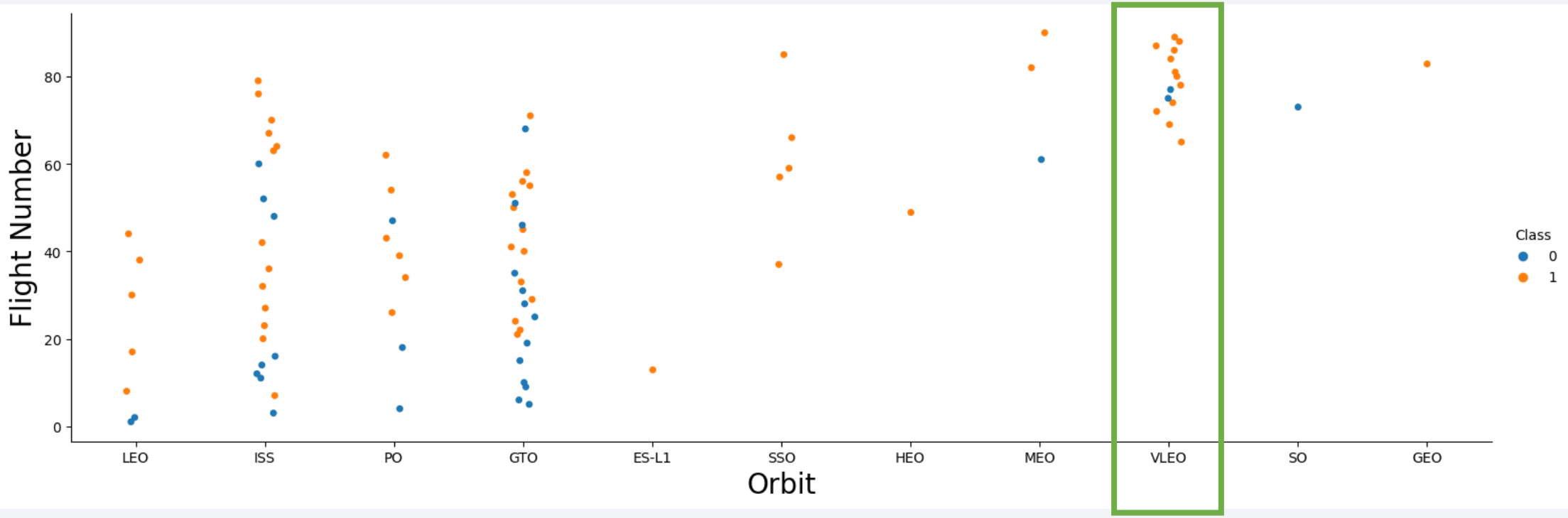
Success Rate vs. Orbit Type





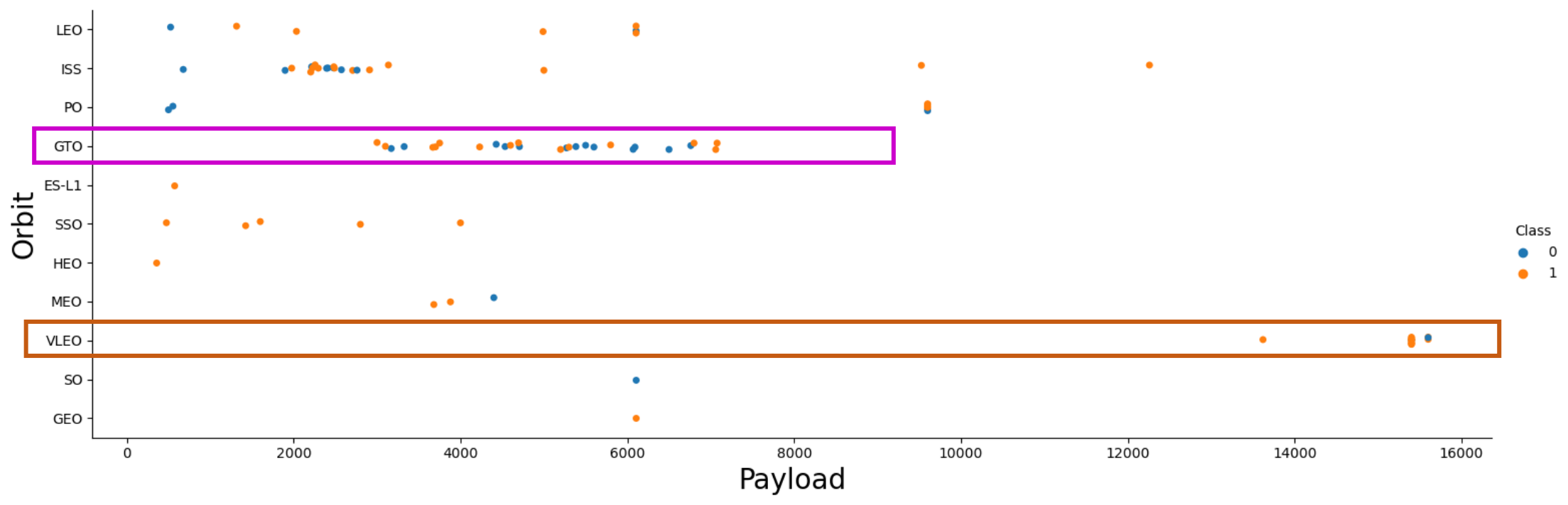
The results demonstrate perfect track records for SpaceX launches for HEO, GEO, HEO, and SSO. SpaceX also has a good recent track record for VLEO, which are usually missions for Starlink.

Flight Number vs. Orbit Type



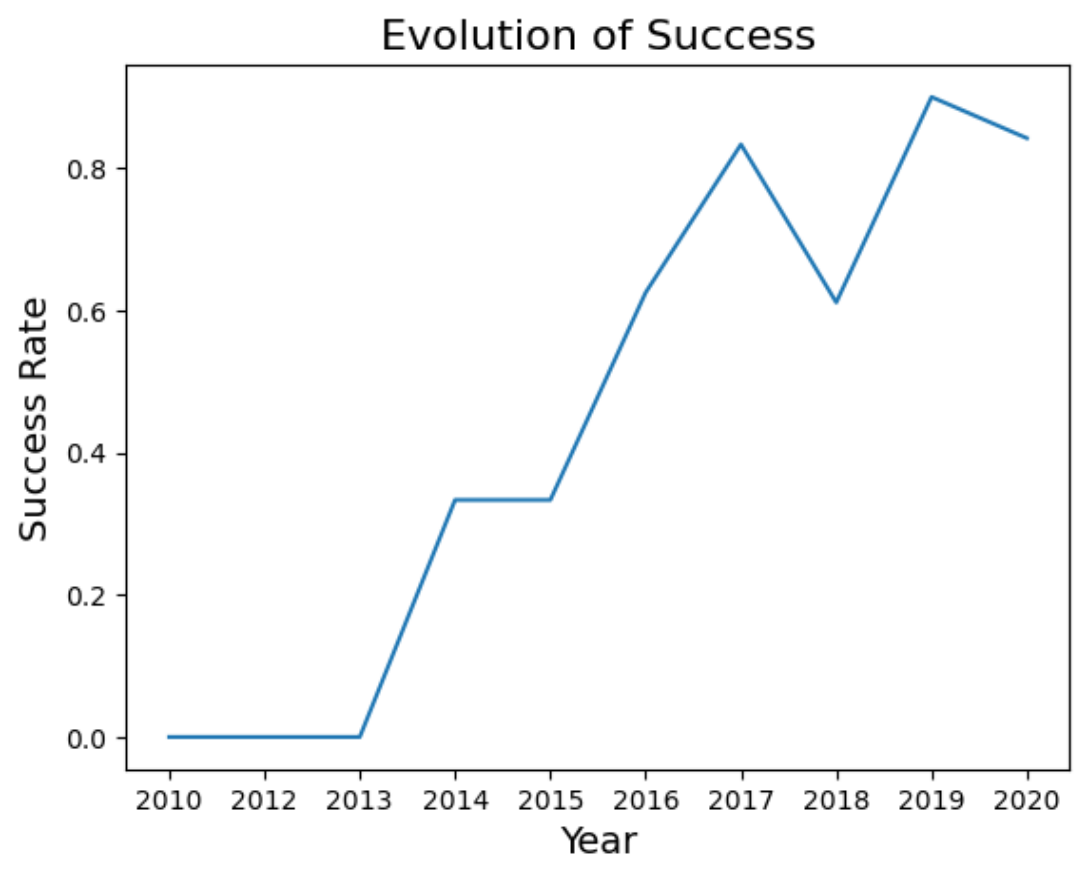
The flight numbers correspond to more recent flights and the chart indicates that recent launches have a far higher percentage of positive outcomes (**orange** dots) than earlier flights and that earlier flight numbers had much higher unsuccessful outcomes (**blue** dots). This is much explicitly demonstrated in the chronological graph of flight successes in a later slide. **SpaceX recently focused on VLEO missions, most of which missions have been successful.**

Payload vs. Orbit Type



The orbit with the highest concentration of heavy payload missions is Very Low Earth Orbit (VLEO), while the orbit with the highest concentration of missions is the Geostationary Transfer Orbit (GTO), from where satellites ultimately travel on their own power to their final Geostationary Earth Orbit (GEO).

Launch Success Yearly Trend



- SpaceX has a strong track record in improved success rates. 2017-2018 represents an outlier in the improving success rate for SpaceX
- **The conclusion of the initial visual EDA is that SpaceX launches enjoy improving success rates on landing of the rocket boosters.**
- **A growing share of SpaceX business is now VLEO missions with heavy payloads launched from Cape Canaveral**

All Launch Site Names

- This Python SQL query retrieves from the database a list of SpaceX facilities for further study.
- The use of the ‘DISTINCT’ keyword means only unique locations being retrieved

Task 1

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

```
[9]: %sql SELECT DISTINCT Launch_Site FROM SPACEXTBL;
```

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

t[9]:

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

Launch Site Names Begin with 'CCA'

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	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing_Outcor
2010-04-06	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachu
2010-08-12	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of Brouere cheese	0	LEO (ISS)	NASA (COTS) NRO	Success	Failure (parachu
2012-05-22	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No atten
2012-08-10	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No atten
2013-01-03	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No atten

- The head() function query gives a quick glimpse of the launch locations' activities.
- This query focuses on a subset of the data in the table SPACXTBL. The query retrieves launch sites beginning 'CCA'.
- The query makes use of the SQL wildcard character '%' to match and retrieve any string with 'CCA'

Total Payload Mass

Task 3

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

In [19]:
```
%sql SELECT SUM(PAYLOAD_MASS__KG_) FROM SPACEXTBL WHERE Customer='NASA (CRS)';
```

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

Out[19]:

SUM(PAYLOAD_MASS__KG_)
45596

The query gives insight to the average historical payload requirement for NASA.
This query retrieves rows from the table SPACEXTBL where the customer is 'NASA'. The SQL function SUM() is used to aggregate the payload mass of all flights for this customer, generating a result of 45.596 kg.

Average Payload Mass by F9 v1.1

Task 4

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

```
[21]: %sql SELECT AVG(PAYLOAD_MASS__KG_) FROM SPACEXTBL WHERE Booster_Version='F9 v1.1';
```

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

[21]:

AVG(PAYLOAD_MASS__KG_)
2928.4

This query identifies the typical payload for missions utilizing the Falcon F9 v1.1 booster.
This query retrieves rows from the table SPACEXTBL where the booster type is a Falcon 9 model. The query applies the SQL function AVG() to the “PAYLOAD MASS KG” column to generate an average payload for missions which used the Falcon 9 booster. The average payload was 2928.4 kg

First Successful Ground Landing Date

Task 5

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

Hint:Use min function

[24]:
```
%sql SELECT Min(Date) FROM SPACEXTBL WHERE Landing_Outcome='Success'
```

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

[24]:

Min(Date)
2018-03-12

This query gives context to the history of SpaceX achieving success landings. This query retrieves rows from the table SPACEXTBL where the landing_outcome is deemed a 'Success'.

The query retrieves data from the 'Date' column and by applying the 'MIN' function, filters the results to the earliest date.

Successful Drone Ship Landing with Payload 4000kg-6000kg

Task 6

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

```
[15]: %sql SELECT DISTINCT(Booster_Version) FROM SPACEXTBL WHERE Landing_Outcome='Success (drone ship)' AND PAYLOAD_MASS__KG_> 4
```

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

[15]:

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

The query focuses on the use of reusable rocket boosters improves the economics for a key segment of the market, flights with 4000kg-6000kg payloads for the Geostationary Transfer Orbit (GTO).
The query is formulated to retrieve only unique booster versions, specifically reusable boosters which landed on drone ships.

Total Number of Successful and Failure Mission Outcomes

Task 7

List the total number of successful and failure mission outcomes

[26]:
```
%sql SELECT COUNT(Mission_Outcome) AS 'Mission Successful' FROM SPACEXTBL WHERE Mission_Outcome LIKE 'Success%';
```

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

[26]:

Mission Successful
100

[27]:
```
%sql SELECT COUNT(Mission_Outcome) AS 'Mission Failure' FROM SPACEXTBL WHERE Mission_Outcome LIKE 'Failure%';
```

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

[27]:

Mission Failure
1

The query retrieves the number of missions deemed a success, demonstrating the very high historical rate of flight success for SpaceX, with only 1 failed mission for 100 successful missions in the target period. For SpaceX, the definition of success for a mission is not dependent on the reusable booster landing successfully.

Boosters Carried Maximum Payload

Task 8

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

[17]:
```
%sql SELECT DISTINCT(Booster_Version) FROM SPACEXTBL WHERE PAYLOAD_MASS__KG_ = (SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEX
```

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

[17]:

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

The query's uses a subquery to obtain a list of the boosters which have carried the heaviest payloads

The subquery calculates the maximum payload for the data and the 'DISTINCT' keywork ensures only unique values are retrieved

2015 Launch Records

[52]:
```
%sql SELECT \
    CASE \
        WHEN strftime('%m', DATE) = '01' THEN 'January' \
        WHEN strftime('%m', DATE) = '02' THEN 'February' \
        WHEN strftime('%m', DATE) = '03' THEN 'March' \
        WHEN strftime('%m', DATE) = '04' THEN 'April' \
        WHEN strftime('%m', DATE) = '05' THEN 'May' \
        WHEN strftime('%m', DATE) = '06' THEN 'June' \
        WHEN strftime('%m', DATE) = '07' THEN 'July' \
        WHEN strftime('%m', DATE) = '08' THEN 'August' \
        WHEN strftime('%m', DATE) = '09' THEN 'September' \
        WHEN strftime('%m', DATE) = '10' THEN 'October' \
        WHEN strftime('%m', DATE) = '11' THEN 'November' \
        WHEN strftime('%m', DATE) = '12' THEN 'December' \
    END AS "Month", \
    Booster_Version, \
    Launch_Site, \
    Landing_Outcome \
FROM SPACEXTBL \
WHERE DATE LIKE '2015-%' AND Landing_Outcome = 'Failure (drone ship)';
```

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

[52]:

Month	Booster_Version	Launch_Site	Landing_Outcome
October	F9 v1.1 B1012	CCAFS LC-40	Failure (drone ship)
April	F9 v1.1 B1015	CCAFS LC-40	Failure (drone ship)

The query recovers the launches by SpaceX in 2015.

Because the query is SQL has problems with retrieving dates, code is required to elaborate the month names

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.

[53]:
```
%sql SELECT DISTINCT Landing_Outcome, COUNT(*) AS Result FROM SPACEXTBL \
WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20' \
GROUP BY Landing_Outcome ORDER BY Result DESC;
```

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

[53]:

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

The query illustrates the frequency of distinct landing outcomes for a specified date range.

The results are filtered and groups and then ordered in descending order of occurrence.

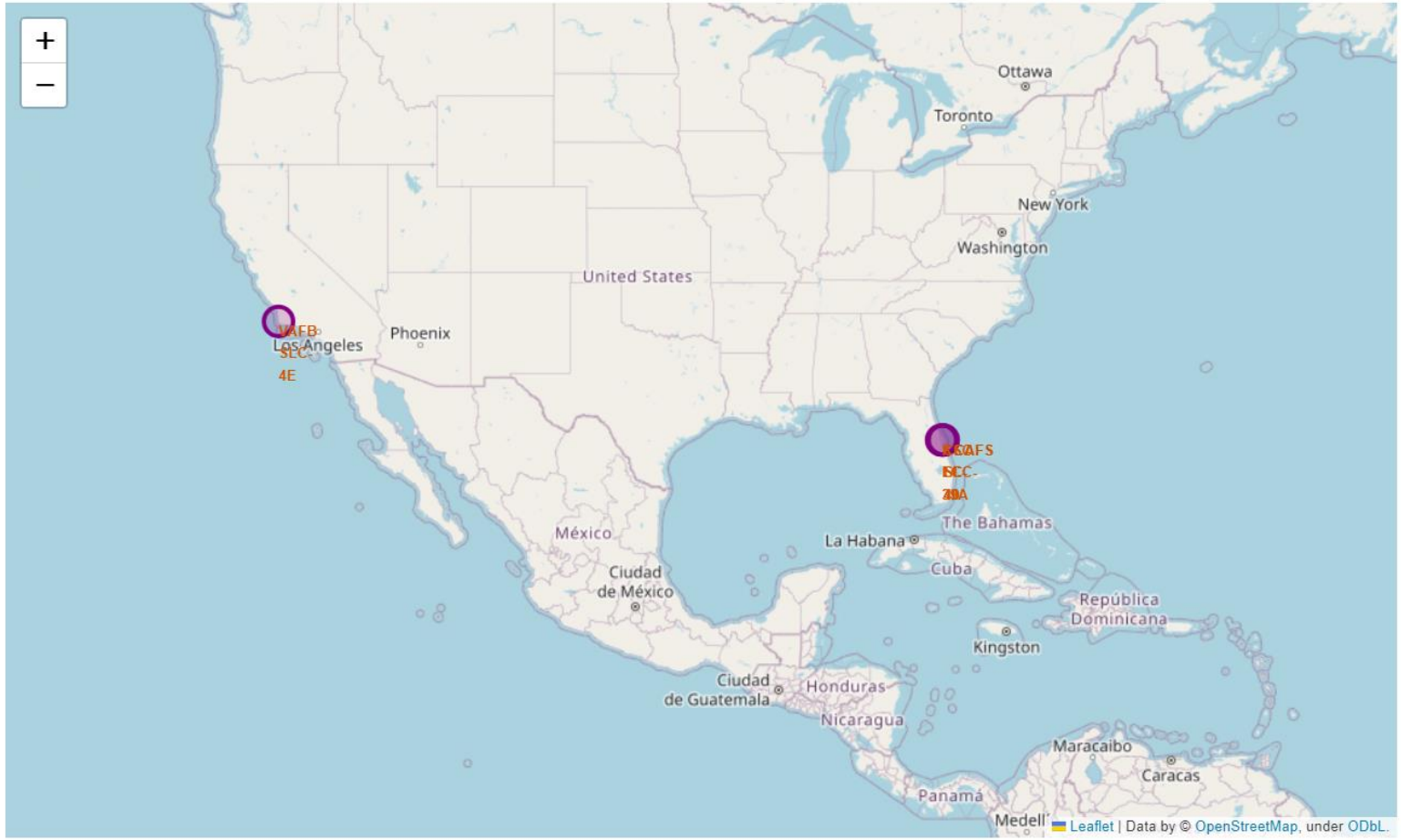
Section 3

Launch Sites Proximities Analysis

All Launch Sites

- Launch sites are located closer to the equator and adjacent to the coast and less inhabited nature reserves
- The site location is because most unsuccessful landings are planned, where Space X performs a controlled landing in the ocean
- SpaceX has launched from the East Coast and West Coast with three locations / four launch pads adjacent to

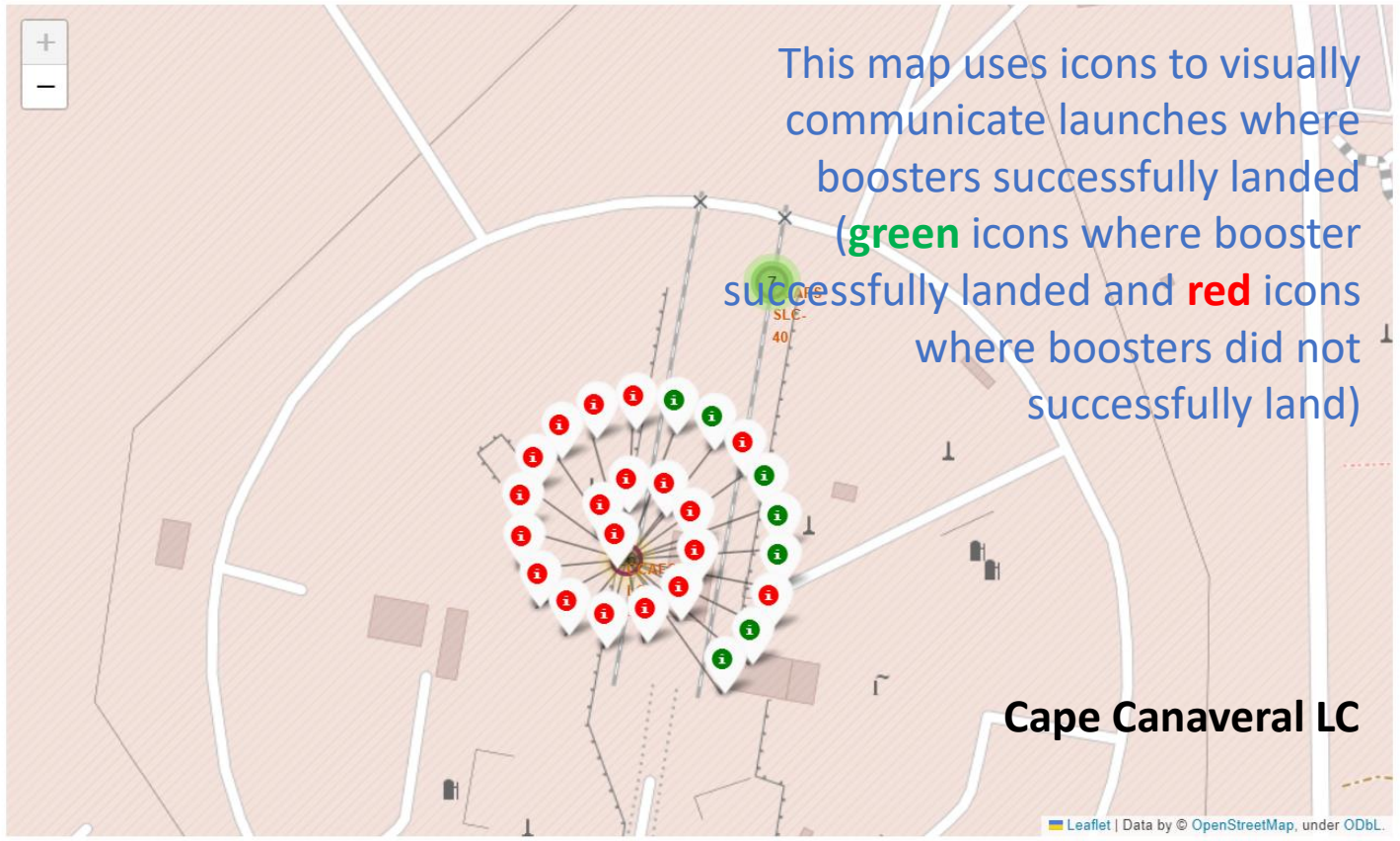
[10]:

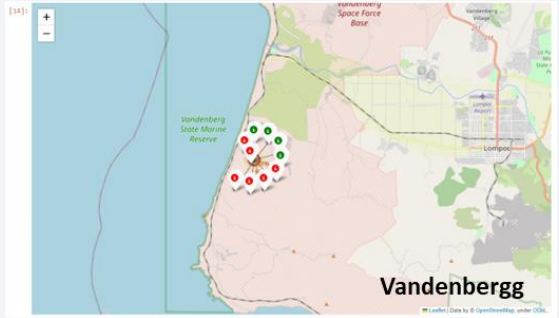


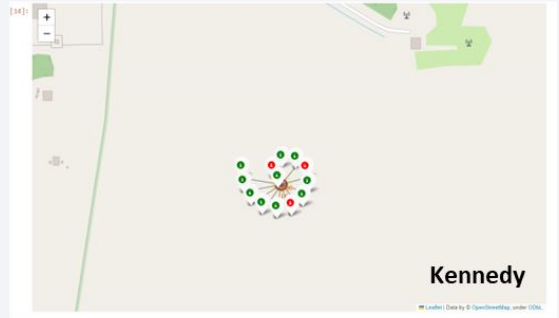
Cluster Visualization for Successful Booster Retrieval

```
for index, row in spacex_df.iterrows():
    # create and add a Marker cluster to the site map
    coordinate = [row['Lat'], row['Long']]
    folium.map.Marker(coordinate, icon=folium.Icon(color='white',icon_color=row['marker_color'])).add_to(marker_cluster)
site_map
```

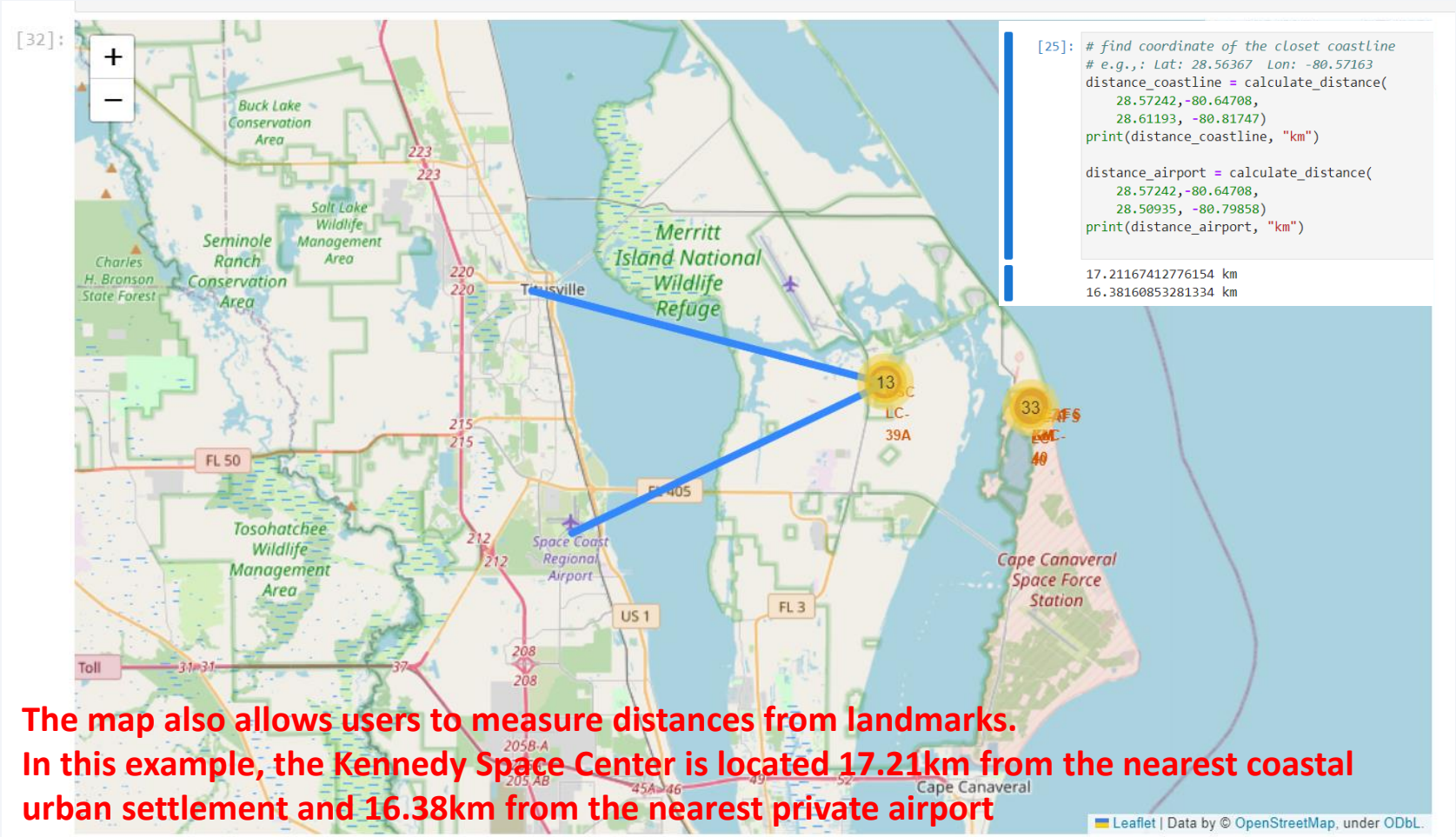
[14]:



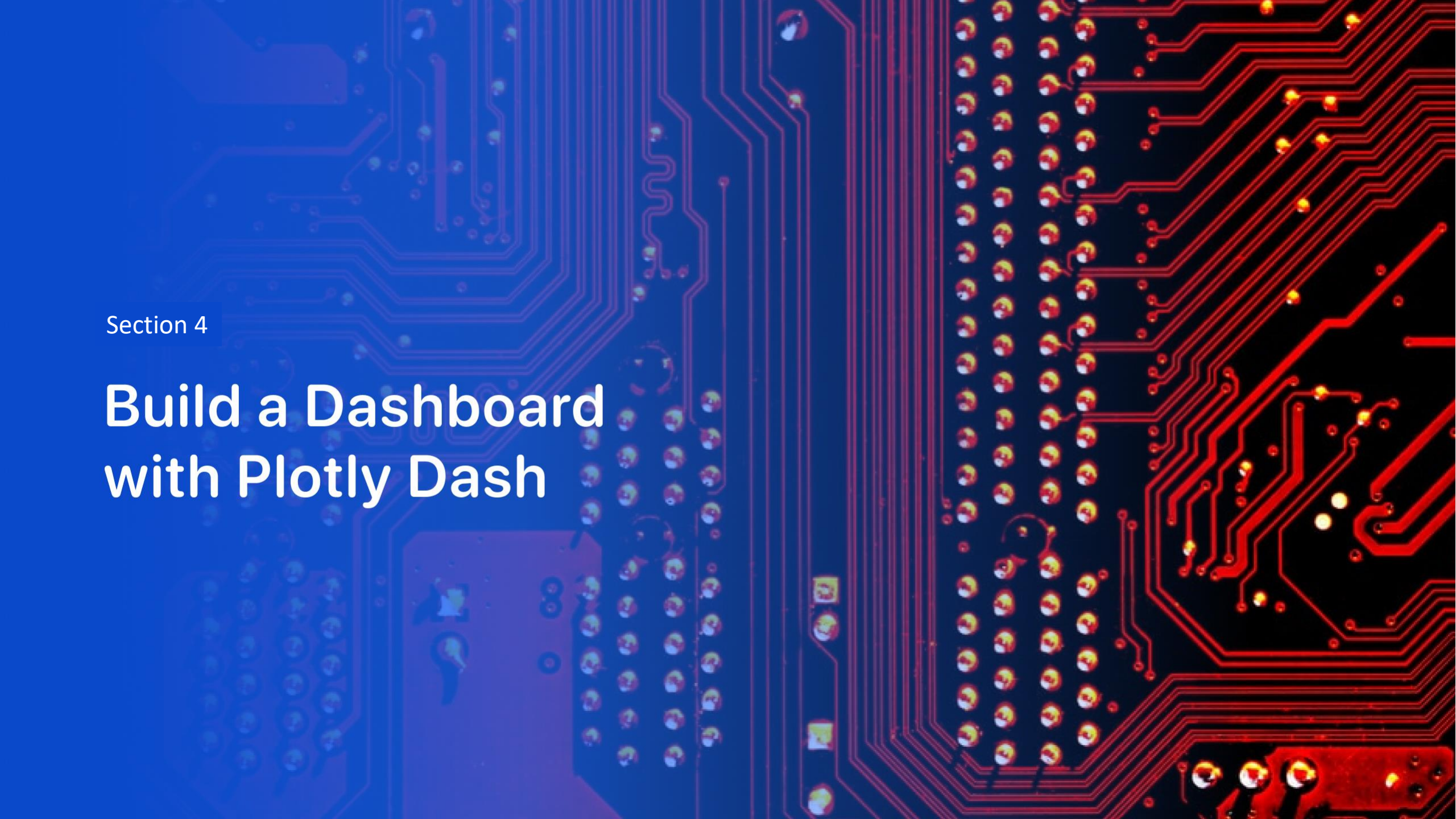




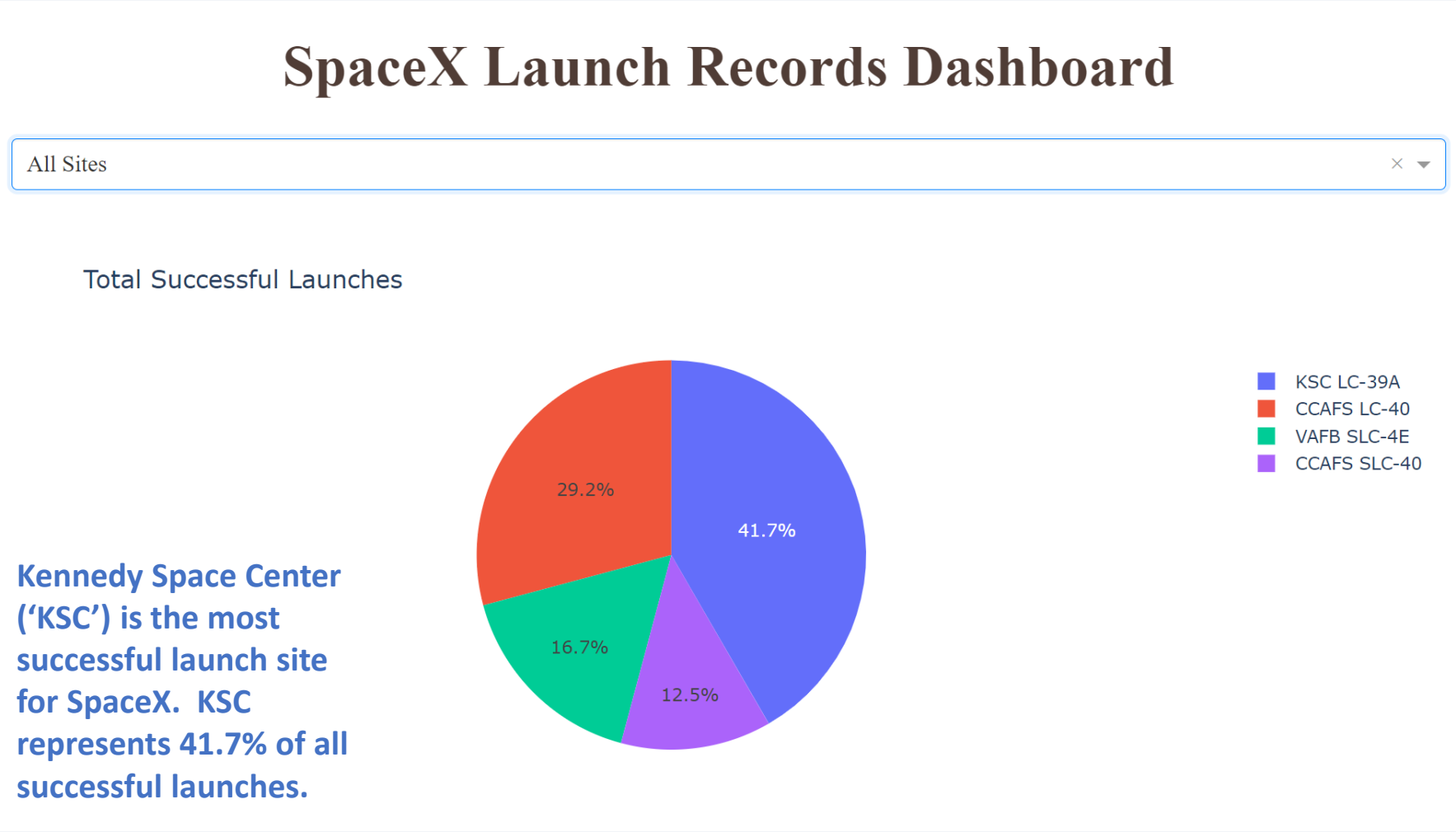
Launch Site Proximity to Urban Locations/Infrastructure



The map also allows users to measure distances from landmarks.
In this example, the Kennedy Space Center is located 17.21km from the nearest coastal urban settlement and 16.38km from the nearest private airport

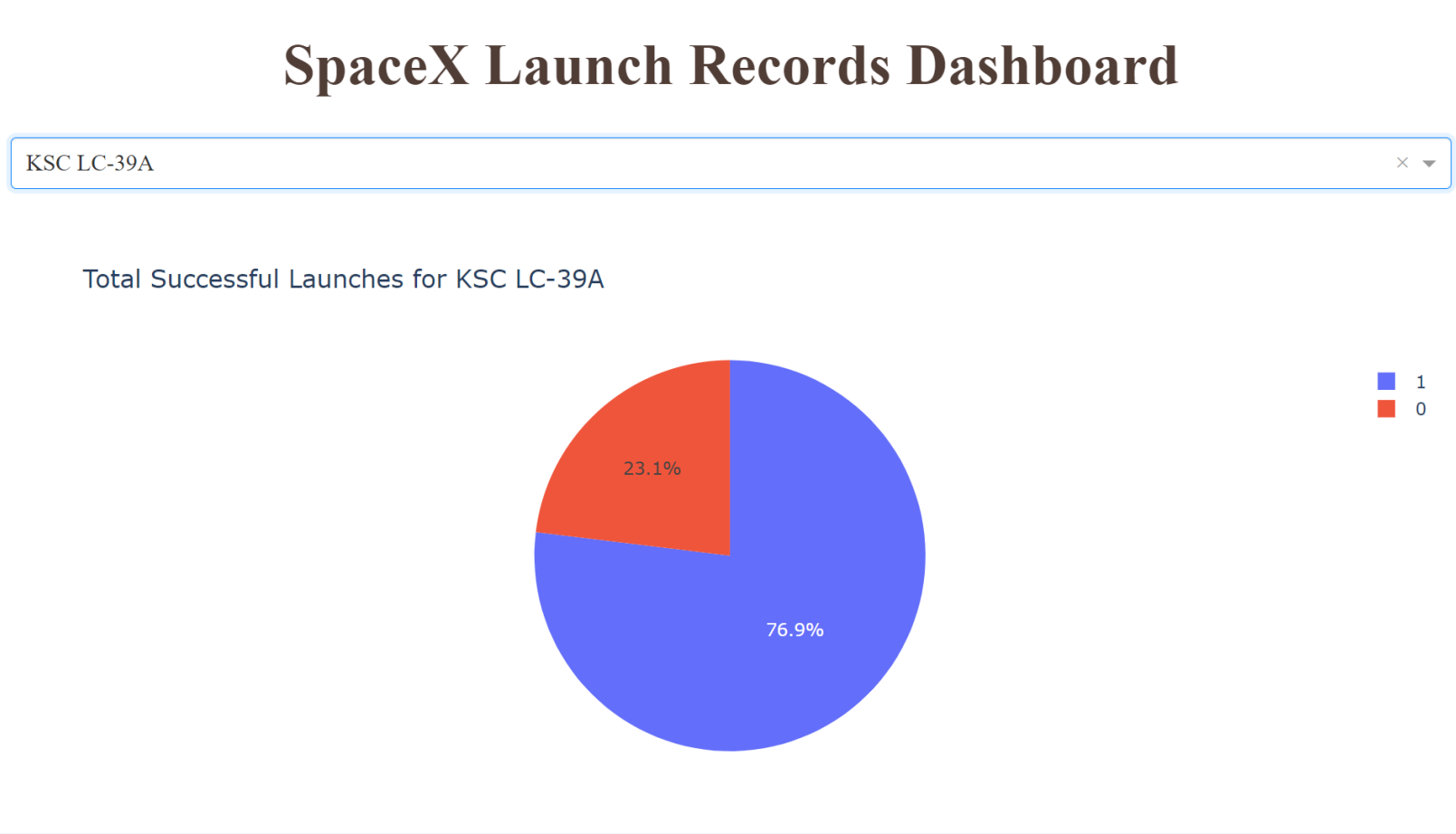

Section 4
Build a Dashboard with Plotly Dash

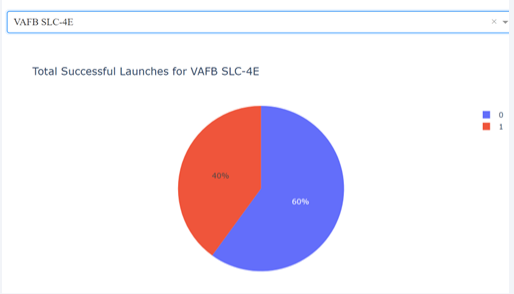
Launch Success by Site

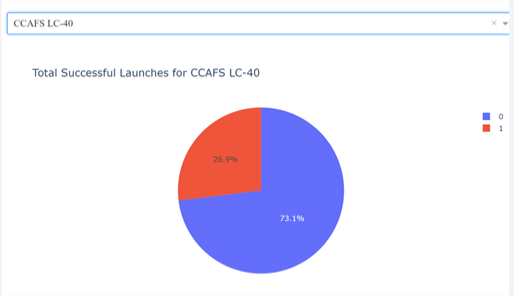


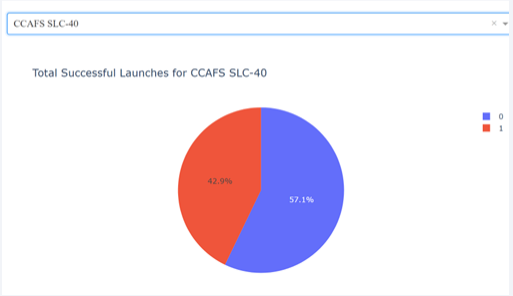
Kennedy Space Center ('KSC') is the most successful launch site for SpaceX. KSC represents 41.7% of all successful launches.

Launch Success by Site









To date, KSC has a success rate of over 76% for launches compared to 73% for CC-LC 40, 60% for VAFB and 57% for CC-SLC-40

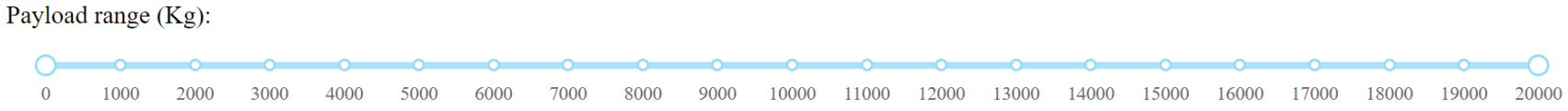
Launch Success by Payload, All Sites

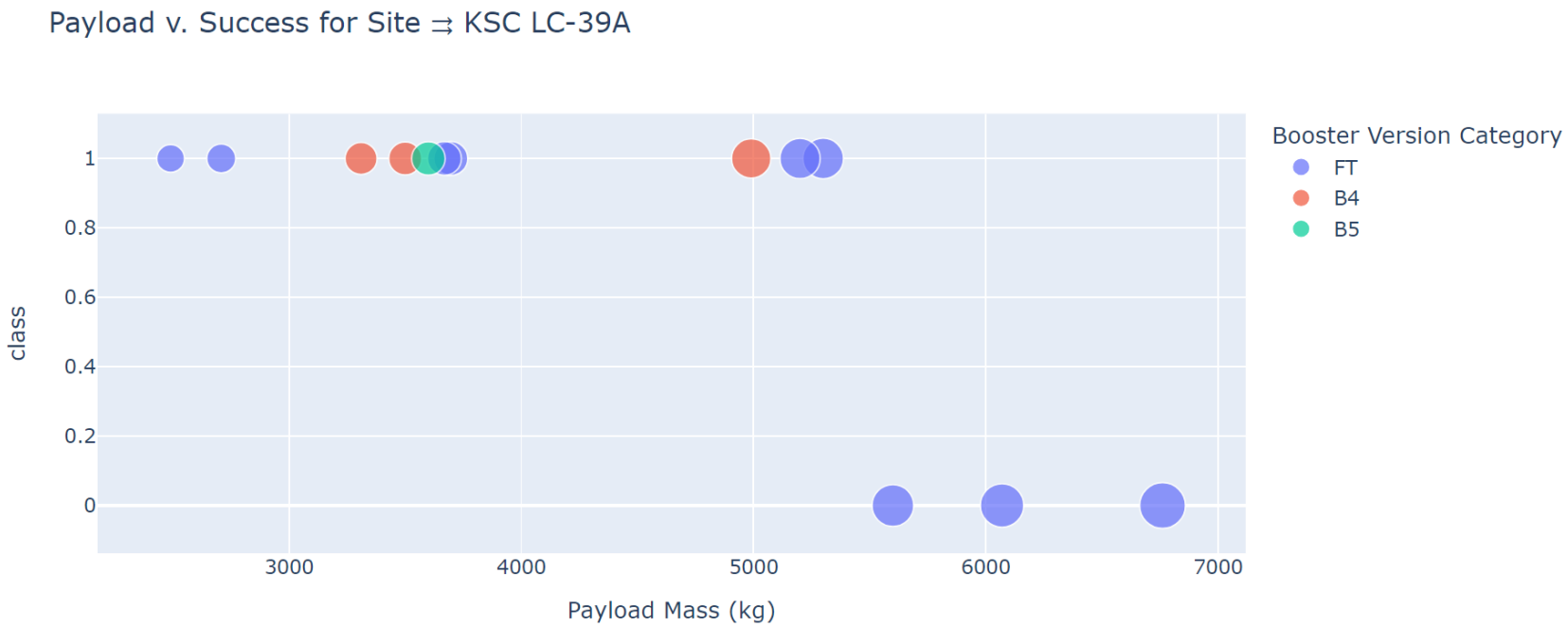
SpaceX historically had less success with larger payloads, recording significantly more failed launches than successful launches with large payloads. However,

All Sites: Payload > 4000 kg. v. Success

All Sites: Payload < 4000 kg. v. Success

Launch Success by Payload & by Site

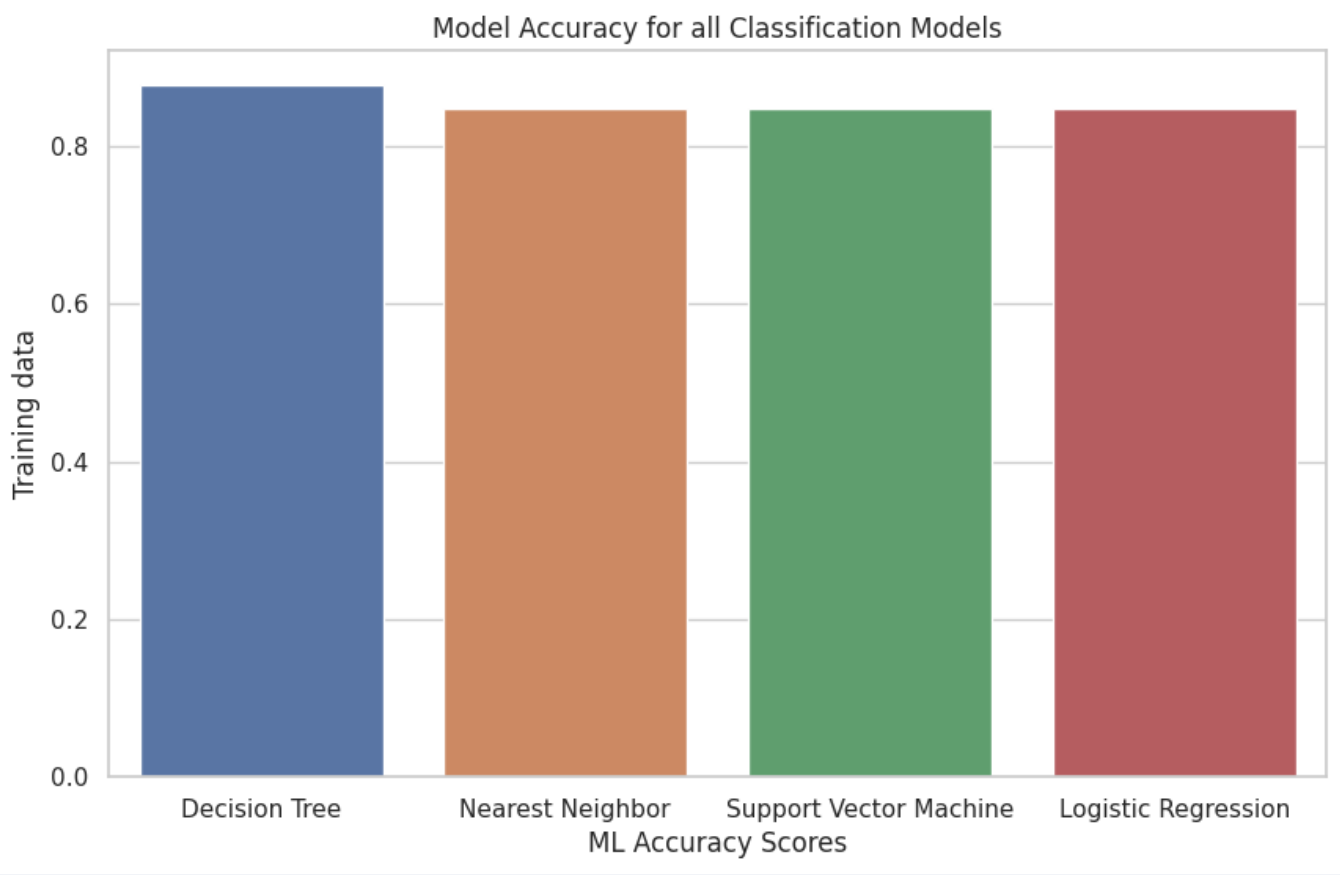




Section 5

Predictive Analysis (Classification)

Classification Accuracy



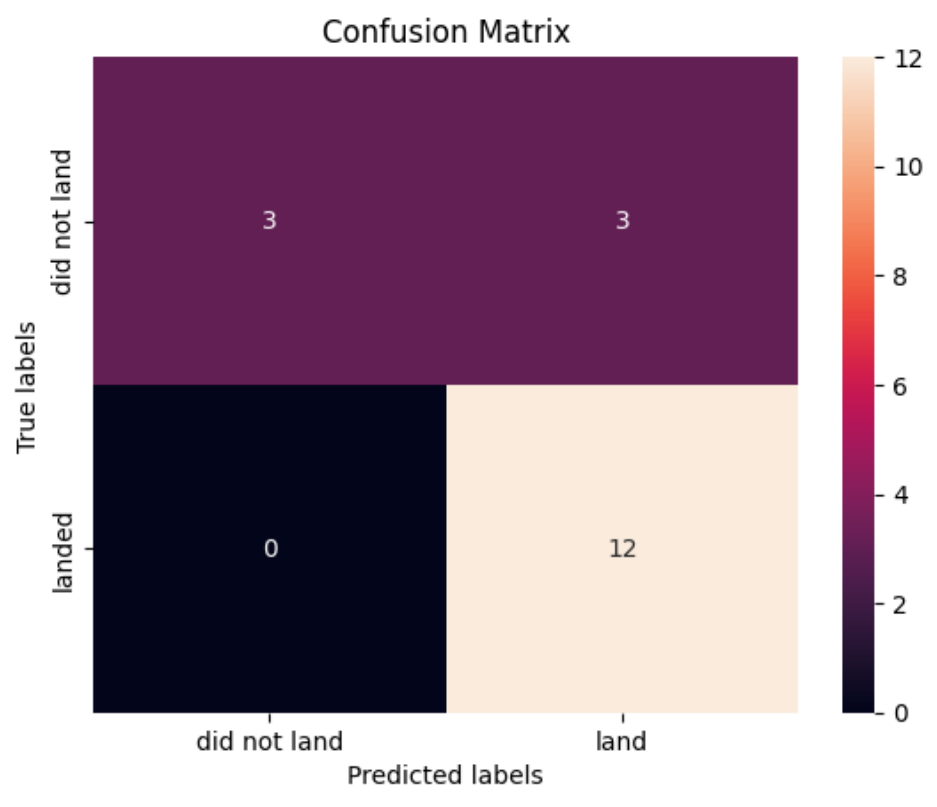
The best method is Decision Tree

	ML Accuracy Scores	Training data	Testing data
2	Decision Tree	0.876786	0.833333
3	Nearest Neighbor	0.848214	0.833333
1	Support Vector Machine	0.848214	0.833333
0	Logistic Regression	0.846429	0.833333

The best performing method was the decision tree method.

The decision tree model gave a 0.87 result on training data compared to just under 0.85 for SVM, KKN and logistic regression models.

Confusion Matrix



Sensitivity: 1.0
Specificity: 0.5
Precision: 0.8
Accuracy: 0.83

F1 Score: 0.89
False Positive Rate: 0.5
False Discovery rate: 0.2

From the confusion matrix, we can infer:

- The Decision Tree classifiers performed well in predicting positive/negatives classes because the number of true positives and true negatives is relatively high.
- The 3 false negative indicate instances where the classifier missed/misclassified negatives
- The number of false positives (0) demonstrates that the models did not classify any instances as positive which were negative

The confusion matrix for linear regression, SMV and K nearest neighbor models had similarly balanced accuracy of 0.75 and these models scored identically on test data.

Conclusions

- Revenue Drivers:
  - Exploratory Data Analysis of SpaceX data demonstrates that the most commercially important business segment for SpaceY will be the GTO orbit with payloads of 3000kg-7000kg
- Cost Drivers:
  - The optimum launch location is Florida (Kennedy Space Center/Cape Canaveral), and the Decision Tree model can be applied to predict at a high accuracy the failure rates (i.e. losses on capital equipment) for flights to different orbits with payloads
- Next Steps
  - The results can be passed on to the business planning and finance teams who can generate revenue and cost forecasts from market data which in turn can demonstrate the potential profitability and financial feasibility of SpaceY

Appendix: Orbit Categories

- **LEO**: Low Earth orbit (LEO)is an Earth-centred orbit with an altitude of 2,000 km (1,200 mi) or less (approximately one-third of the radius of Earth),[1] or with at least 11.25 periods per day (an orbital period of 128 minutes or less) and an eccentricity less than 0.25.[2] Most of the manmade objects in outer space are in LEO [1].
- **VLEO**: Very Low Earth Orbits (VLEO) can be defined as the orbits with a mean altitude below 450 km. Operating in these orbits can provide a number of benefits to Earth observation spacecraft as the spacecraft operates closer to the observation[2].
- **GTO** A geosynchronous orbit is a high Earth orbit that allows satellites to match Earth's rotation. Located at 22,236 miles (35,786 kilometers) above Earth's equator, this position is a valuable spot for monitoring weather, communications and surveillance. Because the satellite orbits at the same speed that the Earth is turning, the satellite seems to stay in place over a single longitude, though it may drift north to south," NASA wrote on its Earth Observatory website [3] .
- **SSO (or SO)**: It is a Sun-synchronous orbit also called a heliosynchronous orbit is a nearly polar orbit around a planet, in which the satellite passes over any given point of the planet's surface at the same local mean solar time [4] .
- **ES-L1** :At the Lagrange points the gravitational forces of the two large bodies cancel out in such a way that a small object placed in orbit there is in equilibrium relative to the center of mass of the large bodies. L1 is one such point between the sun and the earth [5] .
- **HEO** A highly elliptical orbit, is an elliptic orbit with high eccentricity, usually referring to one around Earth [6].
- **ISS** A modular space station (habitable artificial satellite) in low Earth orbit. It is a multinational collaborative project between five participating space agencies: NASA (United States), Roscosmos (Russia), JAXA (Japan), ESA (Europe), and CSA (Canada) [7]
- **MEO** Geocentric orbits ranging in altitude from 2,000 km (1,200 mi) to just below geosynchronous orbit at 35,786 kilometers (22,236 mi). Also known as an intermediate circular orbit. These are "most commonly at 20,200 kilometers (12,600 mi), or 20,650 kilometers (12,830 mi), with an orbital period of 12 hours [8]
- **HEO** Geocentric orbits above the altitude of geosynchronous orbit (35,786 km or 22,236 mi) [9]
- **GEO** It is a circular geosynchronous orbit 35,786 kilometres (22,236 miles) above Earth's equator and following the direction of Earth's rotation [10]
- **PO** It is one type of satellites in which a satellite passes above or nearly above both poles of the body being orbited (usually a planet such as the Earth [11]

Appendix

Data Collection: Launch df.

In [22]:
```
# Create a data from launch_dict
launch_df = pd.DataFrame(launch_dict)
```

Show the summary of the dataframe

In [23]:
```
# Show the head of the dataframe
launch_df.head()
```

Out[23]:

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block
0	1	2006-03-24	Falcon 1	20.0	LEO	Kwajalein Atoll	None None	1	False	False	False	None	NaN
1	2	2007-03-21	Falcon 1	NaN	LEO	Kwajalein Atoll	None None	1	False	False	False	None	NaN
2	4	2008-09-28	Falcon 1	165.0	LEO	Kwajalein Atoll	None None	1	False	False	False	None	NaN
3	5	2009-07-13	Falcon 1	200.0	LEO	Kwajalein Atoll	None None	1	False	False	False	None	NaN
4	6	2010-06-04	Falcon 9	NaN	LEO	CCSFS SLC 40	None None	1	False	False	False	None	1.0

Webscraping: Launch dictionary

To simplify the parsing process, we have provided an incomplete code snippet below to help you to fill up the `launch_dict`. Please complete the following code snippet with TODOs or you can choose to write your own logic to parse all launch tables:

```
extracted_row = 0
#Extract each table
for table_number,table in enumerate(soup.find_all('table',"wikitable plainrowheaders collapsible")):
   # get table row
    for rows in table.find_all("tr"):
        #check to see if first table heading is as number corresponding to launch a number
        if rows.th:
            if rows.th.string:
                flight_number=rows.th.string.strip()
                flag=flight_number.isdigit()
        else:
            flag=False
        #get table element
        row=rows.find_all('td')
        #if it is number save cells in a dictonary
        if flag:
            extracted_row += 1
            # Flight Number value
            # TODO: Append the flight_number into launch_dict with key `Flight No.`
            print(flight_number)
            datatimelist=date_time(row[0])

            # Date value
            # TODO: Append the date into launch_dict with key `Date`
            date = datatimelist[0].strip(',')
            print(date)

            # Time value
            # TODO: Append the time into launch_dict with key `Time`
            time = datatimelist[1]
            print(time)

            # Booster version
            # TODO: Append the bv into launch_dict with key `Version Booster`
            bv=booster_version(row[1])
```

Appendix

Data wrangling: Class labels

```
bad_outcomes=set(landing_outcomes.keys()[[1,3,5,6,7]])
bad_outcomes
```

```
{'False ASDS', 'False Ocean', 'False RTLS', 'None ASDS', 'None None'}
```

TASK 4: Create a landing outcome label from Outcome column

Using the `Outcome`, create a list where the element is zero if the corresponding row in `Outcome` is in the set `bad_outcome`; otherwise, it's one. Then assign it to the variable `landing_class`:

```
# landing_class = 0 if bad_outcome
# landing_class = 1 otherwise

landing_class = [0 if outcome in bad_outcomes else 1 for outcome in df['Outcome']]
print(landing_class)
```

```
[0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]
```

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

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

	Class
0	0
1	0
2	0

Launch Locations: Applying folium.Markers()

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

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

Above coordinates are just plain numbers that can not give you any intuitive insights about where are those launch sites. If you are very good at geography, you can interpret those numbers directly in your mind. If not, that's fine too. Let's visualize those locations by pinning them on a map.

We first need to create a folium `Map` object, with an initial center location to be NASA Johnson Space Center at Houston, Texas.

```
# Start location is NASA Johnson Space Center
nasa_coordinate = [29.559684888503615, -95.0830971930759]
site_map = folium.Map(location=nasa_coordinate, zoom_start=10)
```

We could use `folium.Circle` to add a highlighted circle area with a text label on a specific coordinate. For example,

```
# Create a blue circle at NASA Johnson Space Center's coordinate with a popup label showing its name
circle = folium.Circle(nasa_coordinate, radius=1000, color='#d35400', fill=True).add_child(folium.Popup('NASA Johnson Sp
# Create a blue circle at NASA Johnson Space Center's coordinate with a icon showing its name
marker = folium.map.Marker(
    nasa_coordinate,
    # Create an icon as a text label
    icon=DivIcon(
        icon_size=(20,20),
        icon_anchor=(0,0),
        html='<div style="font-size: 12; color:#d35400;"><b>%s</b></div>' % 'NASA JSC',
        )
    )
site_map.add_child(circle)
site_map.add_child(marker)
```

Appendix

Machine Learning: Training Decision Tree model

Create a decision tree classifier object then create a `GridSearchCV` object `tree_cv` with cv = 10. Fit the object to find the best parameters from the dictionary `parameters`.

In [22]:
```
parameters_tr = {'criterion': ['gini', 'entropy'],
     'splitter': ['best', 'random'],
     'max_depth': [2*n for n in range(1,10)],
     'max_features': ['auto', 'sqrt'],
     'min_samples_leaf': [1, 2, 4],
     'min_samples_split': [2, 5, 10]}
def warn(*args, **kwargs):
    pass
import warnings
warnings.warn = warn
tree = DecisionTreeClassifier()
```

In [23]:
```
tg_cv = GridSearchCV(estimator=tree, param_grid=parameters_tr, cv=10)
tree_cv = tg_cv.fit(X_train, Y_train)
```

In [24]:
```
print("tuned hpyerparameters :(best parameters) ", tree_cv.best_params_)
print("accuracy :", tree_cv.best_score_)
```

```
tuned hpyerparameters :(best parameters)  {'criterion': 'entropy', 'max_depth': 12, 'max_features': 'sqrt', 'min_samples_lea
f': 4, 'min_samples_split': 2, 'splitter': 'random'}
accuracy : 0.8767857142857143
```

Dash App: Creating scatterplot

```
@app.callback(
    Output(component_id='success-payload-scatter-chart', component_property='figure'),
    [Input(component_id='entered_site', component_property='value'),
     Input(component_id='payload-slider', component_property='value')]
)
def get_scatter_chart(entered_site, slider_range):
    low, high = slider_range
    payload_range = (spacex_df['Payload Mass (kg)'] > low) & (spacex_df['Payload Mass (kg)'] < high)
    scatter_df = spacex_df[payload_range]
    if entered_site == 'ALL':
        fig = px.scatter(
            scatter_df,
            x="Payload Mass (kg)",
            y="class",
            title='Payload v. Success: All Sites',
            color="Booster Version Category",
            size='Payload Mass (kg)',
            hover_data=['Booster Version', 'Payload Mass (kg)']
        )
        return fig
    else:
        scatter_df = scatter_df[scatter_df['Launch Site'] == entered_site]
        fig = px.scatter(
            scatter_df,
            x="Payload Mass (kg)",
            y="class",
            title='Payload v. Success for Site ⇉ ' + entered_site,
            color="Booster Version Category",
            size='Payload Mass (kg)',
            hover_data=['Booster Version', 'Payload Mass (kg)']
        )
        return fig
```

Thank you!