SpaceY's Key to Winning the Space Race: Data Science

A presentation by Chris Sinatra
May 24th, 2023

Outline

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

Executive Summary

- Summary of methodologies
  - API Calls
  - Web-Scraping
  - Data Wrangling
  - Exploratory Data Analysis with Data Visualization
  - Exploratory Data Analysis with SQL
  - Interactive Map Construction via Folium
  - Interactive Dashboard Construction via Plotly
  - Machine Learning Algorithm Training

- Summary of all results
  - Data Visualization/SQL Outcomes
  - Map and Dashboard Outcomes
  - Machine Learning Outcomes

Introduction

- SpaceX is making strides in the spacecraft industry via its reusable rocket stages. Most of the time, its first rocket stage is recoverable and reusable, drastically reducing price and time for future launches. You, SpaceY, the new rival company, are interested in predicting the outcomes of SpaceX missions and landings and adopting some of the successful variable combinations for yourselves

- Using data science tools, SpaceY will be able to determine:
  - Is SpaceX's first rocket stage recoverable? Will it land properly and be reusable, saving SpaceX money?
  - What rocket parameters influence whether a launch or landing is successful?

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - API Calls and Web-scraping were used to pull data from SpaceX, IBM, and Wikipedia
- Perform data wrangling
  - Exploring the data types allowed me to change the launch outcomes into a new 'Class' category, utilizing one-hot encoding to allow for later data manipulation
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - Using GridSearchCV, fit_transform(), and fit() on various Machine Learning models, parameters which maximized the R^2 value were determined. Confusion matrices were also created

Data Collection

- API calls were made via requests to datasets of SpaceX and IBM. Responses were uploaded into data frames. Info included were rocket booster versions, cores, payloads, and launchpad details
- An API call to obtain launch data resulted in a json file which was then normalized and converted into a more easily readable Pandas data frame.
- All of the data obtained via API calls were consolidated into a single data frame (with missing values of 'Payload Mass' filled with the average payload mass.
- Additional launch data was web-scraped from Wikipedia by utilizing the HTTP get function and BeautifulSoup. Data were converted into a Pandas data frame

Data Collection – SpaceX API

requests.get() was used to obtain data from the API

The response was appended into empty data frames

For each category of interest, a function was defined that would obtain its data then append it into a data frame

```
# 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, a

```
# 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.

```
# 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 f
block of the core which is a number used to seperate version of cores, the number of times this speci

```
# 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'])
```

- The GitHub URL of the completed SpaceX API calls notebook is here for your convenience:

https://github.com/CSinatra26/CSinatra_IBM_Data_Science_Capstone_Final_Presentation/blob/main/1_data-collection-api.ipynb

Data Collection – SpaceX API (json file)

requests.get() was again used to obtain data from the API

This time, the response was converted into a Pandas data frame using the .json_normalize() function

The first 5 entries in the data frame could be analyzed using the .head() function

Data was eventually added to the prior API calls. All data were consolidated into one data frame

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

[8]: response = requests.get(spacex_url)
```

Check the content of the response

```
[10]: print(response)
```

```
<Response [200]>
```

You should see the response contains massive information about SpaceX launches. Next, let's try to discover some more relevant information for this

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

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

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

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

```
[125]: response.status_code

[125]: 200
```

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

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

Using the dataframe `data` print the first 5 rows

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

[127]:

	static_fire_date_utc	static_fire_date_unix	net	window	rocket	success	failures	details	crew	ships	capsules	
0	2006-03-17T00:00:00.000Z	1.142554e+09	False	0.0	5e9d0d95eda69955f709d1eb	False	[{'time': 33, 'altitude': None, 'reason': 'merlin engine failure'}]	Engine failure at 33 seconds and loss of vehicle	[]	[]	[]	[5eb0e4b5b6c3bb00

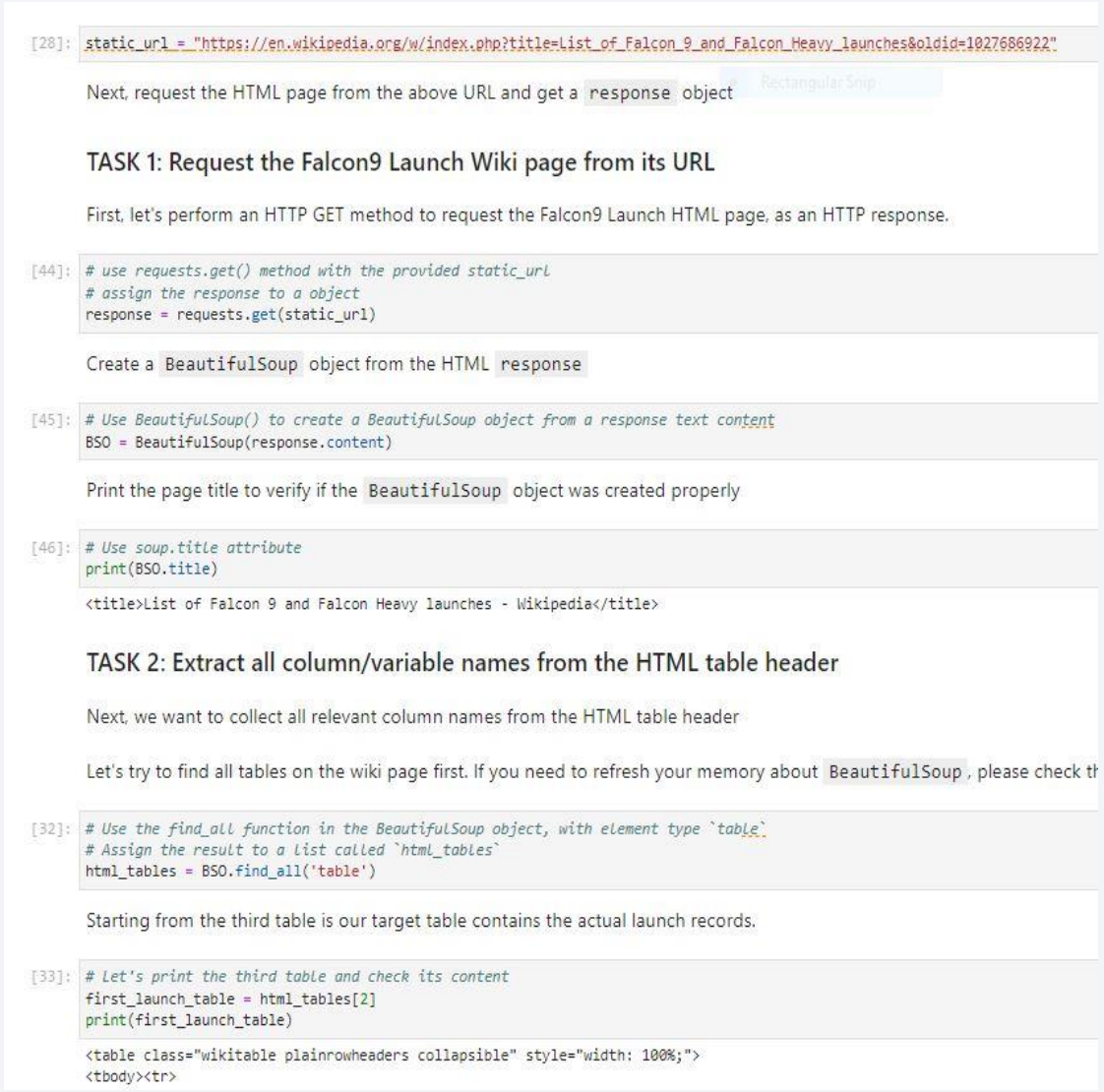
Data Collection - Scraping

requests.get() was used in conjunction with BeautifulSoup to extract information from the SpaceX Falcon 9 Launch details on Wikipedia

The response was then parsed through by searching for all present 'th', signifying headers

The headers and their data were later extracted using a collection of given functions and stored in a dictionary

The dictionary was converted into a Pandas data frame



[28]:
```
static_url = "https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922"
```

Next, request the HTML page from the above URL and get a `response` object

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

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

[44]:
```
# use requests.get() method with the provided static_url
# assign the response to a object
response = requests.get(static_url)
```

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

[45]:
```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
BSO = BeautifulSoup(response.content)
```

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

[46]:
```
# Use soup.title attribute
print(BSO.title)
```

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

TASK 2: Extract all column/variable names from the HTML table header

Next, we want to collect all relevant column names from the HTML table header

Let's try to find all tables on the wiki page first. If you need to refresh your memory about `BeautifulSoup`, please check th

[32]:
```
# Use the find_all function in the BeautifulSoup object, with element type `table`
# Assign the result to a list called `html_tables`
html_tables = BSO.find_all('table')
```

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

[33]:
```
# Let's print the third table and check its content
first_launch_table = html_tables[2]
print(first_launch_table)
```

```
<table class="wikitable plainrowheaders collapsible" style="width: 100%;">
<tbody><tr>
```

- The GitHub URL of the completed web scraping notebook is here for your convenience:

https://github.com/CSinatra26/CSinatra_IBM_Data_Science_Capstone_Final_Presentation/blob/main/2_webscraping.ipynb

Data Wrangling (One-Hot Encoding)

Using .value_counts() and a subsequent .keys() on the 'Outcomes' category, the different types of landing outcomes were determined

The landing outcomes were separated into two different groups (successful and failed)

Using one-hot encoding, the landing outcomes were assigned either a '1' (successful landing) or '0' (failed landing) for a new column 'Class'

The data frame was then appended with the new 'Class' column

```
[11]: for i,outcome in enumerate(landing_outcomes.keys()):
          print(i,outcome)

0 True ASDS
1 None None
2 True RTLS
3 False ASDS
4 True Ocean
5 False Ocean
6 None ASDS
7 False RTLS
```

We create a set of outcomes where the second stage did not land successfully:

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

[12]: {'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 `Outco`

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

This variable will represent the classification variable that represents the outcome of each launc

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

[14]:

	Class
0	0
1	0
2	0
3	0
4	0

- The GitHub URL of the completed data wrangling related notebooks is here for your convenience:

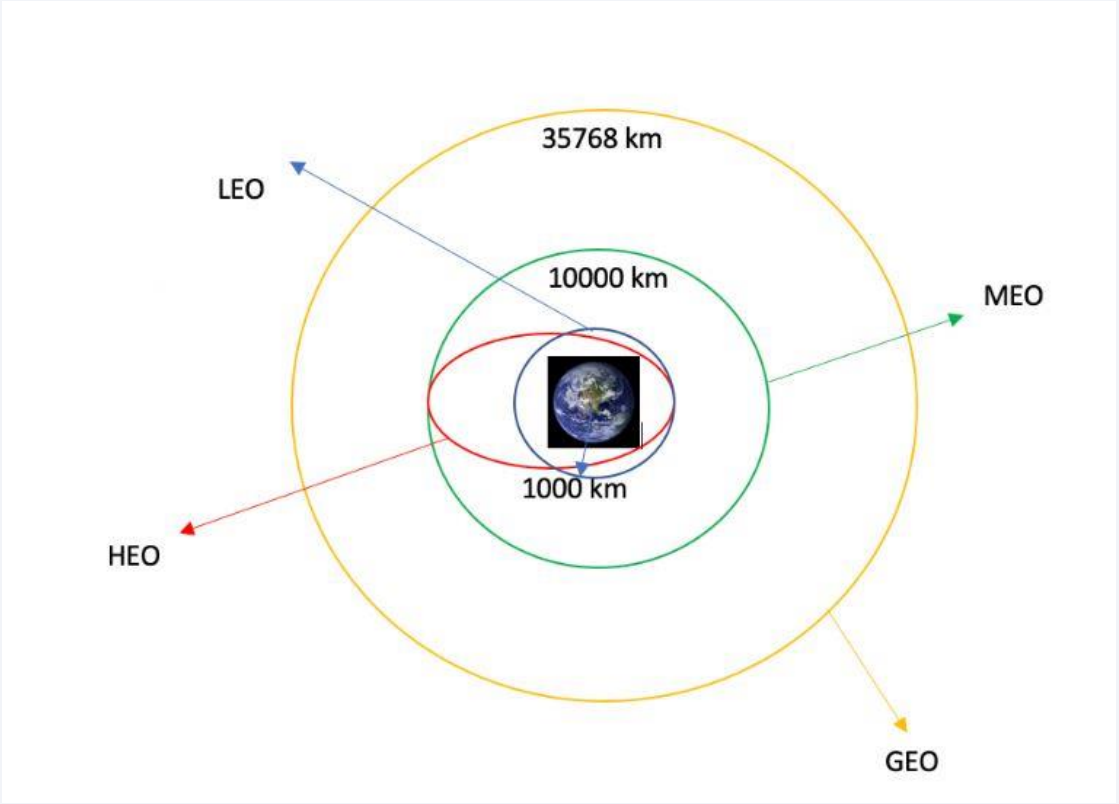
https://github.com/CSinatra26/CSinatra_IBM_Data_Science_Capstone_Final_Presentation/blob/main/3_data_wrangling.ipynb

EDA with Data Visualization (Orbit Success Rate)

A bar chart was used to conveniently visualize success rates of landings from different orbits. Orbit is a key factor in determining landing success or failure

A line graph was used to conveniently visualize success rates of landings by year. Determining trends year-to-year may be influential in determining chance of future success

Additional scatter plots were made comparing categories such as orbit, flight number, launch site, and payload mass



- The GitHub URL of the completed EDA with data visualization notebook is here for your convenience:

https://github.com/CSinatra26/CSinatra_IBM_Data_Science_Capstone_Final_Presentation/blob/main/5_eda_data_visualization.ipynb

EDA with SQL

- By loading and utilizing SQL through a Jupyter notebook, multiple queries involving payload mass, booster versions, dates, and landing success were performed
- Some of the queries included:
  - Average payload mass carried by the Falcon 9 v1.1
  - First successful ground pad landing date
  - Total number of successful missions versus failed missions
  - Boosters that carried the maximum payload mass
  - Boosters that carried payloads within a range of 4000kg and 6000kg
- The GitHub URL of the completed EDA with SQL notebook is here for your convenience: https://github.com/CSinatra26/CSinatra_IBM_Data_Science_Capstone_Final_Presentation/blob/main/4_eda_using_sql.ipynb

Build an Interactive Map with Folium

- Using a Folium map:
  - Circular markers were used to identify four specific SpaceX launch sites that launch F9 rockets
  - Colored markers were used to denote launch success (green) or failure (red)
  - Marker clusters were used to neatly condense the many markers into the vicinity of each site
  - Additional markers were placed on landmarks of interest (highways, railways, coastlines, and cities) and lines were used to measure the distance between launch sites and said landmarks
- Visualizing the successes and failures of each launch site via a Folium map is helpful for making more informed predictions about launches from specific sites
- The GitHub URL of the completed interactive map with Folium map is here for your convenience: https://github.com/CSinatra26/CSinatra_IBM_Data_Science_Capstone_Final_Presentation/blob/main/6_launch_sites_folium.ipynb

Build a Dashboard with Plotly Dash

- Using Plotly a dashboard was created that allows the user to toggle different launch sites and payload weights to determine their effects on launch success
  - A dropdown selection bar was included to allow for selection of a specific launch site or the 'All Sites' option
  - A slider was included in order to adjust the payload weight that the plots would include data for, displaying the difference in success for various weight ranges
  - A pie chart was included to display different launch site success rates (or percentage of total success per site for the 'All Sites' selection)
  - A scatter plot was added to help visualize successes of different booster versions across different payload weights, further helping to elucidate possible barriers to a successful launch amongst a previously unexplored variable relationship
  - The GitHub URL of the completed Plotly Dash lab is here for your convenience: https://github.com/CSinatra26/CSinatra_IBM_Data_Science_Capstone_Final_Presentation/blob/main/7_spacex_dash_app.py

Predictive Analysis (Classification)

- The GitHub URL of the completed predictive analysis lab is here for your convenience: https://github.com/CSinatra26/CSinatra_IBM_Data_Science_Capstone_Final_Presentation/blob/main/8_machine_learning_model_predeictions.ipynb

Results

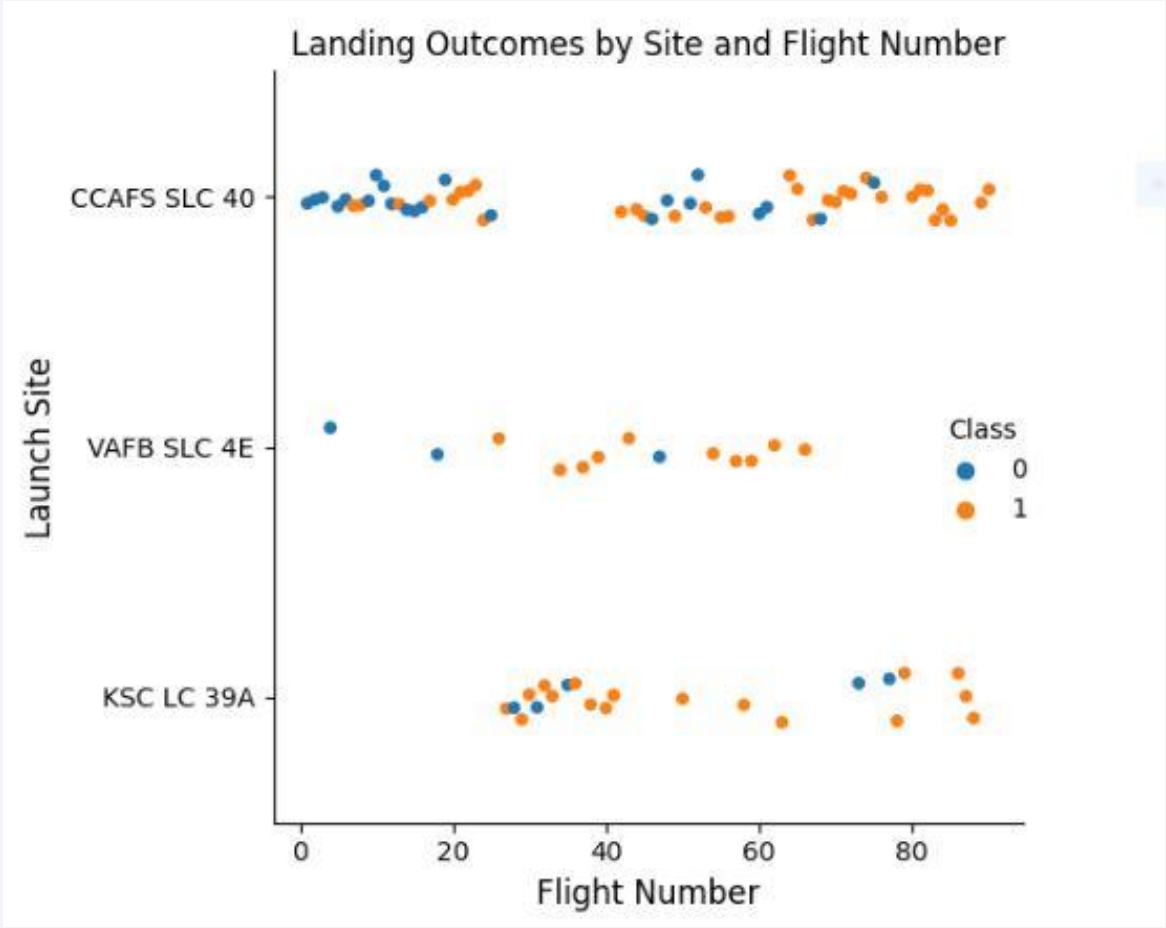
- Exploratory data analysis results and screenshots
- Interactive analytics demo in screenshots
- Predictive analysis results and screenshots

Section 2

Insights drawn from EDA

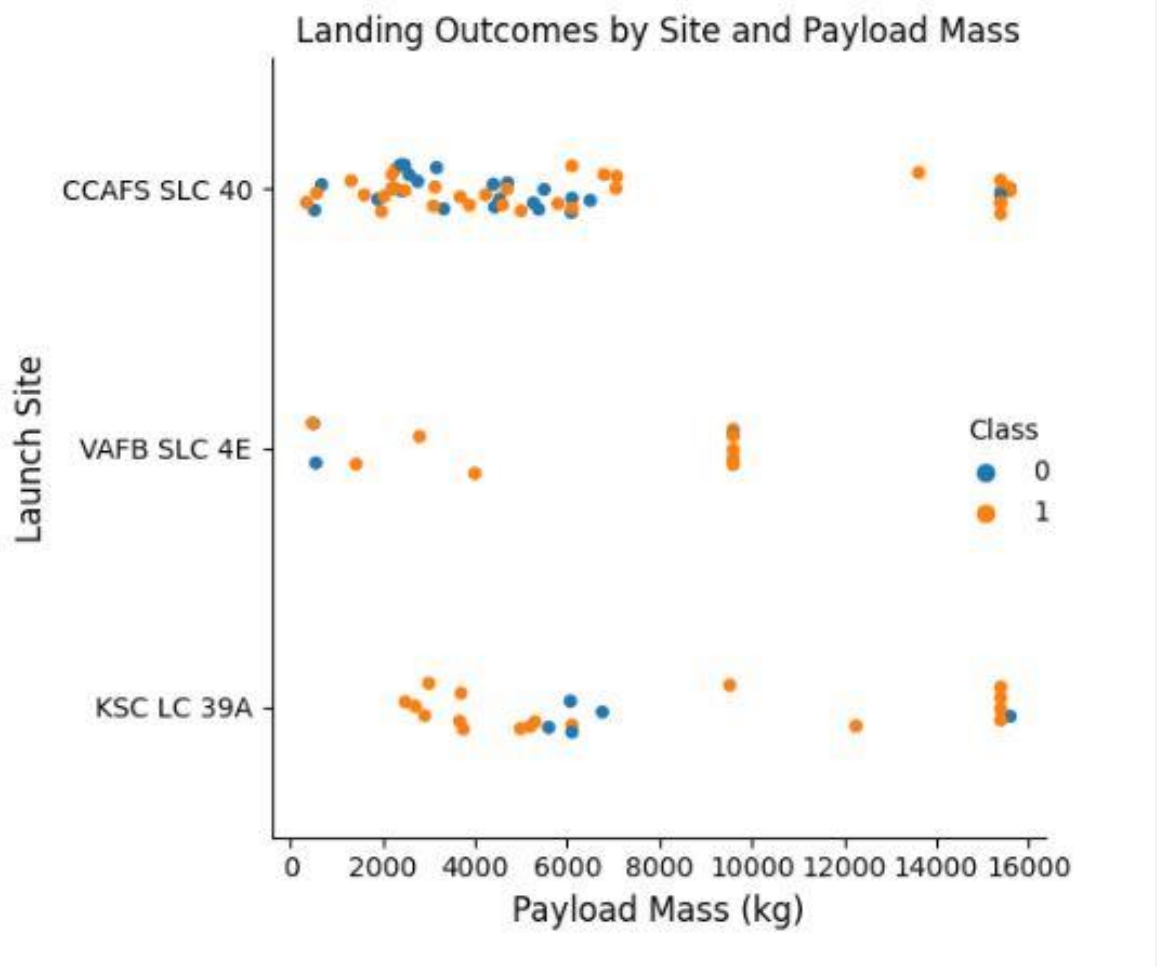
Flight Number vs. Launch Site

- Given that the flight numbers are ordered by date (lower number = earlier date), the success rate has improved over time
- Most of the unsuccessful landings occur at CCAFS SLC 40
- Most of the successful landings occur at KSC LC 39A or VAFB SLC 4E



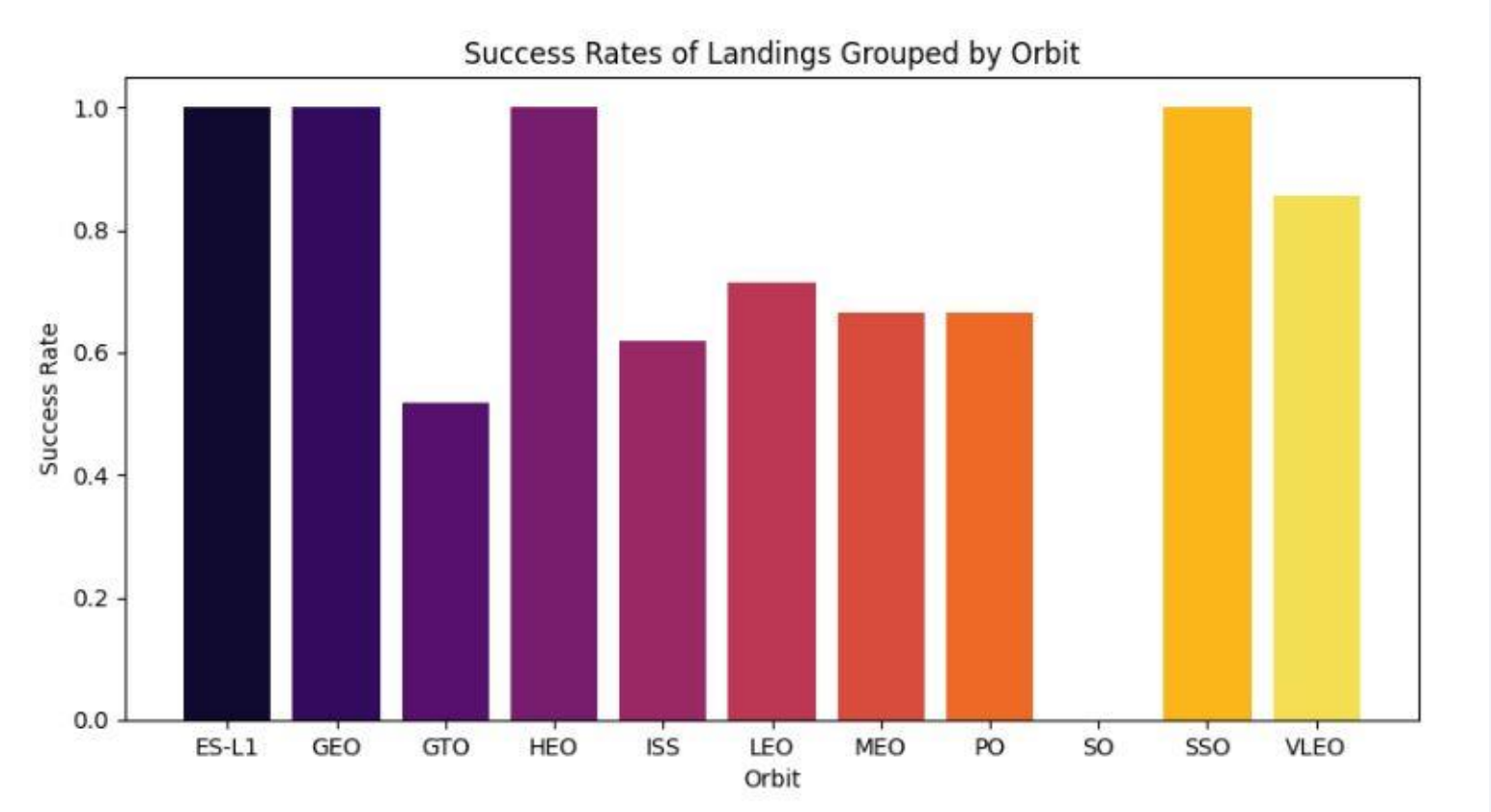
Payload Mass vs. Launch Site

- Heavier payload masses (8000kg+) tend to be successful landings, but more rockets carried lighter payloads, meaning less data available for those heavier payloads
- KSC LC 39A is the most successful in the lower payload mass range (0-5000kg)
- Payload masses between 8000kg and 14000kg have all been successful



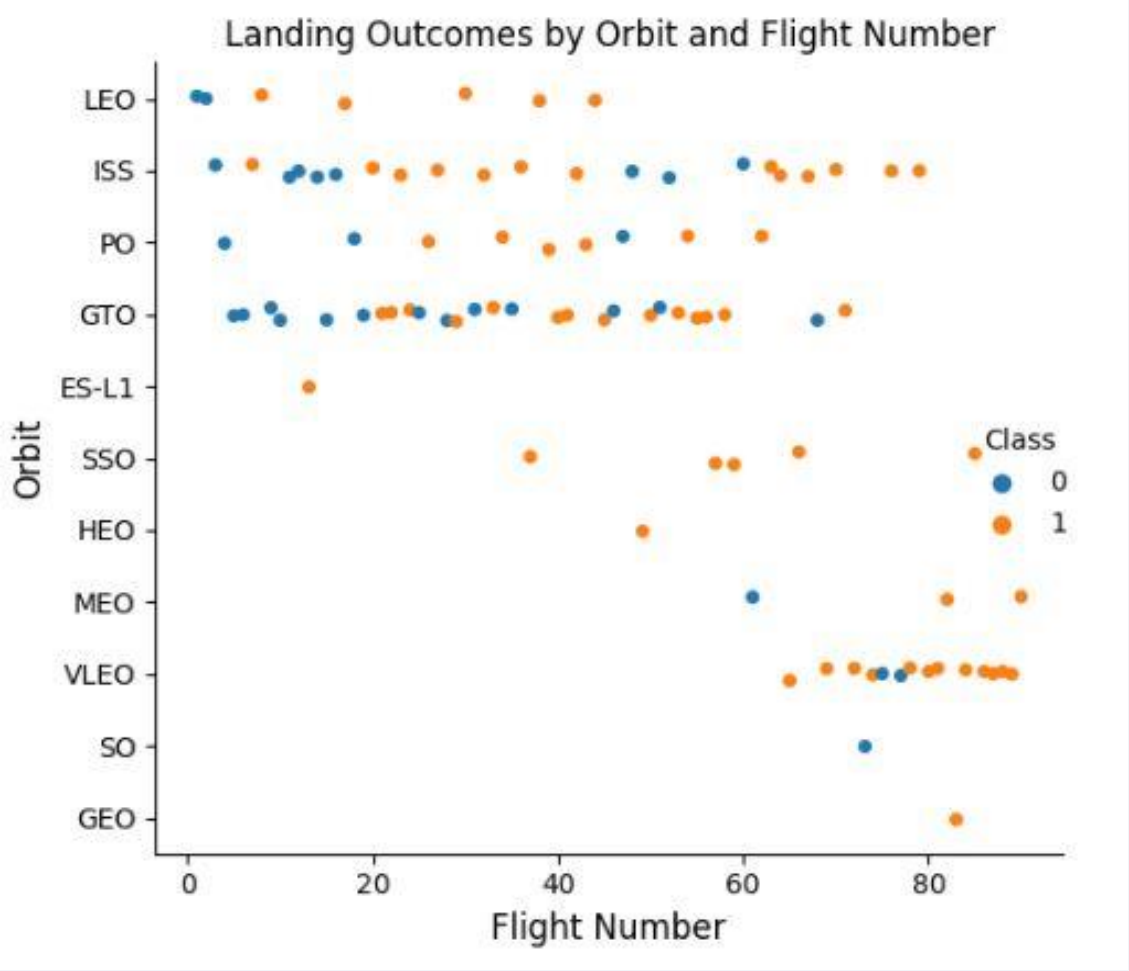
Success Rate vs. Orbit Type

- ES-L1, GEO, HEO, and SSO are the most successful orbits
- VLEO is still very successful
- Launches to GTO, the ISS, or SO have proven very risky for landing (60% success rate or lower)



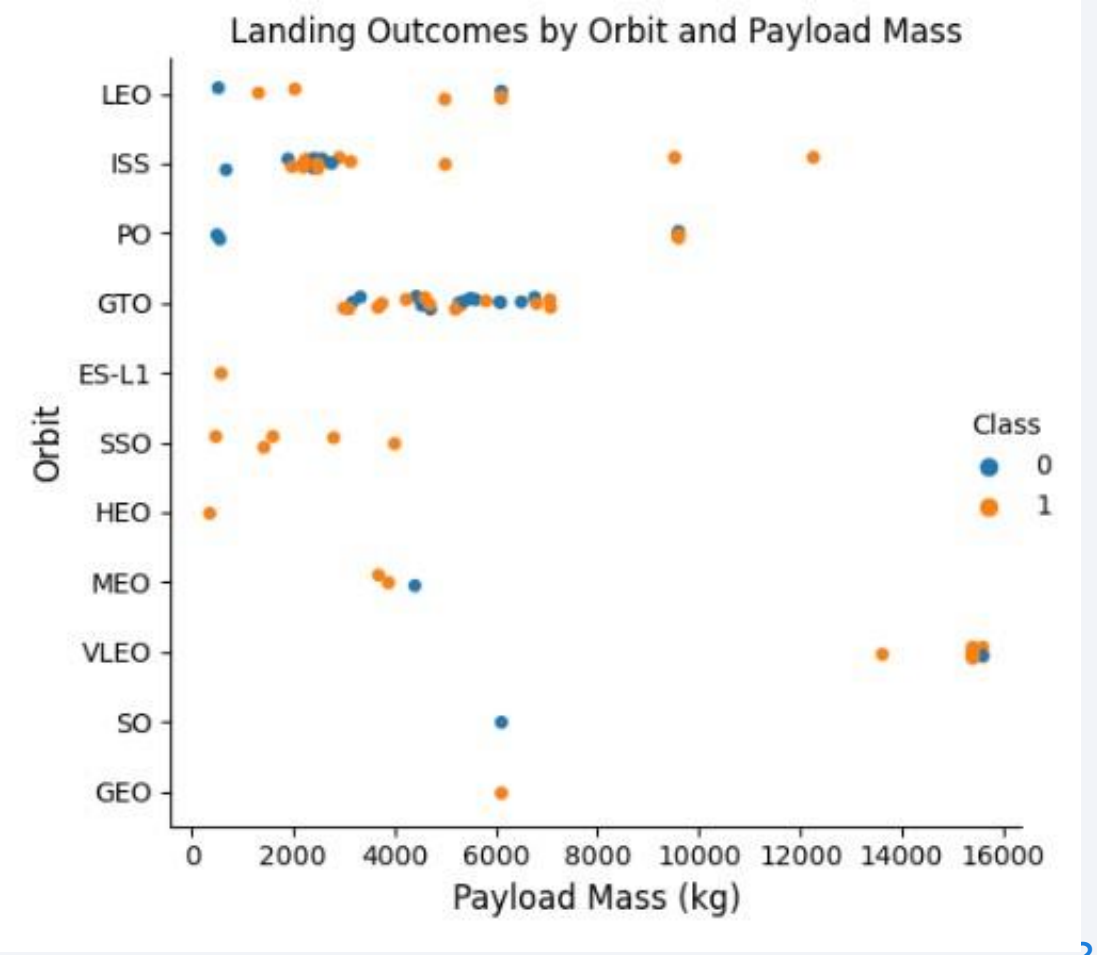
Flight Number vs. Orbit Type

- Launches to orbits including the ISS, PO, and GTO can be confidently deemed 'risky landing' because of the larger amount of attempts
- LEO, SSO and VLEO can confidently be deemed 'likely to succeed' due to the large number of successes
- ES-L1, HEO, MEO, SO, and GEO have minimal launch/land attempts. Their success rates may not be as accurate as other orbits



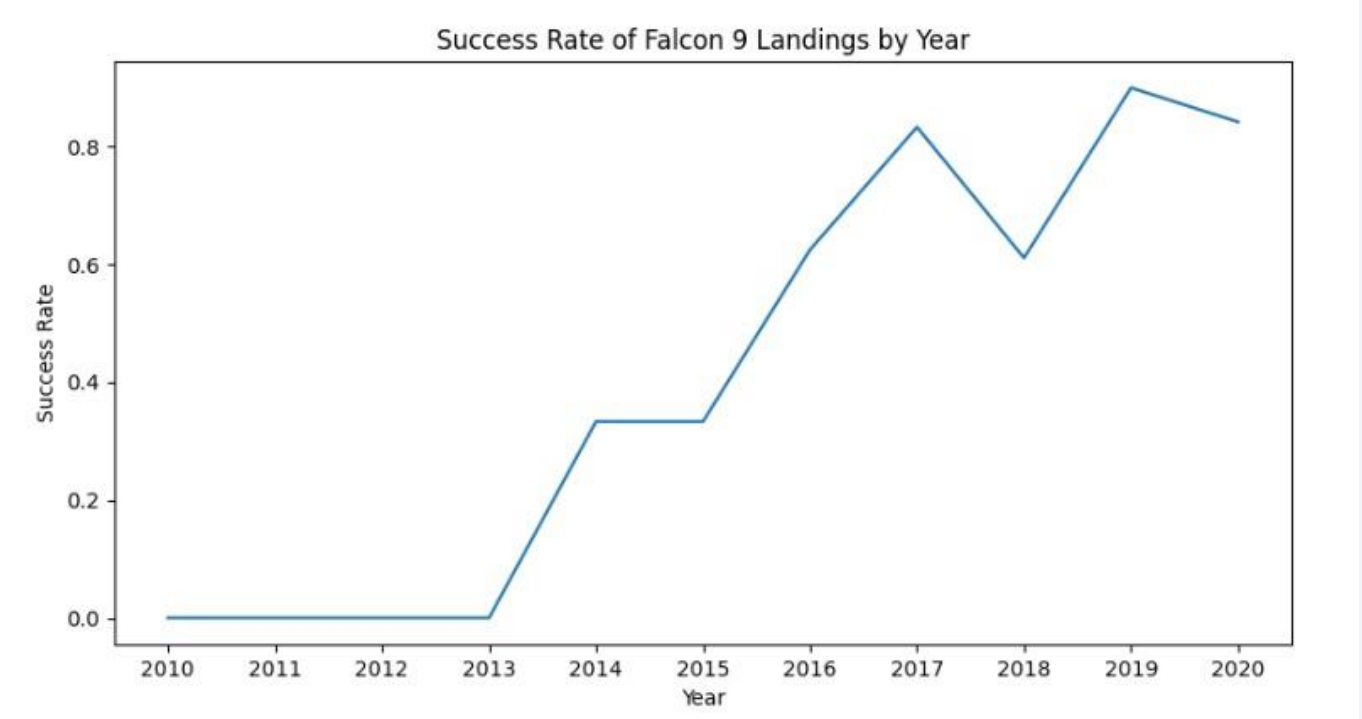
Payload Mass vs. Orbit Type

- This chart is telling of approximately how far each orbit is from Earth's surface. For instance, VLEO (very low Earth orbit) is only about 450km out, so it's easier to move more massive payloads the shorter distance
- Meanwhile, GEO is about 36,000km out, so a lighter payload is used
- If there is a very massive payload going to be shipped very far out (or a payload mass outside of SpaceX's usual mass for that orbit), the chance of failure may increase



Landing Success Yearly Trend

- From 2010 to 2020, the success rate of rocket launches/landings has (basically) only gone up
- There was a small dip in success in 2018, but this does not detract from the overall success rate since there is still a net positive between 2017 and 2019



All Launch Site Names

- The query returned all unique launch site names
- The NOT LIKE '%None%' was added in order to eliminate a 'None' that would appear when querying for the launch sites

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

```
%sql SELECT DISTINCT Launch_site FROM SPACEXTBL \
WHERE Launch_site \
NOT LIKE '%None%'
```

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

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

Launch Site Names Begin with 'CCA'

- The query returns the first 5 records containing a launch site that begins with CCA
- This includes the launch sites CCAFS LC-40 as well as CCAFS SLC-40 (not pictured in the first 5 entries)

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

```
%sql SELECT * FROM SPACEXTBL WHERE Launch_site \
LIKE 'CCA%' \
LIMIT 5
```

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

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing_Outc
06/04/2010	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0.0	LEO	SpaceX	Success	Failure (paracl
12/08/2010	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of Brouere cheese	0.0	LEO (ISS)	NASA (COTS) NRO	Success	Failure (paracl
22/05/2012	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525.0	LEO (ISS)	NASA (COTS)	Success	No att
10/08/2012	0:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500.0	LEO (ISS)	NASA (CRS)	Success	No att
03/01/2013	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677.0	LEO (ISS)	NASA (CRS)	Success	No att

Total Payload Mass

- The query adds all payload masses carried by boosters launched by NASA
- The customer was limited to only NASA and a sum of all values was used to obtain this result

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

```
%sql SELECT SUM(PAYLOAD_MASS__KG_) AS 'Total Payload Mass (kg) from NASA Boosters' FROM SPACEXTBL \
WHERE Customer = 'NASA (CRS)'
```

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

Total Payload Mass (kg) from NASA Boosters
45596.0

Average Payload Mass by F9 v1.1

- The query returns to average payload mass carried by the F9 v1.1 booster
- This was obtained by limiting the booster version to F9 v1.1 and averaging those payload mass values
- This data will be pertinent should the F9 v1.1 booster be used for more massive payloads in the future, since the average 2928.4kg is relatively light

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

```
%sql SELECT AVG(PAYLOAD_MASS__KG_) AS 'Average F9 Payload Mass (kg)' FROM SPACEXTBL \
WHERE Booster_Version = 'F9 v1.1'
```

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

Average F9 Payload Mass (kg)
2928.4

First Successful Ground Landing Date

- The query was used to find the first date during which a ground landing was successful
- Due to the dates not being in a 'year-first' format, the dates were all queried then ordered by year manually. The first listed date 'December 22, 2015', is the first successful ground pad landing date

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

Hint:Use min function

```
%sql SELECT Date AS 'First Success Date' FROM SPACEXTBL \
WHERE "Landing_Outcome" \
LIKE 'Success (ground pad)' \
ORDER BY substr(Date, 7, 4) \
LIMIT 1
```

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

First Success Date
22/12/2015

Successful Drone Ship Landing with Payload between 4000kg and 6000kg

- The query gathered the booster versions that were able to land safely via drone ship while carrying a payload mass between 4000kg and 6000kg by limiting the date to Successful drone ship landings and payload masses to 4000kg-6000kg
- This information can be a useful predictor of success should SpaceX decide to launch another payload between 4000kg and 6000kg

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

```
%sql SELECT DISTINCT Booster_Version FROM SPACEXTBL \
WHERE "Landing_Outcome" = 'Success (drone ship)' \
AND PAYLOAD_MASS__KG_ > 4000 \
AND PAYLOAD_MASS__KG_ < 6000
```

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

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

Total Number of Successful and Failure Mission Outcomes

- The query is two separate queries
  - The first query totals all successful mission outcomes
  - The second query outputs all failed mission outcomes
  - The 'UNION' line combines both queries into one data frame
- SpaceX has over a 99% mission success rate if the rocket gets off the ground

List the total number of successful and failure mission outcomes

```
%sql SELECT Mission_Outcome AS 'Mission Outcome', COUNT(Mission_Outcome) AS 'Occurrences' FROM SPACEXTBL \
WHERE Mission_Outcome \
LIKE '%Success%' \
UNION \
SELECT Mission_Outcome AS 'Mission Outcome', COUNT(Mission_Outcome) AS 'Occurrences' FROM SPACEXTBL \
WHERE Mission_Outcome \
LIKE '%Failure%'
```

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

Mission Outcome	Occurrences
Failure (in flight)	1
Success	100

Boosters Carried Maximum Payload

- The query outputs all booster versions that have carried the max payload weight by filtering the data by maximum payload weight then listing all distinct booster versions
- This data will be useful should SpaceX decide to launch very massive payloads in the future

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

```
%sql SELECT DISTINCT Booster_Version AS 'Booster Version that Held Max Payload Mass' FROM SPACEXTBL \
WHERE PAYLOAD_MASS__KG_ =(SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTBL)
```

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

Booster Version that Held Max 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

2015 Launch Records

- The query outputs records for failed drone ship landings that occurred during 2015. The months are listed instead of the date
- It should be noted that both failures occurred while at the CCAFS LC-40 launch site, which may be pertinent info in the future

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.

Note: SQLLite does not support monthnames. So you need to use substr(Date, 4, 2) as month to get the months and substr(Date,7,4)='2015' for year.

```
%sql SELECT substr(Date,4,2) AS 'Month', "Landing_Outcome", Booster_Version, Launch_Site FROM SPACEXTBL \
WHERE (substr(Date,7,4)='2015' AND "Landing_Outcome" = 'Failure (drone ship)')
```

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

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

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

- The query outputs the number and type of successful landings between the dates of 06/04/2010 and 03/20/2017 and ranks them by total value by ordering by count of each distinct type of landing outcome
- Between the aforementioned dates, there was not much difference between drone ship and ground pad landings.

Rank the count of successful landing_outcomes between the date 04-06-2010 and 20-03-2017 in descending order.

```
%sql SELECT "Landing_Outcome", COUNT(Landing_Outcome) from SPACEXTBL \
WHERE (Date BETWEEN '04-06-2010' AND '20-03-2017') AND Landing_Outcome LIKE '%Success%' \
GROUP BY Landing_Outcome \
ORDER BY COUNT(Landing_Outcome) \
DESC
```

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

Landing_Outcome	COUNT(Landing_Outcome)
Success	20
Success (drone ship)	8
Success (ground pad)	7

Section 3

Launch Sites Proximities Analysis

SpaceX Launch Sites on the Global Map

- SpaceX's launch sites can be seen on this global map as orange dots. There are 3 sites in Florida (USA) and one site in California (USA). All launch sites are on coastlines

Launch Site Map and Success

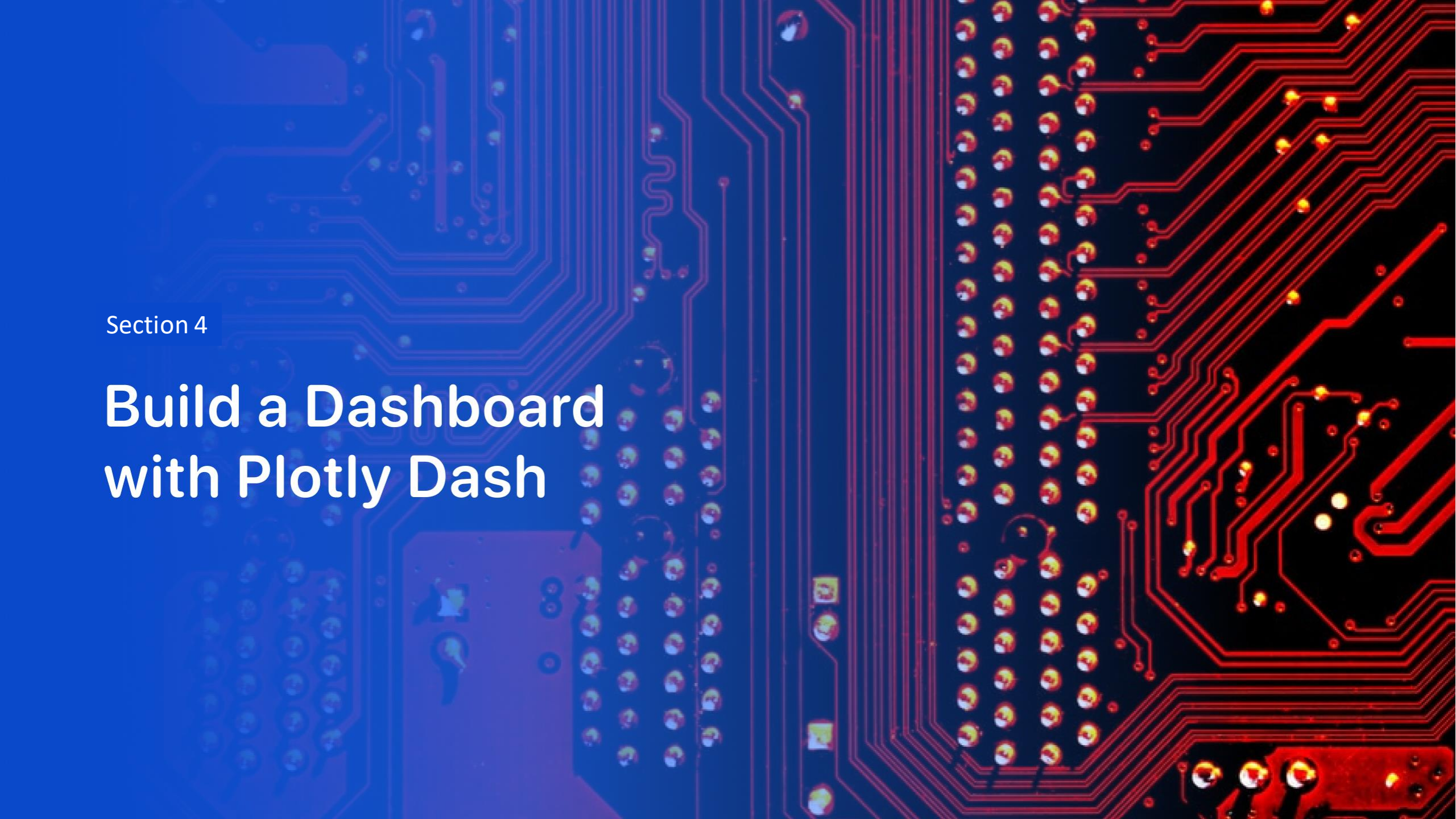
- The California-based VAFB SLC-4E is pictured with markers denoting the SpaceX successes (green markers) and failures (red markers) that took place
- The markers begin collapsed onto one marker that can be clicked on and exploded into the various markers shown

Launch Site Distance to Civilization

- From the launch site, the closest city limit, Lompoc, CA (marked at the bottom left), is 11.61km away
- The closest highway, the convergence of CA-1 and CA-246 (marked at the top right) is 14.01km away
- Civilization is far sufficiently far away from launch sites should a launch go wrong

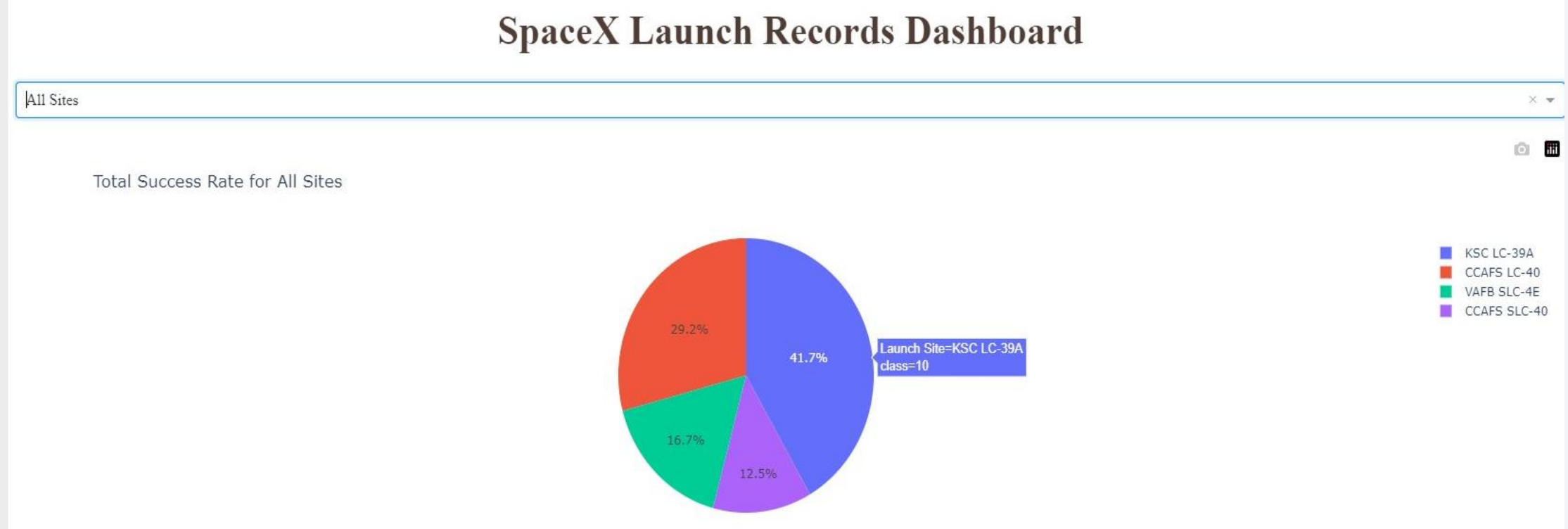
Launch Site Distance to Coast and Rail

- The launch site is 1.35km from the coastline of the Pacific Ocean. The site is 1.27km from the Santa Barbara Subdivision of the Southern Pacific rail line
- The site is close to water. Should a launch go wrong, the rocket can potentially be steered towards the ocean
- The site is close to a railroad. Although civilians do ride this line and launch sites aren't typically near civilian transit, being close to a railroad in order to have heavy rocket parts transported is beneficial


Section 4
Build a Dashboard with Plotly Dash

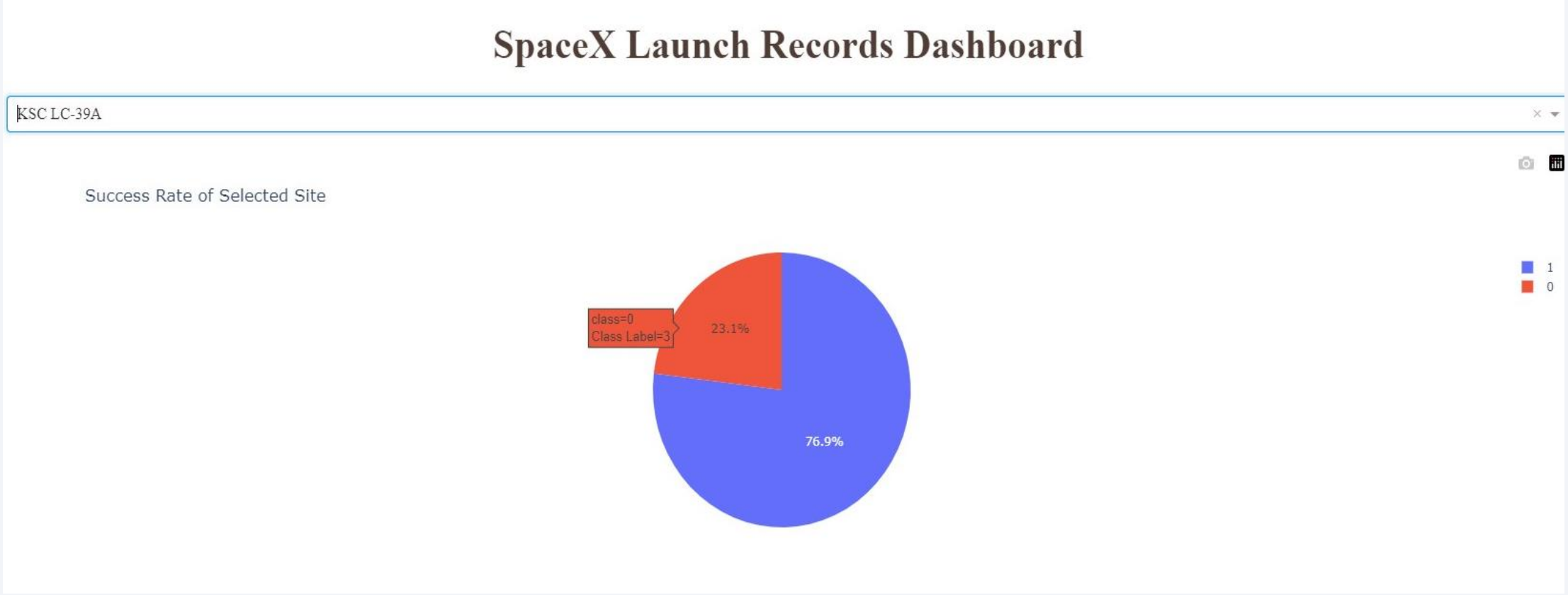
Comparing Launch Site Success

- The dashboard displays that the KSC LC-39A makes up 41.7% of all launch site success
- The next most successful site is the CCAFS LC-40 with 29.2%
- This will be important for determining success when SpaceX denotes which launch site will be used for a future mission



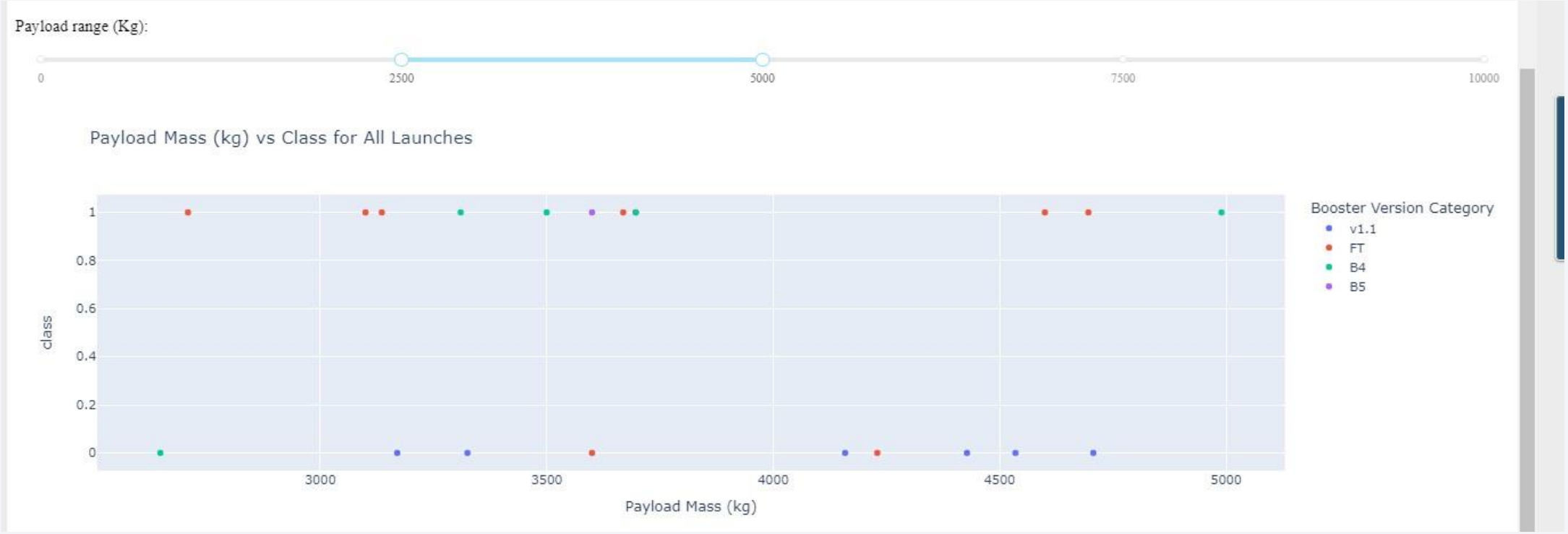
Success Rate at KSC LC-39A

- The launch site with the highest success rate is the KSC LC-39A with a 76.9% success rate.
- This could easily be determined visualize thanks to the Plotly pie chart and dropdown selection bar



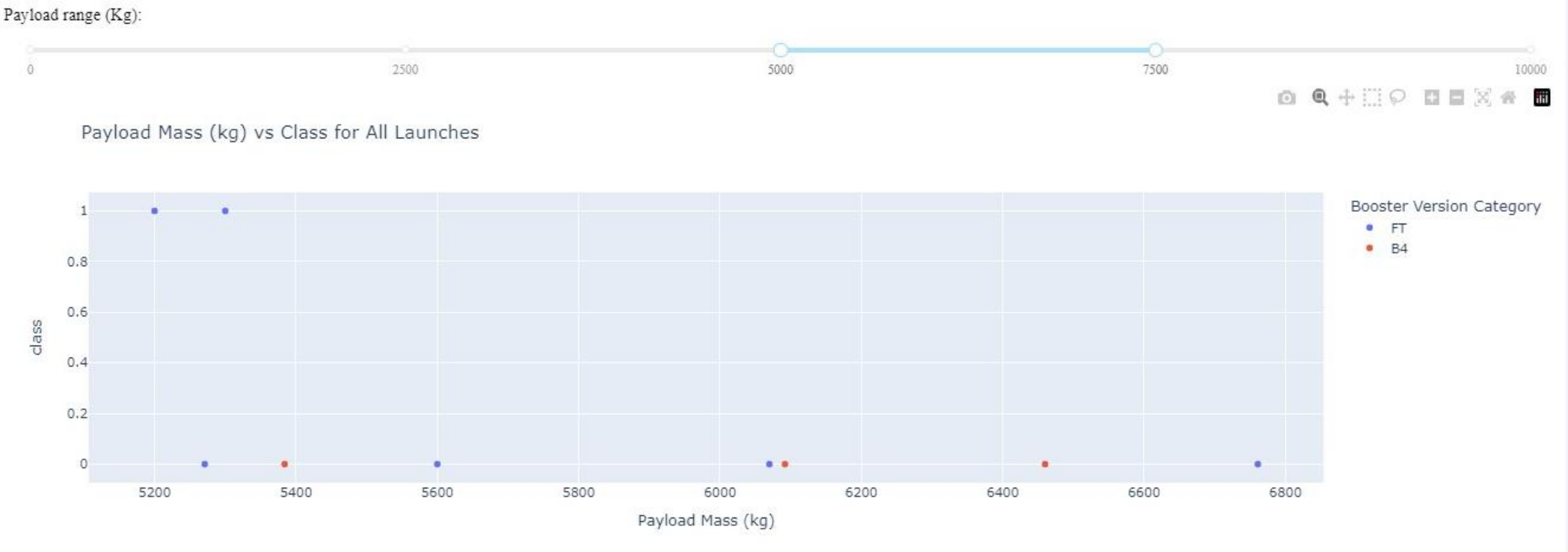
Scatter Plot of Success Rate for Varying Payloads

- Using the payload slider, it can be seen that there is a varying level of success for payload masses between 2500kg and 5000kg.
- The FT booster version performs the best at this weight range while the B5 performs the worst



Scatter Plot of Success Rate for Varying Payloads

- Adjusting the payload slider to a payload mass between 5000kg and 7500kg shows an overwhelmingly high failure rate when compared to lighter payload masses
- The FT booster is pretty unreliable at this weight range. The B4 had yet to be successful at the given weight range

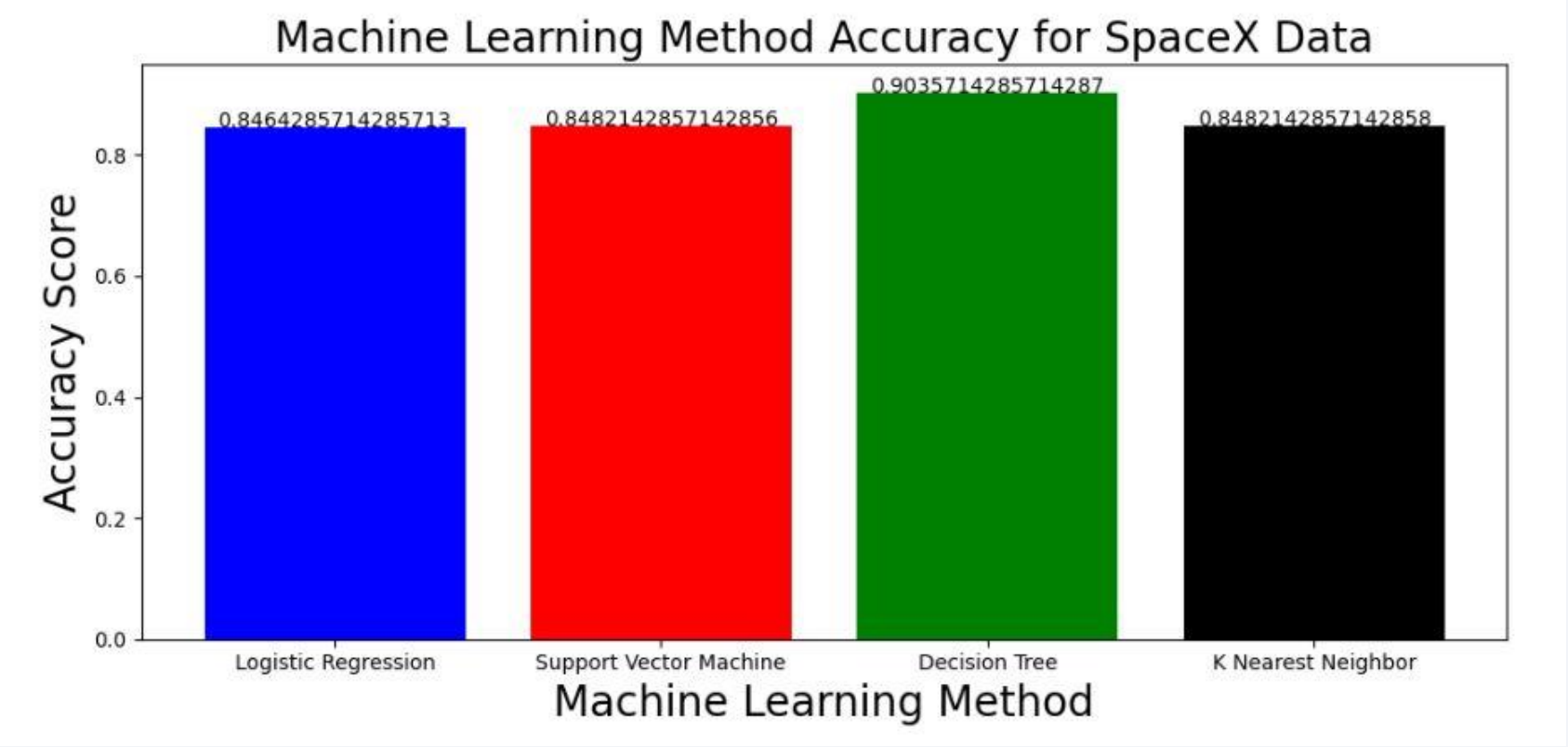


Section 5

Predictive Analysis (Classification)

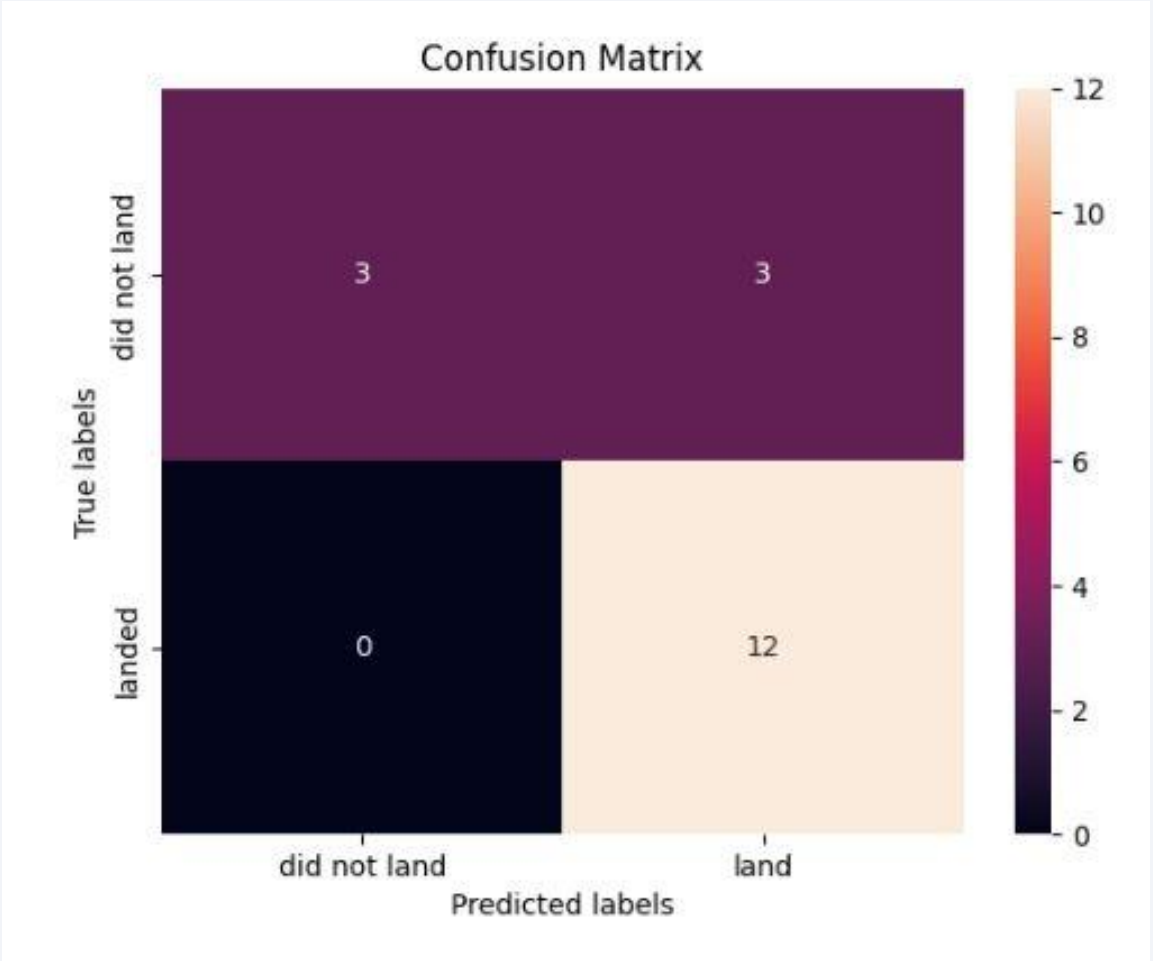
Classification Accuracy

- The model with the highest accuracy is the decision tree method with over a 90% potential accuracy
- Despite the other methods hovering around 84% maximum accuracy, the decision tree is still significantly more accurate



Confusion Matrix of Decision Tree Method

- The confusion matrix for the decision tree method shows that it did not create any Type 2 error (false negative), denoted by the black '0' box on the bottom left
- All error was Type 1 error (false positive), denoted by the purple '3' box at the top right (predicted to land but did not land)
- The model was mostly accurate, denoted by the top left and bottom right boxes



Conclusions

- The highest payload mass success rate was at heaviest weights (10,000kg+), but most flights have lower mass payloads (7000kg or less)
- The orbits with highest success rates were closer to Earth, such as LEO, VLEO, and SSO. Farther orbits, such as GTO and ISS, were much less successful. Covering more distance means there is more of a chance something breaks or malfunctions during travel
- The KSC LC-39A site has seen the most launch/land success out of all SpaceX sites
- The yearly success rate trend line for SpaceX rockets launches/landings has a positive slope, predicting an even higher success rate in years to come
- SQL queries and the dashboard have helped determine which combinations of boosters, payload weights, and launch sites have seen mission/landing success and which have seen failure. These methods will be useful for predicting outcomes of future flights using not-yet seen combinations of boosters and payloads
- The decision tree model was the most accurate machine learning algorithm for the SpaceX data when properly tuned. Though, SpaceY should be weary of Type 1 error in the future

Appendix

- API Calls
  CSV: https://github.com/CSinatra26/CSinatra_IBM_Data_Science_Capstone_Final_Presentation/blob/main/Appendix_1_API_CSV.csv
- Webscraped
  CSV: https://github.com/CSinatra26/CSinatra_IBM_Data_Science_Capstone_Final_Presentation/blob/main/Appendix_2_Webscraped_CSV.csv
- Data Wrangling
  CSV: https://github.com/CSinatra26/CSinatra_IBM_Data_Science_Capstone_Final_Presentation/blob/main/Appendix_3_Data_Wrangling_CSV.csv
- Title Slide Image Source: https://www.istockphoto.com/photos/rocket
- Full Github Repository
  Link: https://github.com/CSinatra26/CSinatra_IBM_Data_Science_Capstone_Final_Presentation.git

Thank you!

Shoot for the stars!