IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Harvinder Saini

16 April 2025
GitHub: https://github.com/harvindersainiibm/Applied-Data-Science-Capstone

Outline

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

Executive Summary

- Summary of methodologies
- Summary of all results

Introduction

- Project background and context
- Problems you want to find answers

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - SpaceX Rest API
  - Web Scraping from Wikipedia
- Perform data wrangling
  - Data cleaning (e.g. replacing missing payload mass with its mean; removing irrelevant columns)
  - Create a landing outcome label
  - Data transformation (One Hot Encoding for categorical features, Standardization of numerical features)
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - Linear Regression, K Nearest Neighbors, Support Vector Machine, and Decision Tree models have been built and evaluated for the best classifier

Data Collection

SpaceX REST API	Web Scraping from Wikipedia
• SpaceX launches is gathered from the SpaceX REST API. • The url starts with api.spacexdata.com/v4/ • The information gathered include: the rocket used, payload delivered, launch specifications, landing specifications, and landing outcome.	• SpaceX lunches is gathered from Wikipedia: • https://en.wikipedia.org/wiki/List_of_Falcon_9_and_Falcon_Heavy_launches • The launch records are stored in a HTML table

Data Collection – SpaceX API

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

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

Check the content of the response

```
[11]: print(response.content)
```


```
b'[{"fairings":{"reused":false,"recovery_attempt":false,"recovered":false,"ships":[]},"links":{"patch":{"small":"https://images2.imgbox.com/94/f2/NN6Ph45
r_o.png","large":"https://images2.imgbox.com/5b/02/QcxHUb5V_o.png"},"reddit":{"campaign":null,"launch":null,"media":null,"recovery":null},"flickr":{"smal
l":[],"original":[]},"presskit":null,"webcast":"https://www.youtube.com/watch?v=0a_00nJ_Y88","youtube_id":"0a_00nJ_Y88","article":"https://www.space.com/
2196-spacex-inaugural-falcon-1-rocket-lost-launch.html","wikipedia":"https://en.wikipedia.org/wiki/DemoSat"},"static_fire_date_utc":"2006-03-17T00:00:00.
000Z","static_fire_date_unix":1142553600,"net":false,"window":0,"rocket":"5e9d0d95eda69955f709d1eb","success":false,"failures":[{"time":33,"altitude":nul
l,"reason":"merlin engine failure"}],"details":"Engine failure at 33 seconds and loss of vehicle","crew":[],"ships":[],"capsules":[],"payloads":["5eb0e4b
5b6c3bb0006eeb1e1"],"launchpad":"5e9e4502f5090995de566f86","flight_number":1,"name":"FalconSat","date_utc":"2006-03-24T22:30:00.000Z","date_unix":1143239
400,"date_local":"2006-03-25T10:30:00+12:00","date_precision":"hour","upcoming":false,"cores":[{"core":"5e9e289df35918033d3b2623","flight":1,"gridfins":f
alse,"legs":false,"reused":false,"landing_attempt":false,"landing_success":null,"landing_type":null,"landpad":null}],"auto_update":true,"tbd":false,"laun
ch_library_id":null,"id":"5eb87cd9ffd86e000604b32a"},{"fairings":{"reused":false,"recovery_attempt":false,"recovered":false,"ships":[]},"links":{"patch":
{"small":"https://images2.imgbox.com/f9/4a/ZboXReNb_o.png","large":"https://images2.imgbox.com/80/a2/bkWotCIS_o.png"},"reddit":{"campaign":null,"launch":
null,"media":null,"recovery":null},"flickr":{"small":[],"original":[]},"presskit":null,"webcast":"https://www.youtube.com/watch?v=Lk4zQ2wP-Nc","youtube_i
d":"Lk4zQ2wP-Nc","article":"https://www.space.com/3590-spacex-falcon-1-rocket-fails-reach-orbit.html","wikipedia":"https://en.wikipedia.org/wiki/DemoSa
t"},"static_fire_date_utc":null,"static_fire_date_unix":null,"net":false,"window":0,"rocket":"5e9d0d95eda69955f709d1eb","success":false,"failures":[{"tim
e":301,"altitude":289,"reason":"harmonic oscillation leading to premature engine shutdown"}],"details":"Successful first stage burn and transition to sec
ond stage, maximum altitude 289 km, Premature engine shutdown at T+7 min 30 s, Failed to reach orbit, Failed to recover first stage","crew":[],"ships":
[],"capsules":[],"payloads":["5eb0e4b6b6c3bb0006eeb1e2"],"launchpad":"5e9e4502f5090995de566f86","flight_number":2,"name":"DemoSat","date_utc":"2007-03-21
T01:10:00.000Z","date_unix":1174439400,"date_local":"2007-03-21T13:10:00+12:00","date_precision":"hour","upcoming":false,"cores":[{"core":"5e9e289ef35918
```

Data Collection – SpaceX API

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:

[12]:
```
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

[13]:
```
response.status_code
```

[13]:
```
200
```

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

[14]:
```
# Use json_normalize meethod to convert the json result into a dataframe
response.json()
```

[14]:

```
[{'fairings': {'reused': False,
   'recovery_attempt': False,
   'recovered': False,
   'ships': []},
  'links': {'patch': {'small': 'https://images2.imgbox.com/94/f2/NN6Ph45r_o.png',
    'large': 'https://images2.imgbox.com/5b/02/QcxHUb5V_o.png'},
```

Data Collection – SpaceX API

[16]:
```python
json_data=response.json()
data = pd.json_normalize(json_data)
```

Using the dataframe `data` print the first 5 rows

[17]:
```python
# Get the head of the dataframe
data.head(5)
```

[17]:

	static_fire_date_utc	static_fire_date_unix	net	window	rocket	success	failures	details	crew	ships	capsules	payloa
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	[]	[]	[]	[5eb0e4b5b6c3bb0006eeb1
1	None	NaN	False	0.0	5e9d0d95eda69955f709d1eb	False	[{'time': 301, 'altitude': 289, 'reason': 'harmonic oscillation leading to premature engine shutdown'}]	Successful first stage burn and transition to second stage, maximum altitude 289 km, Premature engine shutdown at T+7 min 30 s, Failed to reach	[]	[]	[]	[5eb0e4b6b6c3bb0006eeb1

Data Collection - Scraping

Create Beautiful Soap object

```
[64]: # assign the response to a object
      page = requests.get(static_url)
      page

[64]: <Response [200]>

[74]: # Calculate the mean value of PayloadMass column

      html_tables = beauti_soup.find_all('table')

[75]: # BeautifulSoup() to create a BeautifulSoup object from a response text content
      content = page.text
      BeautifulSoup = BeautifulSoup(content, 'html.parser')

[76]: # Use soup.title attribute
      page_title = BeautifulSoup.title
      print(page_title)
      html_tables = BeautifulSoup.find_all('table')
      first_launch_table = html_tables[2]
      print(first_launch_table)

      <title>List of Falcon 9 and Falcon Heavy launches - Wikipedia</title>
      <table class="wikitable plainrowheaders collapsible" style="width: 100%;">
      <tbody><tr>
      <th scope="col">Flight No.
      </th>
      <th scope="col">Date and<br/>time (<a href="/wiki/Coordinated_Universal_Time" title="Coordinated Universal Time">UTC</a>)
      </th>
      <th scope="col"><a href="/wiki/List_of_Falcon_9_first-stage_boosters" title="List of Falcon 9 first-stage boosters">Version,<br/>Booster</a> <sup class
      ="reference" id="cite_ref-booster_11-0"><a href="#cite_note-booster-11"><span class="cite-bracket">[</span>b<span class="cite-bracket">]</span></a></sup
      >
      </th>
```

Data Collection - Scraping

```
[77]: column_names = []

      # Apply find_all() function with `th` element on first_launch_table
      # Iterate each th element and apply the provided extract_column_from_header() to get a column name
      # Append the Non-empty column name (`if name is not None and len(name) > 0`) into a list called column_names
      for i in first_launch_table.find_all('th'):
          if extract_column_from_header(i)!=None:
              if len(extract_column_from_header(i))>0:
                  column_names.append(extract_column_from_header(i))
```

```
[78]: print(column_names)
```

```
['Flight No.', 'Date and time ( )', 'Launch site', 'Payload', 'Payload mass', 'Orbit', 'Customer', 'Launch outcome']
```

```
[79]: launch_dict= dict.fromkeys(column_names)

      # Remove an irrelvant column
      del launch_dict['Date and time ( )']

      # Let's initial the launch_dict with each value to be an empty list
      launch_dict['Flight No.'] = []
      launch_dict['Launch site'] = []
      launch_dict['Payload'] = []
      launch_dict['Payload mass'] = []
      launch_dict['Orbit'] = []
      launch_dict['Customer'] = []
      launch_dict['Launch outcome'] = []
      # Added some new columns
      launch_dict['Version Booster']=[]
      launch_dict['Booster landing']=[]
      launch_dict['Date']=[]
      launch_dict['Time']=[]
```

Data Collection - Scraping

Dragon

```
[98]: launch_dict= dict.fromkeys(column_names)

# Remove an irrelvant column
del launch_dict['Date and time ( )']

# Let's initial the launch_dict with each value to be an empty list
launch_dict['Flight No.'] = []
launch_dict['Launch site'] = []
launch_dict['Payload'] = []
launch_dict['Payload mass'] = []
launch_dict['Orbit'] = []
launch_dict['Customer'] = []
launch_dict['Launch outcome'] = []
# Added some new columns
launch_dict['Version Booster']=[]
launch_dict['Booster landing']=[]
launch_dict['Date']=[]
launch_dict['Time']=[]
```

```
[99]: #append data
date =datatimelist[0].strip(',')
launch_dict['Date'].append(date)
```

```
[100]: df=pd.DataFrame(launch_dict)
```

Data Wrangling

We will import the following libraries.

[3]:
```
# Pandas is a software library written for the Python programming language for data manipulation and analysis.
import pandas as pd
#NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection c
import numpy as np
```

Data Analysis

Load Space X dataset, from last section.

[4]:
```
df=pd.read_csv("https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/dataset_part_1.csv")
df.head(10)
```

[4]:

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block	ReusedCount	Serial	Longit
0	1	2010-06-04	Falcon 9	6104.959412	LEO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B0003	-80.577
1	2	2012-05-22	Falcon 9	525.000000	LEO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B0005	-80.577
2	3	2013-03-01	Falcon 9	677.000000	ISS	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B0007	-80.577
3	4	2013-09-29	Falcon 9	500.000000	PO	VAFB SLC 4E	False Ocean	1	False	False	False	NaN	1.0	0	B1003	-120.610
4	5	2013-12-03	Falcon 9	3170.000000	GTO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B1004	-80.577

Data Wrangling

[5]:
```
df.isnull().sum()/len(df)*100
```

[5]:
```
FlightNumber       0.000000
Date               0.000000
BoosterVersion     0.000000
PayloadMass        0.000000
Orbit              0.000000
LaunchSite         0.000000
Outcome            0.000000
Flights            0.000000
GridFins           0.000000
Reused             0.000000
Legs               0.000000
LandingPad        28.888889
Block              0.000000
ReusedCount        0.000000
Serial             0.000000
Longitude          0.000000
Latitude           0.000000
dtype: float64
```

Identify which columns are numerical and categorical:

[6]:
```
df.dtypes
```

[6]:
```
FlightNumber       int64
Date              object
BoosterVersion    object
PayloadMass      float64
Orbit             object
LaunchSite        object
Outcome           object
Flights            int64
GridFins            bool
```

Use the method `value_counts()` on the column `LaunchSite` to determine the number of launches on each site:

[7]:
```
# Apply value_counts() on column LaunchSite
df['LaunchSite'].value_counts()
```

[7]:
```
LaunchSite
CCAFS SLC 40    55
KSC LC 39A      22
VAFB SLC 4E     13
Name: count, dtype: int64
```

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

- **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

Data Wrangling

Use the method `.value_counts()` to determine the number and occurrence of each orbit in the column `Orbit`

```
[8]: # Apply value_counts on Orbit column

     df['Orbit'].value_counts()
```

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

TASK 3: Calculate the number and occurence of mission outcome of the orbits

Use the method `.value_counts()` on the column `Outcome` to determine the number of `landing_outcomes` .Then

```
[9]: # landing_outcomes = values on Outcome column
     landing_outcomes= df['Outcome'].value_counts()
     landing_outcomes
```

```
[9]: Outcome
     True ASDS      41
     None None      19
     True RTLS      14
     False ASDS      6
```

Data Wrangling

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

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

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

```
[10]: {'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`; othe
variable `landing_class`:

```
[11]: # landing_class = 0 if bad_outcome
      # landing_class = 1 otherwise
      def outcome_to_class(outcome_value):
          if outcome_value in bad_outcomes:
              return 0
          else:
              return 1

      landing_class = df['Outcome'].apply(outcome_to_class)
      landing_class
```

```
[11]: 0     0
      1     0
      2     0
      3     0
      4     0
            ..
      85    1
      86    1
```

Data Wrangling

This variable will represent the classification variable that represents the outcome of each launch. If the valu
first stage landed Successfully

[12]:
```python
df['Class']=landing_class
df[['Class']].head(8)
```

[12]:

	Class
0	0
1	0
2	0
3	0
4	0
5	0
6	1
7	1

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

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

[13]:
```python
df["Class"].mean()
```

[13]:
```
np.float64(0.6666666666666666)
```

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

[14]:

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block	R
0	1	2010-06-04	Falcon 9	6104.959412	LEO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	
1	2	2012-05-22	Falcon 9	525.000000	LEO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	
2	3	2013-03-01	Falcon 9	677.000000	ISS	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	
3	4	2013-09-29	Falcon 9	500.000000	PO	VAFB SLC 4E	False Ocean	1	False	False	False	NaN	1.0	
4	5	2013-12-03	Falcon 9	3170.000000	GTO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	

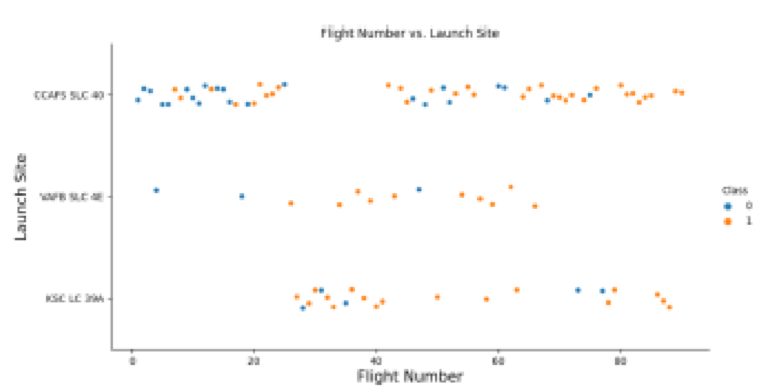
[15]:
```python
df.to_csv("dataset_part_2.csv", index=False)
```

[14]:

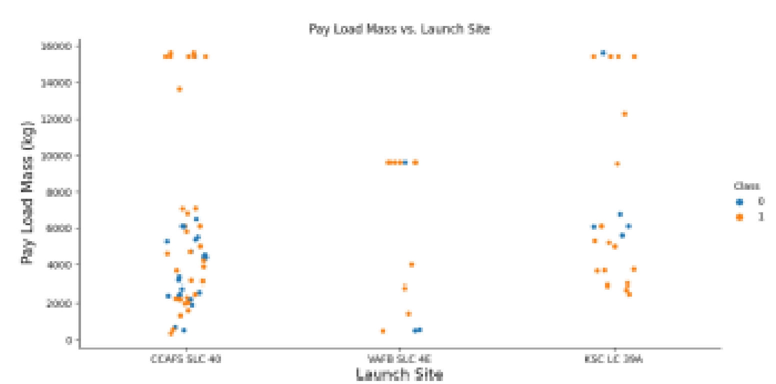
FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins

EDA with Data Visualization

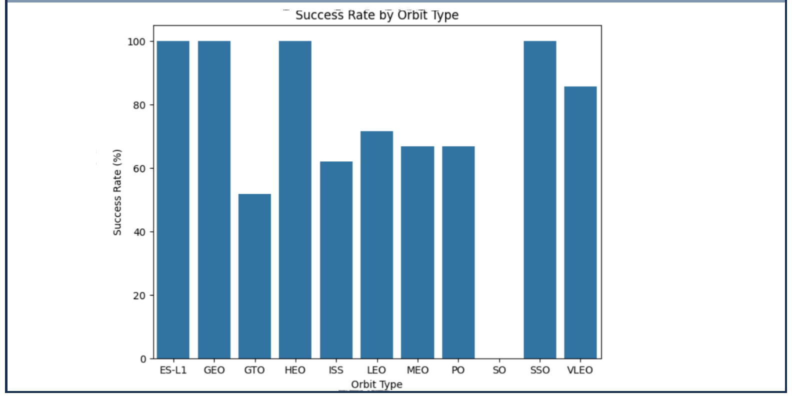
Flight Number and Launch Site



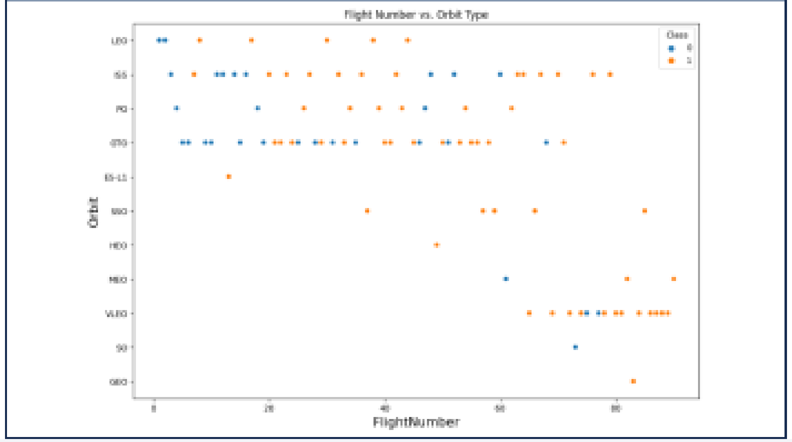
Payload Mass and Launch Site



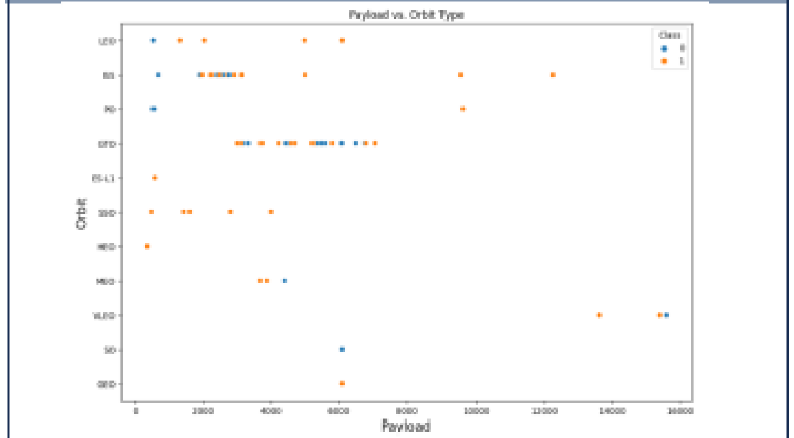
Success Rate and Orbit Type



Flight Number and Orbit Type



Payload Mass and Orbit Type



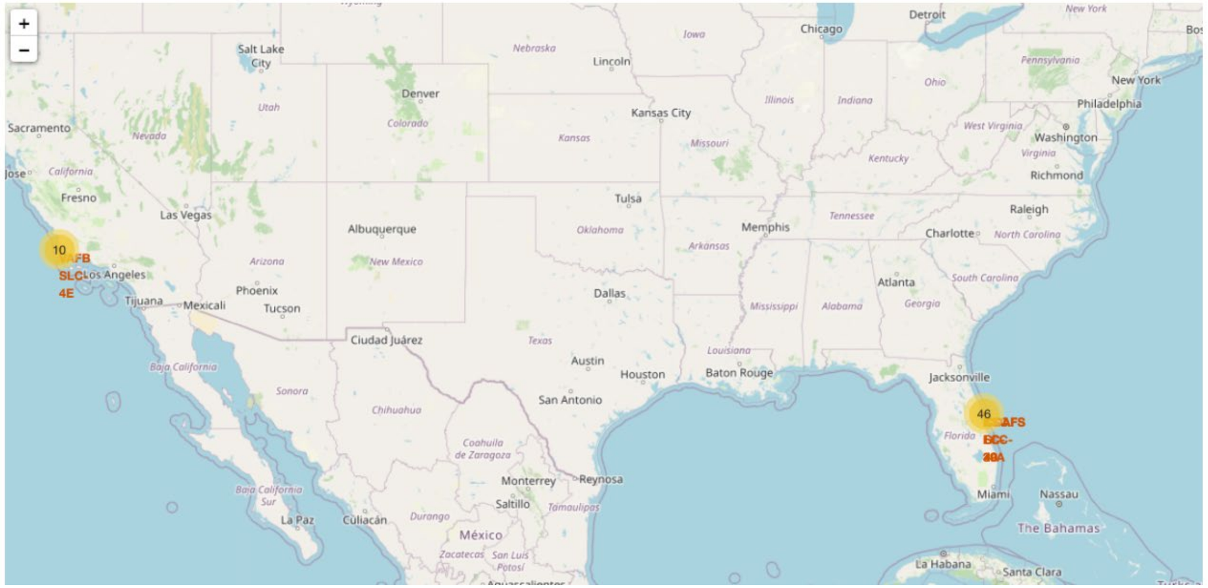
Trend of success over the years

EDA with SQL

- Displaying the names of the unique launch sites in the space mission
- Displaying 5 records where launch sites begin with the string 'KSC
- Displaying the total payload mass carried by boosters launched by NASA (CRS)
- Displaying average payload mass carried by booster version F9 v1.1
- Listing the date where the successful landing outcome in drone ship was achieved.
- Listing the names of the boosters which have success in ground pad and have payload mass greater than 4000 but less than 6000
- Listing the total number of successful and failure mission outcomes
- Listing the names of the booster versions which have carried the maximum payload mass
- Listing the records which will display the month names, successful landing outcomes in ground pad booster versions, launch site for the months in year 2017
- Ranking the count of successful landing outcomes between the date 2010 06 04 and 2017 03 20 in descending order

Build an Interactive Map with Folium

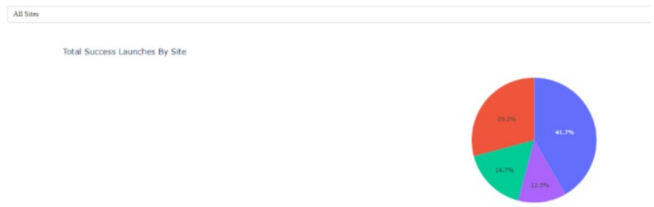
- An interactive map offers a dynamic way to visualize and analyze launch site data, facilitating decision-making and insights intothe spatial relationships of launch activities. For instance, the success rate may depend on the location and proximities of a launch site, i.e., the initial position of rocket trajectories.
- We created three maps for:
- Visualization of each launch site in an interactive map.
- Visualization of launch sites on the map based on fail or success. This gives us insights into the performance of launch sites.
- Visualization of the geographical context and proximityof the launch sites to railway, highway, coastline, etc



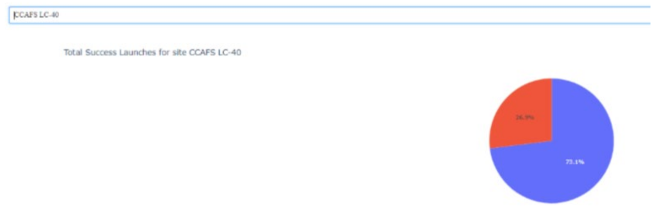
Build a Dashboard with Plotly Dash

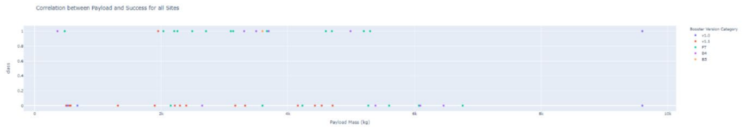
- PlotlyDash applications allows us to perform interactive visual analytics on SpaceX lunch data in real time.
- We developed a dashboard that enables us to:
- Visualize a pie chart depicting success based on the selected launch site.
- Explore the correlation between success and payload by utilizing a Range Slider to choose the payload range, presented through a scatter plot.

- Pie chart for all sites are selected



- Pie chart for is selected

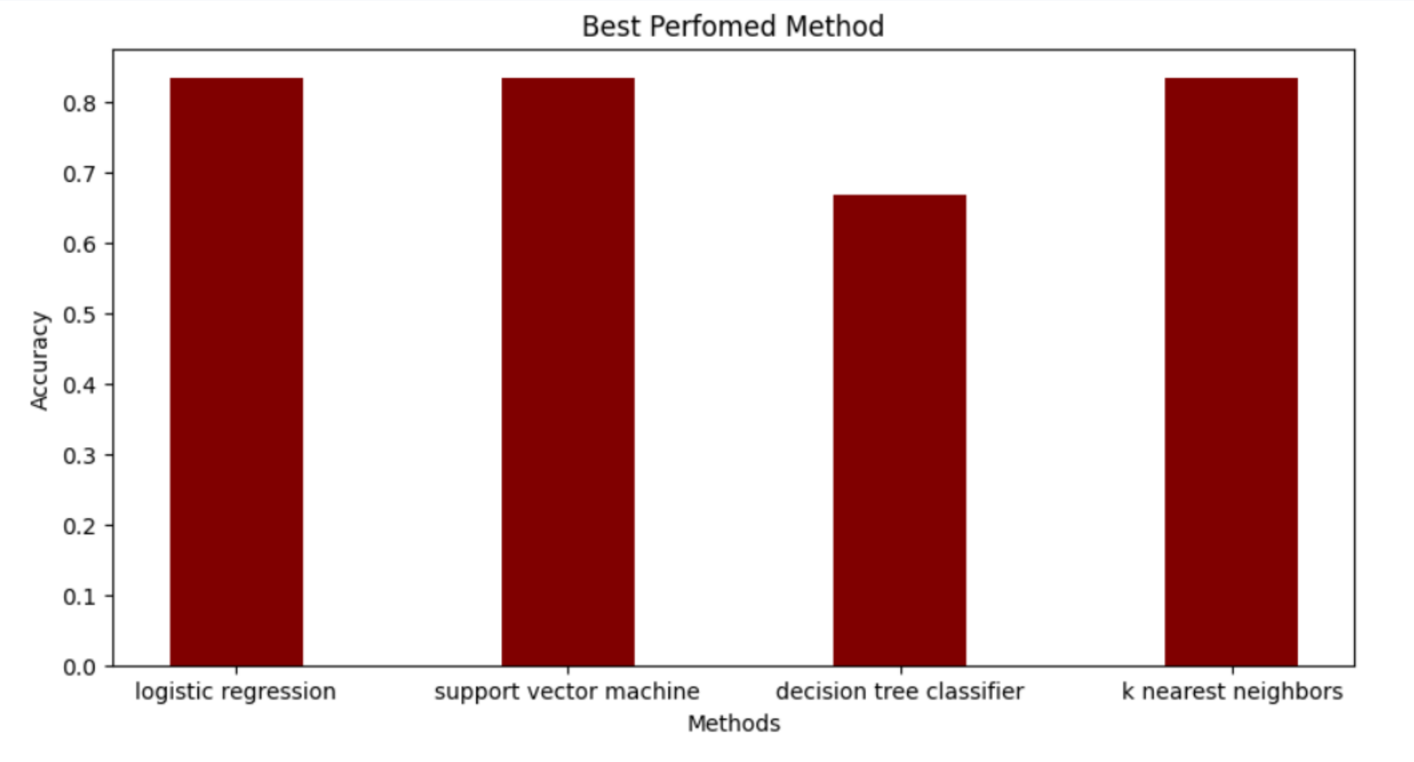




```
import numpy as np import matplotlib.pyplot as plt fig = plt.figure(figsize = (10, 5)) # creating the bar plot plt.bar(methods, accu, color ='maroon', width = 0.4) plt.xlabel("Methods") plt.ylabel("Accuracy") plt.title(
```

Predictive Analysis (Classification)

Four classification models including Linear Regression, K Nearest Neighbors, Decision Tree models, and Support Vector Machine, were built and compared.



Results

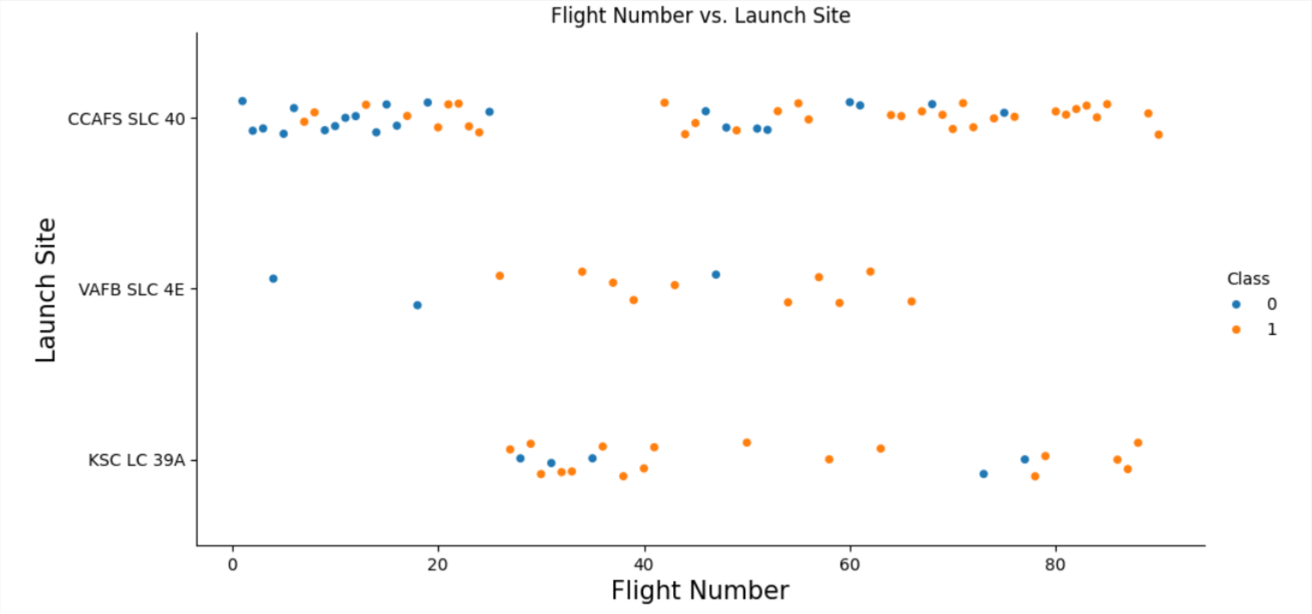
- The first successful ground pad landing took place on December 22, 2015.
- The success rates for SpaceX launches increased over time (2013-2020).
- The location of launch appears to be a significant contributing factor to the success of missions
- KSC LC-39A has the most successful launches compared to other sites.
- Low weighted payloads perform better than the heavier payloads.
- ES-L1, GEO, HEO, and SSO orbits achieved the highest success rate.
- For heavy payloads, the rates of success are higher for VLO and ISS orbits.
- Decision Tree model is the best in terms of prediction accuracy for this dataset.
- For additional information, refer to slides 18 through 45.

Section 2

Insights drawn from EDA

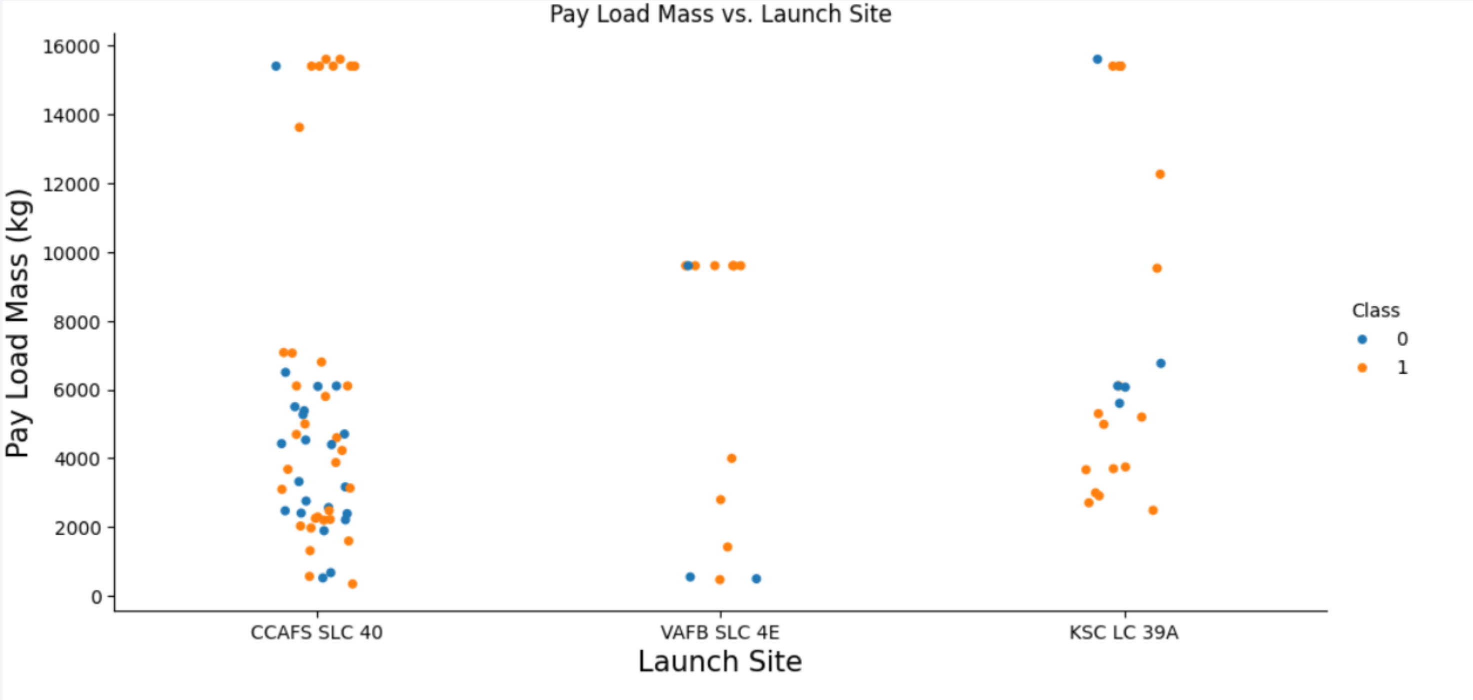
Flight Number vs. Launch Site

- Most launches took off from CCAFS SLC 40 launch site
- •The initial launches were mostly carried out from the CCAFS SLC 40 launch site
- •The majority of recent launches from CCAFS SLC 40 resulted in success
- •The success rates for all launch sites improved over time.



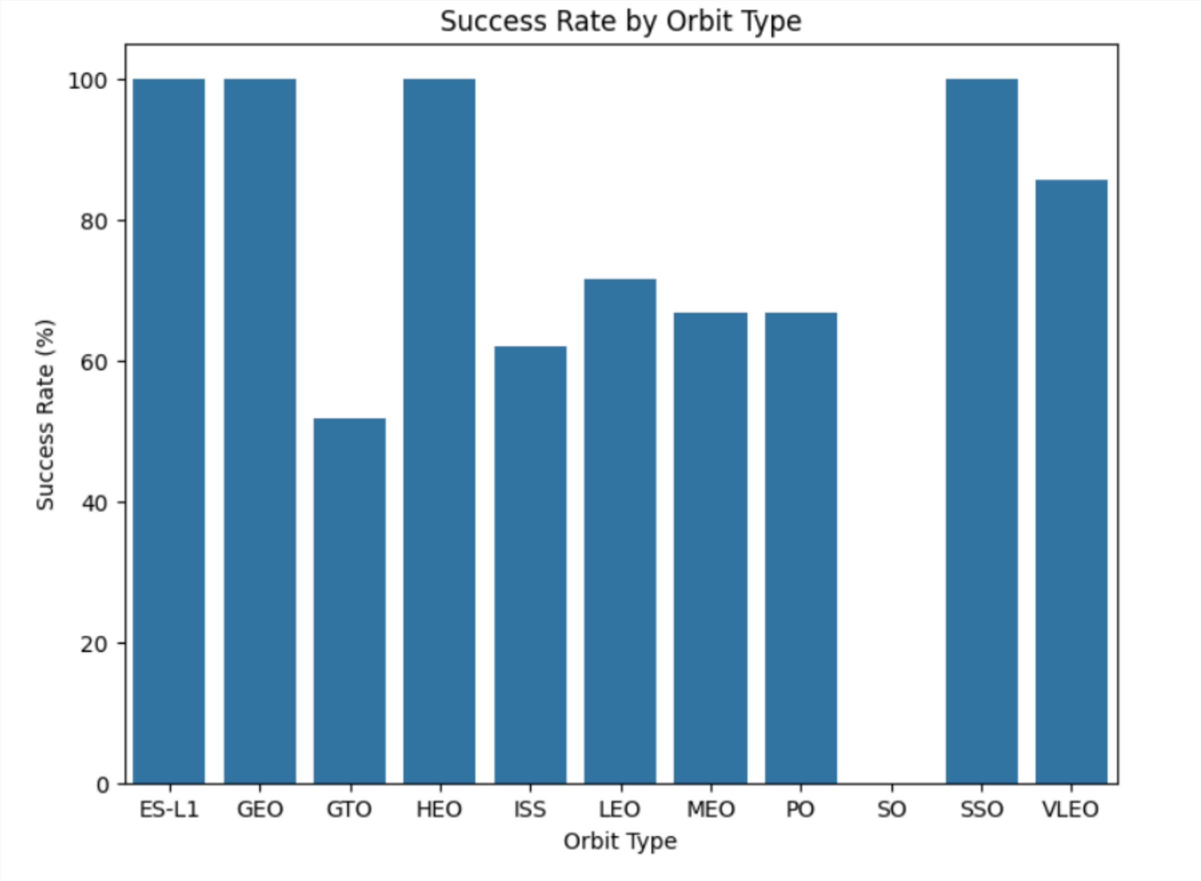
Payload vs. Launch Site

- VAFB SLC 4E launch site conducts launches with lower payloads (zero launches for >10000 kg)
- •CCAFS SLC 40 hosts a higher number of launches involving both higher and lower payloads.
- •Most payloads weighing over 9000 kg have achieved successful outcomes.



Success Rate vs. Orbit Type

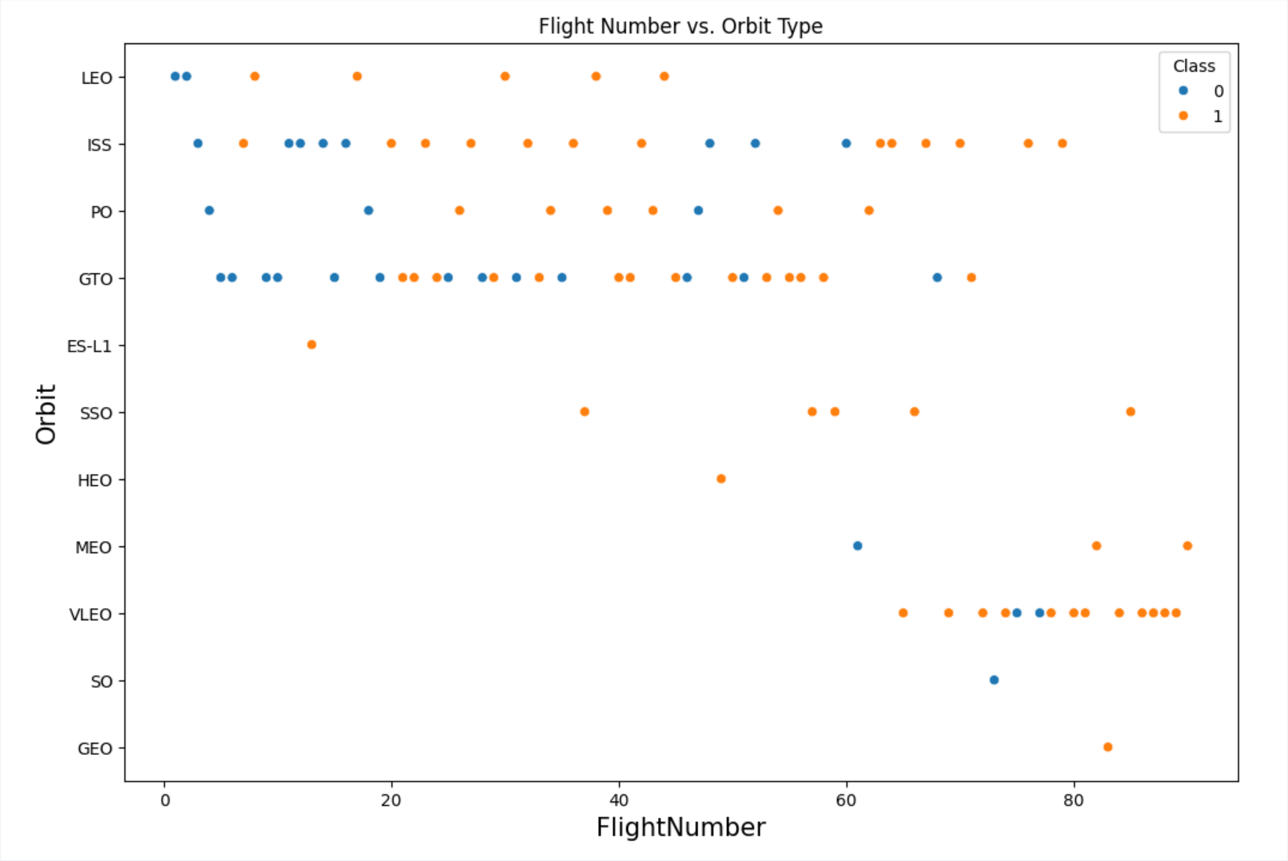
- ES-L1, GEO, HEO, and SSO orbits achieved the highest success rate at 100%,
- followed by VLEO with a success rate >80%, and LEO with a success rate >70%.



Flight Number vs. Orbit Type

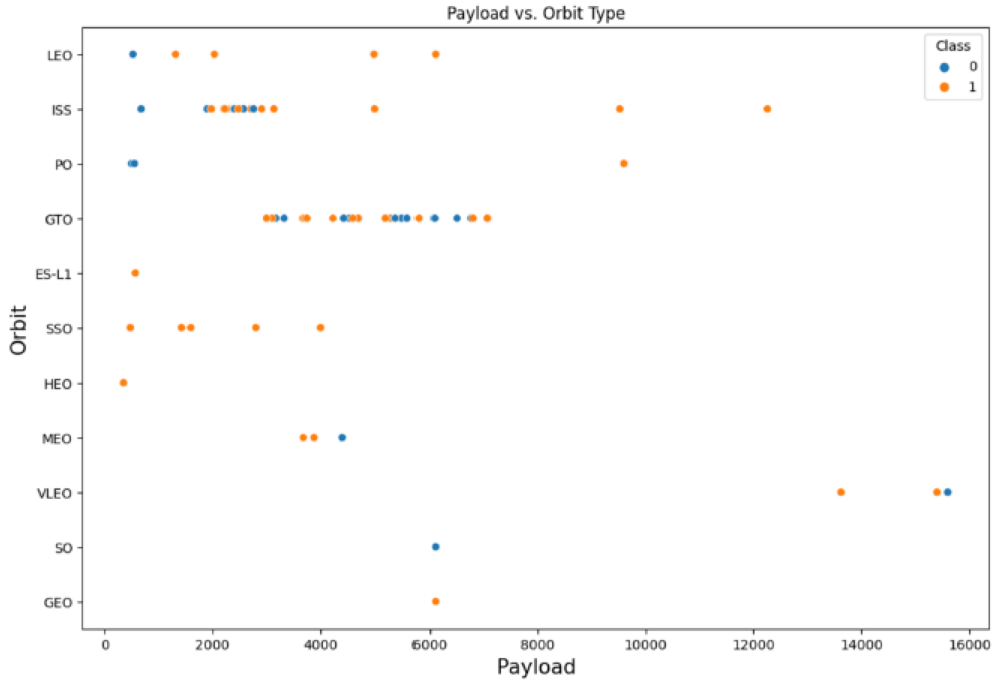
- In the recent years, there has been a transition towards launching missions into Very Low Earth Orbits (VLEO) with a significantly high rate of success.

- While the GTO orbit experiences a low success rate, there appears to be no discernible relationship between flight number and the rate of success in this orbit.



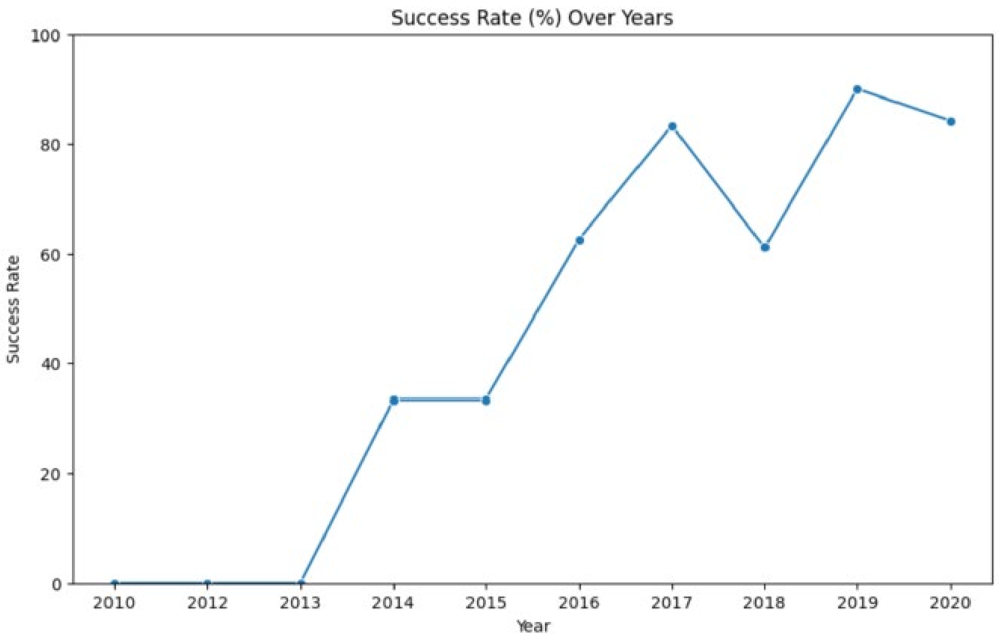
Payload vs. Orbit Type

- For heavy payloads, the rates of success are higher for VLO and ISS orbits.
- •In the case of GTO, there seems no apparent relationship between payload and the rate of success.



Launch Success Yearly Trend

- The rate of success has seen a notable rise since 2013 and continued to rise until 2020, possibly attributed to technological advancements.
- •The first three years (2010-2013) seem to have been a phase focused on fine-tuning and technological enhancement.



%sql SELECT DISTINCT Launch_Site FROM SPACEXTABLE;

All Launch Site Names

- Find the names of the unique launch sites

CCAFS LC-40

VAFB SLC-4E

KSC LC-39A

CCAFS SLC-40

- Present your query result with a short explanation here
- %sql SELECT DISTINCT Launch_Site FROM SPACEXTABLE;

Launch Site Names Begin with 'CCA'

- Find 5 records where launch sites begin with `CCA`

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

- Present your query result with a short explanation here

%sql SELECT * FROM SPACEXTABLE WHERE Launch_Site LIKE 'CCA%' LIMIT 5;

Total Payload Mass

- Calculate the total payload carried by boosters from NASA
- Present your query result with a short explanation here

Task 3

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

[15]:
```
%sql SELECT SUM(PAYLOAD_MASS__KG_) AS Total_payload_mass FROM SPACEXTABLE WHERE Customer = 'NASA (CRS)';
```

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

[15]:

Total_payload_mass
45596

Average Payload Mass by F9 v1.1

- Calculate the average payload mass carried by booster version F9 v1.1
- Present your query result with a short explanation here

Task 4

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

[16]:
```
%sql SELECT AVG(PAYLOAD_MASS__KG_) AS Average_Payload_Mass FROM SPACEXTABLE WHERE Booster_Version = 'F9 v1.1';
```

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

[16]:

Average_Payload_Mass
2928.4

First Successful Ground Landing Date

- Find the dates of the first successful landing outcome on ground pad
- Present your query result with a short explanation here

Task 5

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

Hint:Use min function

[17]:
```
%sql SELECT MIN(DATE) AS First_successful_landing_gound FROM SPACEXTABLE WHERE Landing_Outcome = 'Success (ground pad)';
```

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

[17]:

First_successful_landing_gound
2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

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

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

[18]:
```
%%sql SELECT Booster_Version
FROM SPACEXTABLE
WHERE Landing_Outcome = 'Success (drone ship)' AND PAYLOAD_MASS__KG_>4000 AND PAYLOAD_MASS__KG_<6000;
```

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

[18]:

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

Total Number of Successful and Failure Mission Outcomes

- Calculate the total number of successful and failure mission outcomes

Task 7

List the total number of successful and failure mission outcomes

[19]:
```
%%sql SELECT Mission_Outcome, COUNT(*) AS Total_count
FROM SPACEXTABLE
GROUP BY Mission_Outcome;
```

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

[19]:

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

Boosters Carried Maximum Payload

- List the names of the booster which have carried the maximum payload mass

Task 8

List all the booster_versions that have carried the maximum payload mass. Use a subquery.

[20]:
```
%%sql SELECT Booster_Version
FROM SPACEXTABLE
WHERE PAYLOAD_MASS__KG_ = (
    SELECT MAX(PAYLOAD_MASS__KG_)
    FROM SPACEXTABLE
);
```

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

[20]:

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

2015 Launch Records

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

[14]:
```
%%sql SELECT
 CASE
        WHEN Date like '%-01-%' THEN 'January'
        WHEN Date like '%-02-%' THEN 'February'
        WHEN Date like '%-03-%' THEN 'March'
        WHEN Date like '%-04-%' THEN 'April'
        WHEN Date like '%-05-%' THEN 'May'
        WHEN Date like '%-06-%' THEN 'June'
        WHEN Date like '%-07-%' THEN 'July'
        WHEN Date like '%-08-%' THEN 'August'
        WHEN Date like '%-09-%' THEN 'September'
        WHEN Date like '%-10-%' THEN 'October'
        WHEN Date like '%-11-%' THEN 'November'
        WHEN Date like '%-12-%' THEN 'December'
    END AS Month,
    Landing_Outcome AS Failure_Landing_Outcome,
    Booster_Version,
    Launch_Site
FROM SPACEXTABLE
WHERE Date like'%2015%' AND Landing_Outcome LIKE 'Failure (drone ship)';
```

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

[14]:

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

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

The tally of landing results (including Failure on drone ships or Success on ground pads) occurring between June 4, 2010, and March 20, 2017, is presented in a descending order:

•As shown, the highest count among the landing outcomes is attributed to "No attempt", totaling 10. This is followed by "Success (ground pad) ", "Success (drone ship) ", and "Failure (drone ship) " each occurring 5 times, while "Controlled (ocean)" is observed 3 times

```
[15]: %%sql SELECT Landing_Outcome, COUNT(*) AS Outcome_Count
      FROM SPACEXTABLE
      WHERE Date BETWEEN '2010-06-04' AND '2017-03-20'
      GROUP BY Landing_Outcome
      ORDER BY Outcome_Count DESC;
```

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

[15]:

Landing_Outcome	Outcome_Count
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

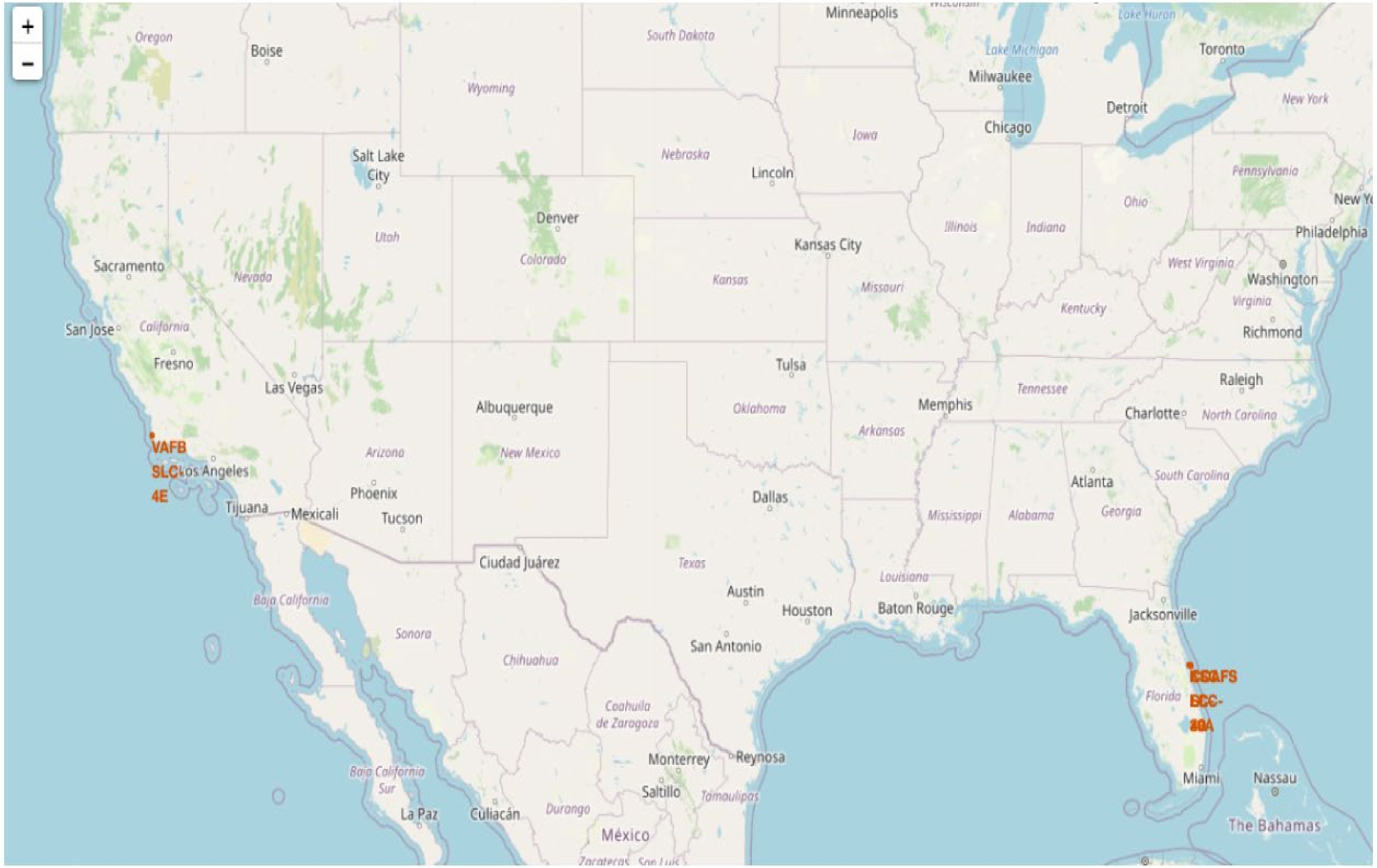
Section 3

Launch Sites Proximities Analysis

Folium Map – Launch Site Locations

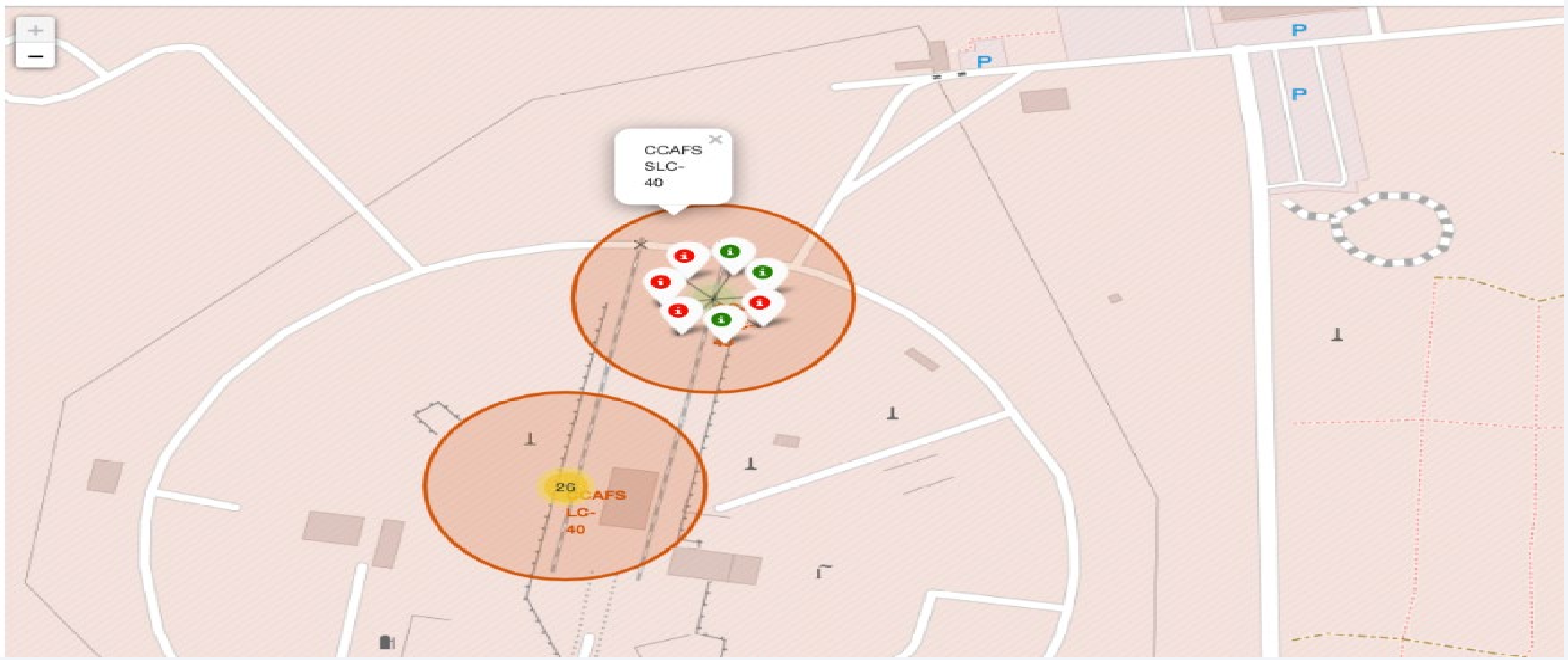
VAFB SLC-4E is situated near the western coastline, while KSC LC-39A, CCAFLC-40, and CCAFSLC-40 are positioned along the eastern coastline.

•Upon zooming in, it seems both CCAFLC-40 and CCAFSLC-40 are situated in very close proximity to each other.



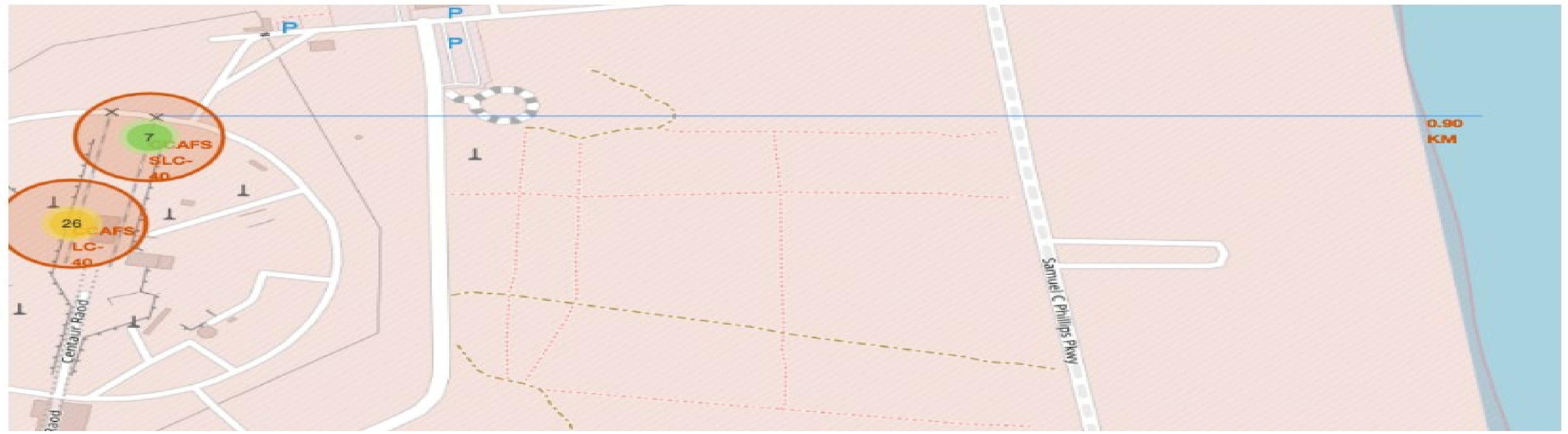
Folium Map Screenshot 2 – Success/Failed Launches

Upon zooming in, we can see the success (green) and failure (red) marks for each site.
Out of 13 launches, KSC LC-39A has achieved the highest success rate with 10 successful missions (10/13=76.9%)



Folium Map –Proximity of Launch Sites to Other Areas

These figures show a Polyline between CCAFSLC-40to the selected coastline point, etc.



Section 4

Build a Dashboard with Plotly Dash

Dashboard Screen Shot 1 –Total SuccessLaunches By All Sites

The location of launch appears to be a significant contributing factor to the success of missions. KSC LC-39A has the most successful launches compared to other sites.

Dashboard Screen Shot 2 –Launch Site With Highest Launch Success Ratio

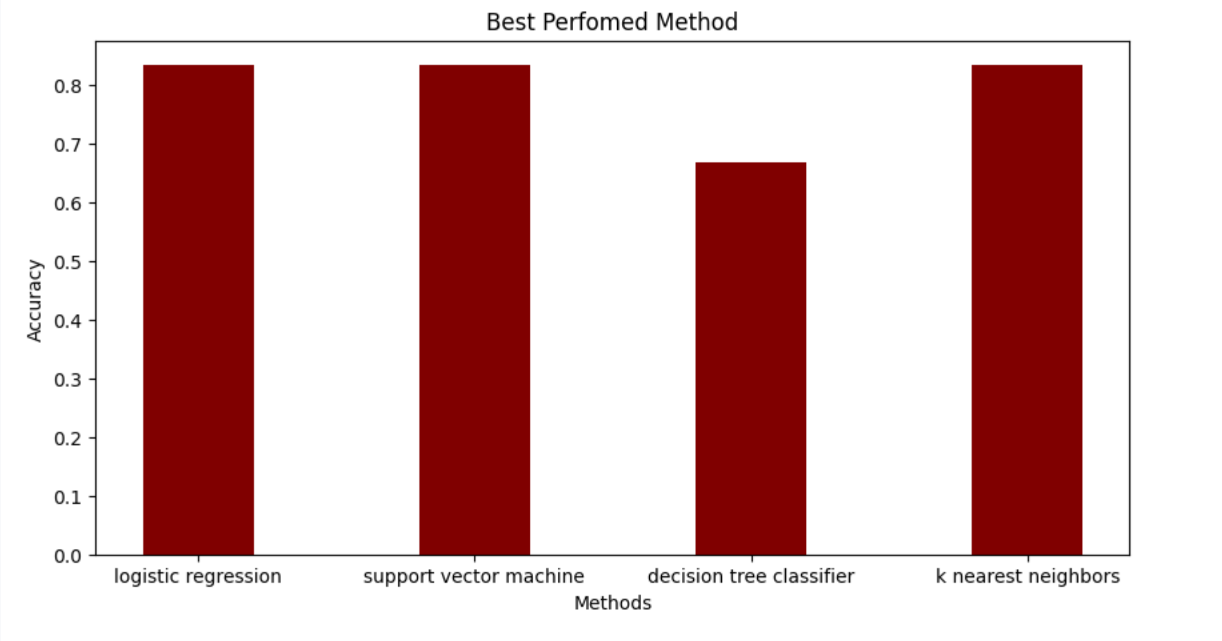
Using the dropdown menu on the dashboard allows for viewing single site launches.
At KSC LC-39A, 76.9% of the launches resulted in success, while 23.1% experienced failure.

Section 5

Predictive Analysis (Classification)

Classification Accuracy

Decision Tree model achieved the highest accuracy at 88 %, while the SVM performs the best in terms of Area Under the Curve at 0.96 (not shown).



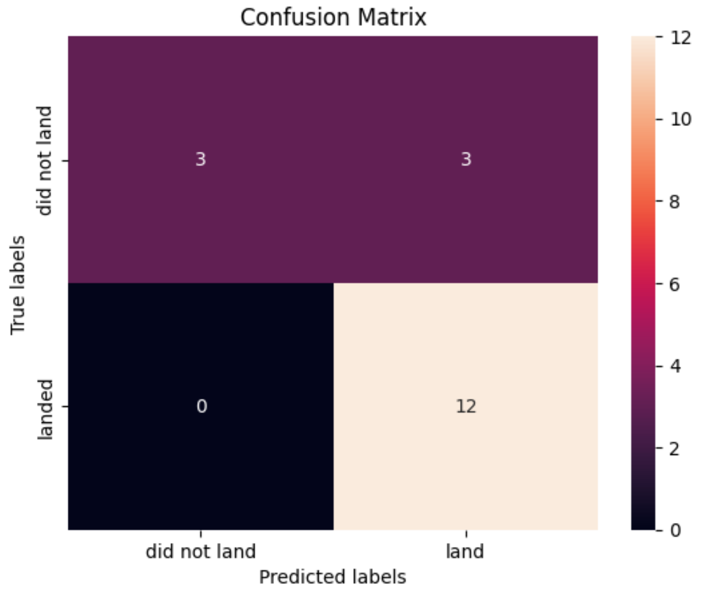
Confusion Matrix

```
accu=[]
methods=[]
accu.append(logreg_cv.score(X_test,Y_test))
methods.append('logistic regression')
logreg_cv.score(X_test,Y_test)
```

```
0.8333333333333334
```

Lets look at the confusion matrix:

```
yhat=logreg_cv.predict(X_test)
plot_confusion_matrix(Y_test,yhat)
```

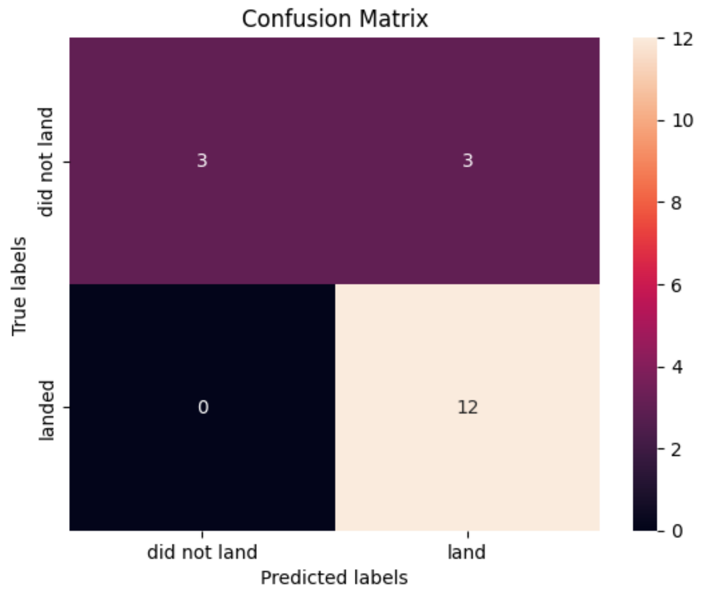



```
accu.append(svm_cv.score(X_test,Y_test))
methods.append('support vector machine')
svm_cv.score(X_test,Y_test)
```

```
0.8333333333333334
```

We can plot the confusion matrix

```
yhat=svm_cv.predict(X_test)
plot_confusion_matrix(Y_test,yhat)
```

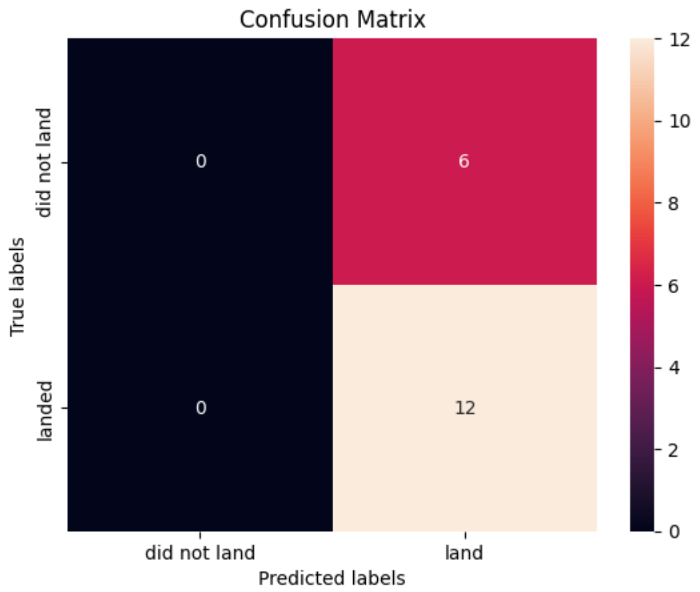



```
[31]: accu.append(tree_cv.score(X_test,Y_test))
      methods.append('decision tree classifier')
      tree_cv.score(X_test,Y_test)
```

```
[31]: 0.6666666666666666
```

We can plot the confusion matrix

```
[32]: yhat = tree_cv.predict(X_test)
      plot_confusion_matrix(Y_test,yhat)
```



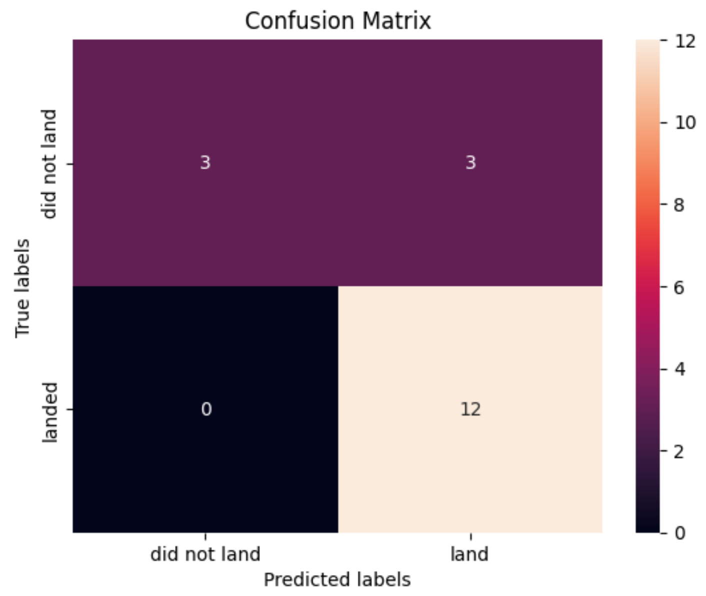
Confusion Matrix

```
[36]: accu.append(knn_cv.score(X_test,Y_test))
      methods.append('k nearest neighbors')
      knn_cv.score(X_test,Y_test)
```

```
[36]: 0.8333333333333334
```

We can plot the confusion matrix

```
[37]: yhat = knn_cv.predict(X_test)
      plot_confusion_matrix(Y_test,yhat)
```



All four models can distinguish between the different classes.

•However, the major problem for LR, SVM, and KNN is False Positives (n=3)

•Decision Tree has the least False Positive (n=1)

Conclusions

- The success rates for SpaceX launches increased over time (2013-2020).
- •KSC LC-39A has the most successful launches compared to other sites.
- •Low weighted payloads perform better than the heavier payloads.
- •ES-L1, GEO, HEO, and SSO orbits achieved the highest success rate.
- •For heavy payloads, the rates of success are higher for VLO and ISS orbits.
- •Decision Tree models are the best in terms of prediction accuracy for this dataset.
- Through the utilization of available data and comprehensive analysis, rocket companies can pinpoint the most effective techniques for diminishing launch expenses. This approach averts the risk of losing clients and ensures they remain competitive in the market

Appendix

- url for the code
- https://github.com/harvindersainiibm/Applied-Data-Science-Capstone

Thank you!