IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Ishan Bhaway

9th Aug 2022

Outline

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

Executive Summary

- Summary of methodologies
  - Data collection through API and Webscraping
  - Data Wrangling
  - Exploratory data analysis using:
    - SQL
    - Data Visualization
  - Interactive Visual Analytics using Folium
  - Machine Learning Prediction
- Summary of all results
  - Exploratory Data Analysis results
  - Interactive analytics dashboard screenshots
  - Predictive Analysis outcome

Introduction

- Project background and context
  - In this capstone, we will predict if the Falcon 9 first stage will land successfully. SpaceX advertises Falcon 9 rocket launches on its website with a cost of 62 million dollars; other providers cost upward of 165 million dollars each, much of the savings is because SpaceX can reuse the first stage. Therefore if we can determine if the first stage will land, we can determine the cost of a launch. This information can be used if an alternate company wants to bid against SpaceX for a rocket launch.
- Problems you want to find answers
  - Identify the factors that will influence the landing
  - Relationship between each of the factors and the kind of influence they have on the landing outcome
  - Predicting the best factors/conditions needed for the better probability of successful landing

Section 1

Methodology

Methodology

Executive Summary

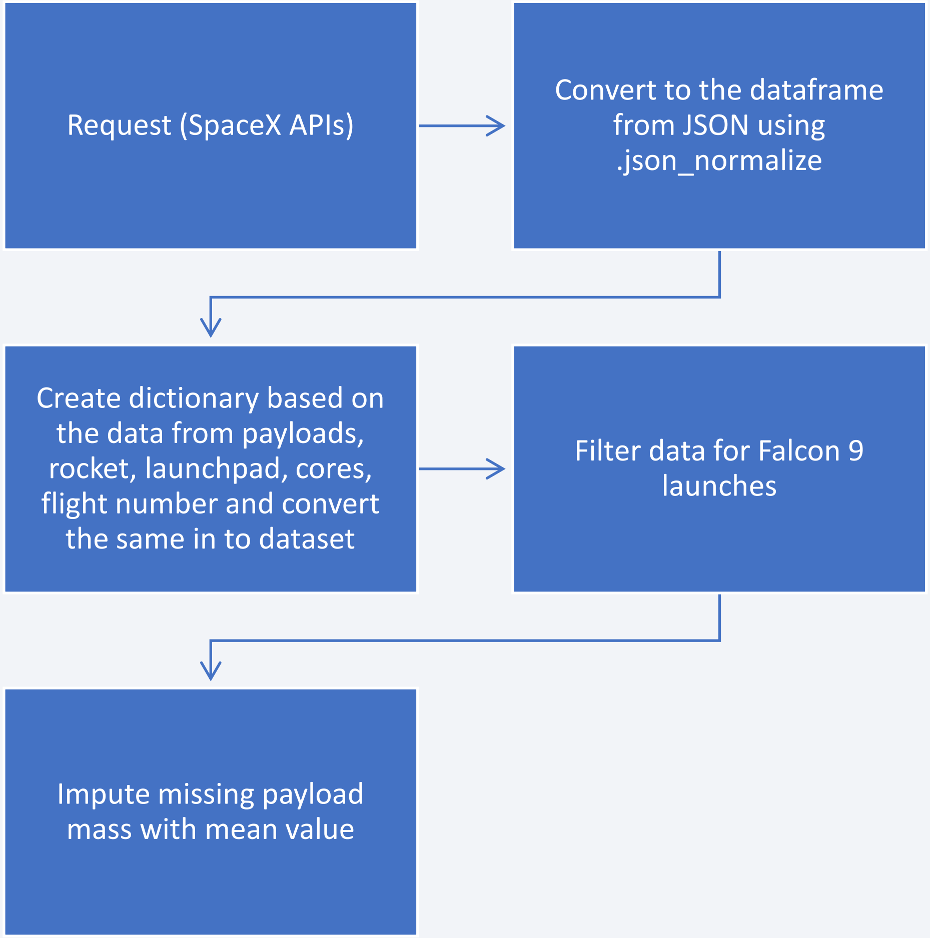
- Data collection methodology:
  - Data was collected using SpaceX REST API and webscrapping from Wikipedia
- Perform data wrangling
  - One hot encoding was performed on the categorical features present in data collected
  - Restricting to only required/necessary columns
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - How to build, tune, evaluate classification models

Data Collection

- Describe how data sets were collected.
- Data sets are collected from Space X API for:
    - https://api.spacexdata.com/v4/rockets/ - Rockets
    - https://api.spacexdata.com/v4/launchpads/ - LaunchPads
    - https://api.spacexdata.com/v4/payloads/ - Payloads
    - https://api.spacexdata.com/v4/cores/ - Cores

- You need to present your data collection process use key phrases and flowcharts

Data Collection – SpaceX API




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

```python
response = requests.get(spacex_url)
```

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

```python
# Lets take a subset of our dataframe keeping only the features we want and the flight number, and date_utc.
data = data[['rocket', 'payloads', 'launchpad', 'cores', 'flight_number', 'date_utc']]

# We will remove rows with multiple cores because those are falcon rockets with 2 extra rocket boosters and rows that have multipl
data = data[data['cores'].map(len)==1]
data = data[data['payloads'].map(len)==1]

# Since payloads and cores are lists of size 1 we will also extract the single value in the list and replace the feature.
data['cores'] = data['cores'].map(lambda x : x[0])
data['payloads'] = data['payloads'].map(lambda x : x[0])

# We also want to convert the date_utc to a datetime datatype and then extracting the date leaving the time
data['date'] = pd.to_datetime(data['date_utc']).dt.date

# Using the date we will restrict the dates of the launches
data = data[data['date'] <= datetime.date(2020, 11, 13)]
```

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

Then, we need to create a Pandas data frame from the dictionary launch_dict.

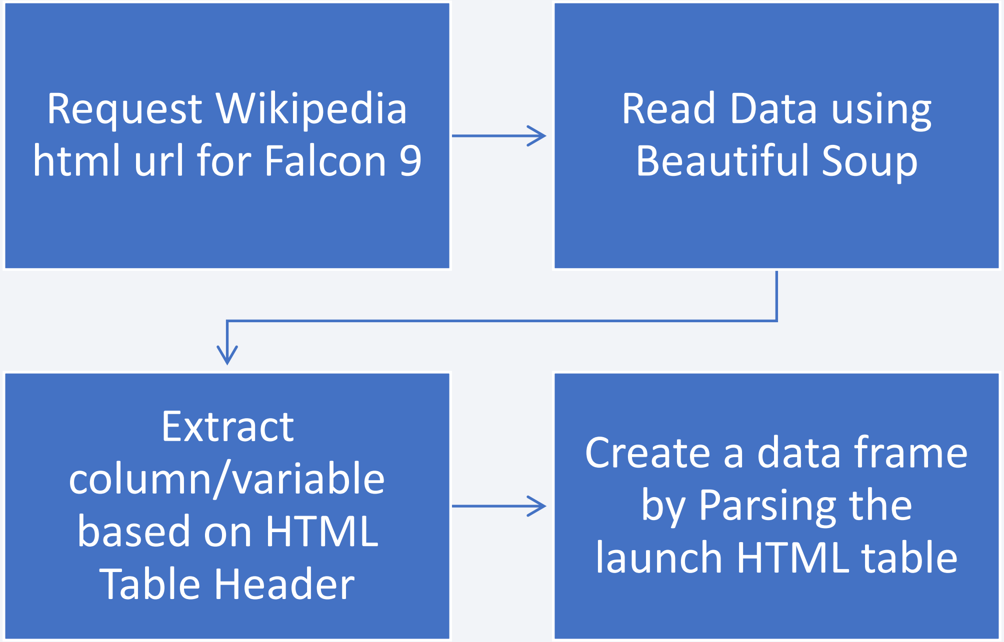
```python
# Create a data from launch_dict
l_data=pd.DataFrame(launch_dict)
```

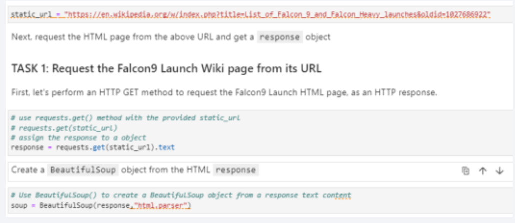
```python
# Hint data['BoosterVersion']!='Falcon 1'
data_falcon9=l_data[l_data['BoosterVersion']=='Falcon 9']
```

```python
# Calculate the mean value of PayloadMass column
apload=data_falcon9.PayloadMass.mean()
# Replace the np.nan values with its mean value
data_falcon9["PayloadMass"].replace(np.nan, apload, inplace=True)
```

URL : https://github.com/ishanbhaway/testrepo/blob/923f92a569e50bdbf02e61f12a9d01d1e5ec9b6c/jupyter-labs-spacex-data-collection-api.ipynb

Data Collection - Scraping





```
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

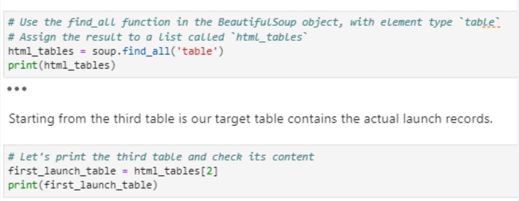
TASK 1: Request the Falcon9 Launch Wiki page from its URL

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

```
# use requests.get() method with the provided static_url
# requests.get(static_url)
# assign the response to a object
response = requests.get(static_url).text
```

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

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



```
# Use the find_all function in the BeautifulSoup object, with element type `table`
# Assign the result to a list called `html_tables`
html_tables = soup.find_all('table')
print(html_tables)
```

•••

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

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

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

```
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])
            launch_dict['Flight No.'].append(datatimelist)

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

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

            # Booster version
            # TODO: Append the bv into launch_dict with key `Version Booster`
            bv=booster_version(row[1])
            if not(bv):
                bv=row[1].a.string
            print(bv)
            launch_dict['Version Booster'].append(bv)

            # Launch Site
            # TODO: Append the bv into launch_dict with key `Launch Site`
            launch_site = row[2].a.string
            #print(launch_site)
            launch_dict['Launch site'].append(launch_site)

            # Payload
            # TODO: Append the payload into launch_dict with key `Payload`
            payload = row[3].a.string
            #print(payload)
            launch_dict['Payload'].append(payload)

            # Payload Mass
            # TODO: Append the payload_mass into launch_dict with key `Payload mass`
            payload_mass = get_mass(row[4])
            #print(payload)
            launch_dict['Payload mass'].append(payload_mass)

            # Orbit
            # TODO: Append the orbit into launch_dict with key `Orbit`
            orbit = row[5].a.string
            #print(orbit)
            launch_dict['Orbit'].append(orbit)

            # Customer
            # TODO: Append the customer into launch_dict with key `Customer`
```

URL : https://github.com/ishanbhaway/testrepo/blob/923f92a569e50bdbf02e61f12a9d01d1e5ec9b6c/jupyter-labs-webscraping.ipynb

Data Wrangling

- Converting Outcomes into Training labels:
  - 1 – successful landing – True Ocean, True ASDS
  - 0 – failure landing – False RLTS, False ASDS

Calculate the number of Launches → **Calculate the number and occurrence of each orbit** → **Calculate the number and occurence of mission outcome per orbit type** → **Create a landing outcome label from Outcome column**

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

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

```
# Apply value_counts on Orbit column
df['Orbit'].value_counts()
```

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

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

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

```
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:

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

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

```
df[df['LaunchSite']=='CCAFS SLC 40'].shape
```

```
(55, 17)
```

```
df.to_csv("dataset_part_2.csv", index=False)
```

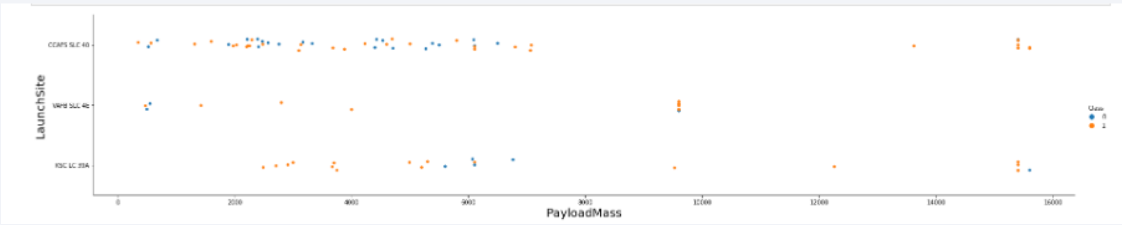
```
# landing_class = 0 if bad_outcome

# landing_class = 1 otherwise
landing_class = df['Outcome'].apply(lambda landing_class: 0 if landing_class in bad_outcomes else 1)
```

URL : https://github.com/ishanbhaway/testrepo/blob/923f92a569e50bdbf02e61f12a9d01d1e5ec9b6c/labs-jupyter-spacex-Data%20wrangling.ipynb

EDA with Data Visualization

- Scatter Graph: *To plot relationship between variables*
  - Flight Number and Payload Mass
  - Flight Number and Launch Site
  - Payload and Launch Site
  - FlightNumber and Orbit type
  - Payload and Orbit type

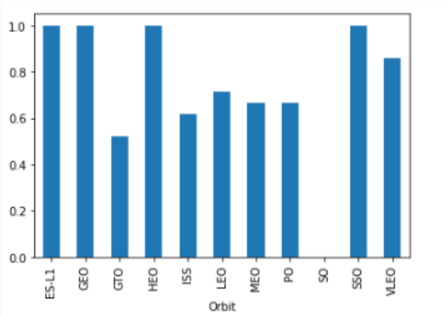


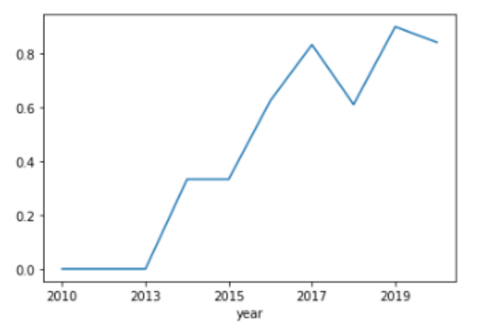
Bar Graph: *To plot relationship between Categorical variable and corresponding values against a particular variable*

- Success rate of each orbit type

Line Graph: To plot trend of a variable for a given case – can be used for comparison

- Success yearly trend





URL : https://github.com/ishanbhaway/testrepo/blob/923f92a569e50bdbf02e61f12a9d01d1e5ec9b6c/jupyter-labs-eda-dataviz.ipynb

EDA with SQL

- Display the names of the unique launch sites in the space mission
- Display 5 records where launch sites begin with the string 'CCA'
- Display the total payload mass carried by boosters launched by NASA (CRS)
- Display average payload mass carried by booster version F9 v1.1
- List the date when the first succesful landing outcome in ground pad was acheived.
- List the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000
- List the total number of successful and failure mission outcomes
- List the names of the booster_versions which have carried the maximum payload mass. Use a subquery
- List the records which will display the month names, failure landing_outcomes in drone ship ,booster versions, launch_site for the months in year 2015.
- Rank the count of successful landing_outcomes between the date 04-06-2010 and 20-03-2017 in descending order.

URL : https://github.com/ishanbhaway/testrepo/blob/923f92a569e50bdbf02e61f12a9d01d1e5ec9b6c/jupyter-labs-eda-sql-coursera_sqllite.ipynb

Build an Interactive Map with Folium

- Markers and circle indicates launch sites – like NASA Johnson Space Center
- Grouping of data points in a cluster considering they refer to same coordinates
    - Green indicates successful and Red indicates Failure in landing
- Line markers indicates between launch site and respective locations – coast, railways etc
- The markers assist in understanding the data in reference to live maps

URL :
https://github.com/ishanbhaway/testrepo/blob/923f92a569e50bdbf02e61f12a9d01d1e5ec9b6c/lab_jupyter_launch_site_location.ipynb

Build a Dashboard with Plotly Dash

- Graphs and plots –
  - Successful Launches by Site
  - Payload and Success by sites
- This allows quick assessment of the relationship between payload, launch sites and successful launches

URL :
https://github.com/ishanbhaway/testrepo/blob/923f92a569e50bdbf02e61f12a9d01d1e5ec9b6c/spacex_dash_app.py

Predictive Analysis (Classification)

Data Preparation	Model Generation	Model Evaluation	Model Comparison
• Load dataset • Data transformation • Standardize dataset • Splitting the datasets into train and test sets	• Configure the parameters for GridSearchCV • Apply the respective parameters on the Machine learning algorithms • Train the models with train datasetds	• Identify the best parameters for the respective model • Assess each model's accuracy based on the test dataset • Generate Confusion matrix basis the same	• Compare the models accuracy for selection of the model applicable

URL : https://github.com/ishanbhaway/testrepo/blob/a439e423c99a2e5603ec37e30456a30459c8c51e/SpaceX_Machine%20Learning%20Prediction_Part_5.ipynb

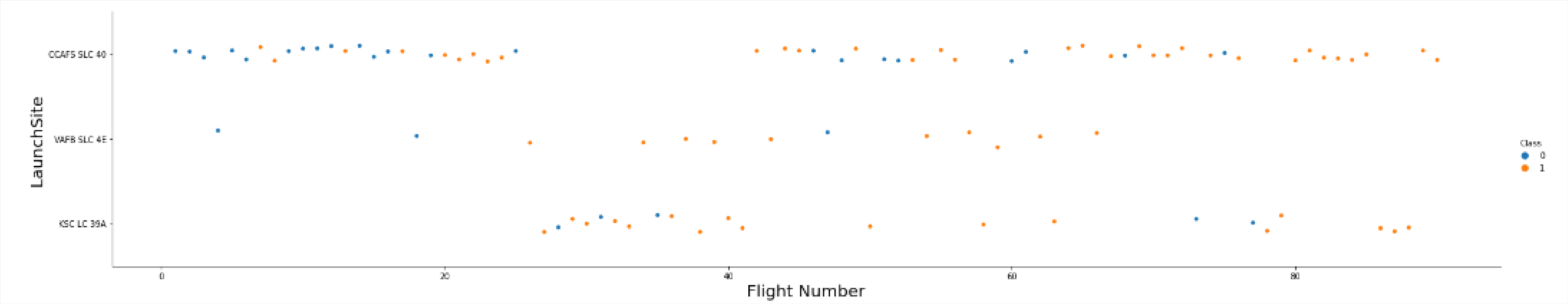
Results

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

Section 2

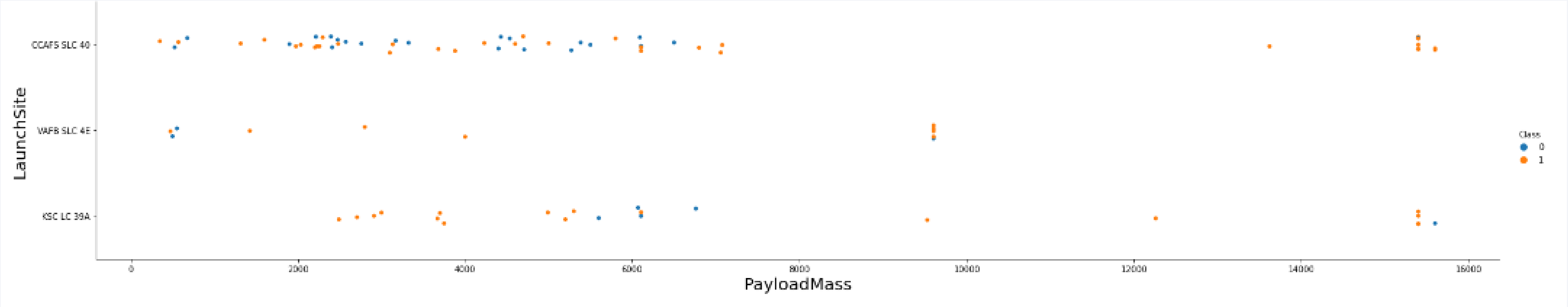
Insights drawn from EDA

Flight Number vs. Launch Site



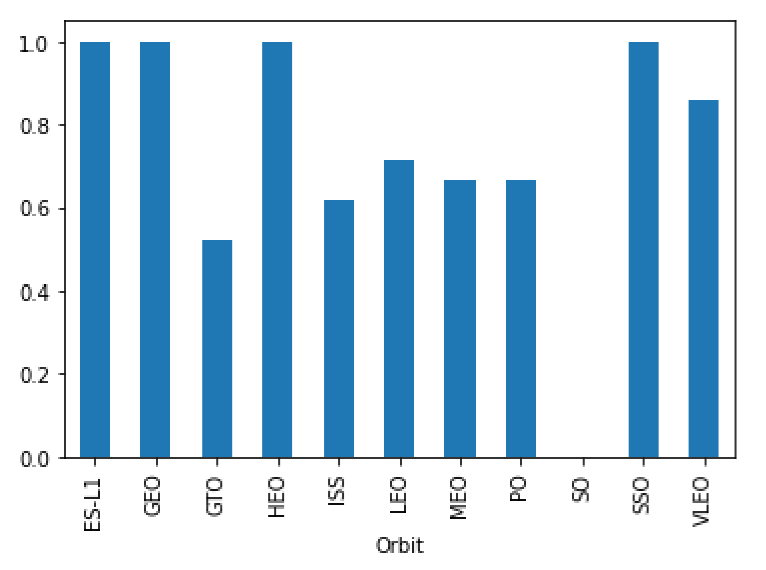
- Failures tend to decrease over time and Success increase over time
- CCAF5 SLC 40 has most number of launches also holds most number of success

Payload vs. Launch Site



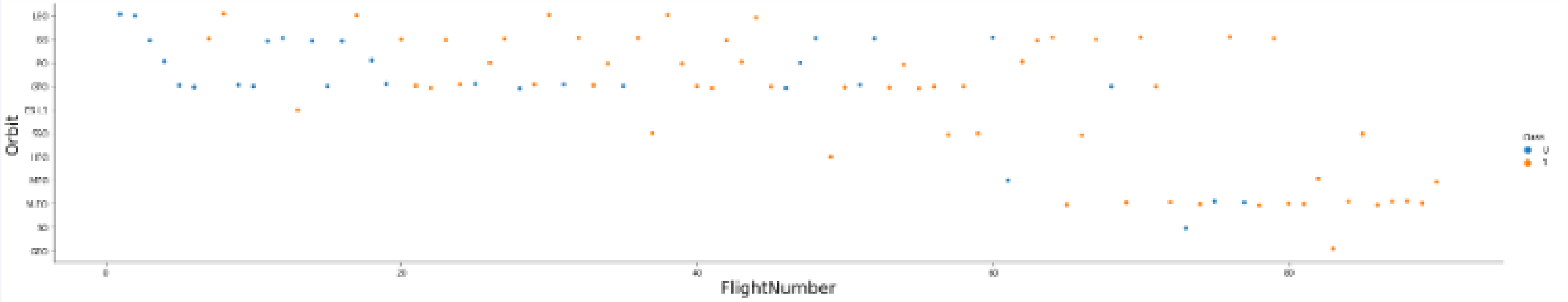
- The Success rate is observed to be higher in cases where the Payload is higher

Success Rate vs. Orbit Type



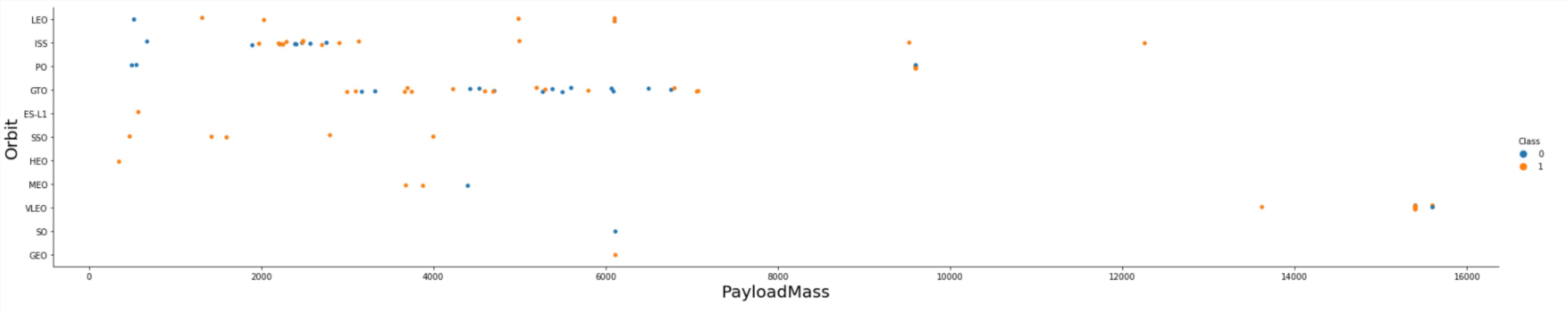
- While SSO, HEO, GEO and ES-L1 have 100% success rate; it is to be noted that ES-L1, GEO and HEO have done only one launch

Flight Number vs. Orbit Type



- Success rate has improved for all orbits over time relatively
- VLEO orbit even though recent can be observed can be considered with higher success rate

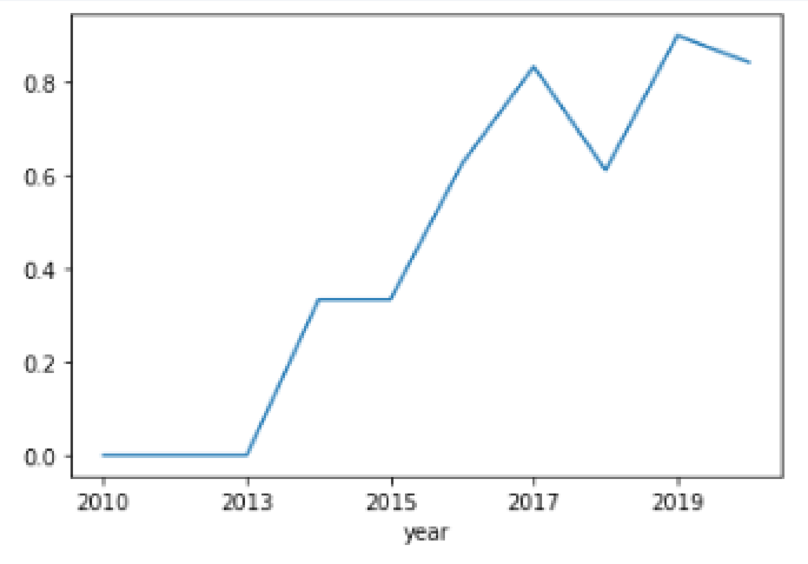
Payload vs. Orbit Type



- There is no clear pattern associated with orbit type and Payload
- Except ISS, most of the orbit type are concentrated towards in specific range of payload

Launch Success Yearly Trend

- Success rate significantly rose in 2013 and started stabilizing in year 2017



All Launch Site Names

- Use Distinct to get unique Launch sites

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

```
%sql select distinct(launch_site) from spacextbl
```

```
 * 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'

- Use limit 5 for top 5 statements and "CCA%" with WHERE clause to filter the data starting with CCA

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 _Outcome
04-06-2010	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
08-12-2010	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)
22-05-2012	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
08-10-2012	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
01-03-2013	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

Total Payload Mass

- Use Customer as “NASA (CRS)” for payload masses

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

```
%sql select sum(payload_mass__kg_) from spacextbl where customer = 'NASA (CRS)'
```

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

sum(payload_mass__kg_)
45596

Average Payload Mass by F9 v1.1

- Use WHERE query to identify booster version

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

```
%sql select avg(payload_mass__kg_) from spacextbl where Booster_Version = 'F9 v1.1'
```

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

avg(payload_mass__kg_)
2928.4

First Successful Ground Landing Date

- Use MIN() function to get the result

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

Hint:Use min function

```
%sql select min(date) from spacextbl where `Landing _Outcome`  = 'Success (ground pad)'
```

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

min(date)
01-05-2017

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, using BETWEEN clause and other WHERE clauses

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 PAYLOAD_MASS__KG_ BETWEEN 4000 AND 6000 AND `Landing _Outcome` = 'Success (drone ship)';
```

```
 * 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

- Calculate the total number of successful and failure mission outcomes using WHERE clause along with UNION

```
%sql SELECT "Success" as mission, COUNT(*) AS QTY FROM SPACEXTBL \
where mission_outcome like 'Success%' \
union \
SELECT "Failure" as mission, COUNT(*) AS QTY FROM SPACEXTBL \
where mission_outcome like 'Failure%'
```

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

mission	QTY
Failure	1
Success	100

Boosters Carried Maximum Payload

- Use sub query with MAX function to filter data using WHERE clause

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

```
%sql select booster_version from spacextbl \
where payload_mass__kg_ = (select max(payload_mass__kg_) from spacextbl)
```

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

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

2015 Launch Records

- Use SUBSTR to identify the year 2015 based on year position in the respective date format alongwith *Failure* filter

```
%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` like 'Failure%'
```

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

month	Landing _Outcome	Booster_Version	Launch_Site
01	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

- Use the Count upon grouping landing outcomes (such as Failure (drone ship) or Success (ground pad)) between the date 2010-06-04 and 2017-03-20. This is followed by using Dense_Rank in descending order on Count

```
%sql select `Landing _Outcome`, count(*) as freq, dense_rank() over ( order by Count(*) desc) countrank\
from SPACEXTBL WHERE DATE BETWEEN '04-06-2010' AND '20-03-2017' \
GROUP BY `Landing _Outcome`
```

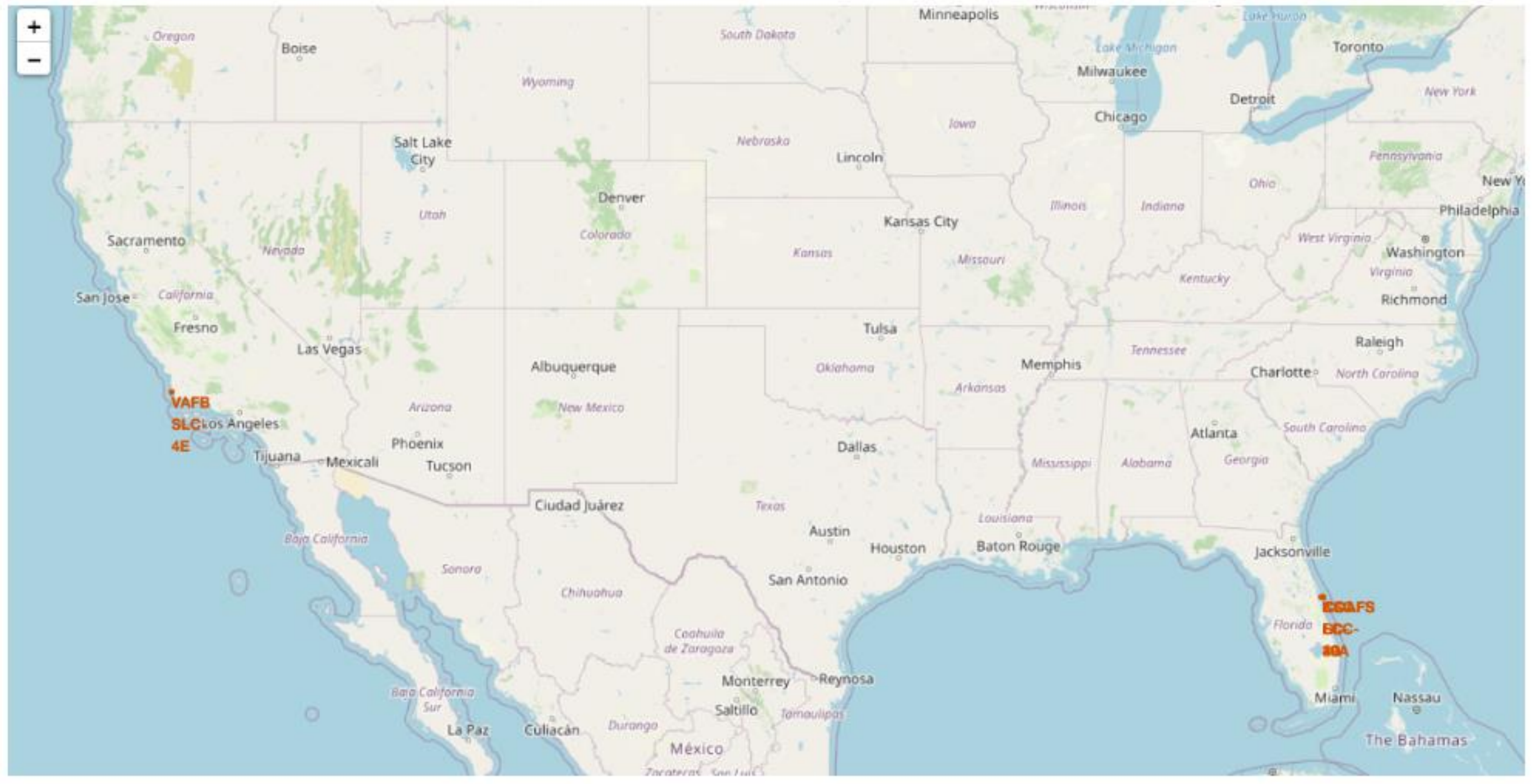
```
 * sqlite:///my_data1.db
Done.
```

Landing _Outcome	freq	countrank
Success	20	1
No attempt	10	2
Success (drone ship)	8	3
Success (ground pad)	6	4
Failure (drone ship)	4	5
Failure	3	6
Controlled (ocean)	3	6
Failure (parachute)	2	7
No attempt	1	8

Section 3

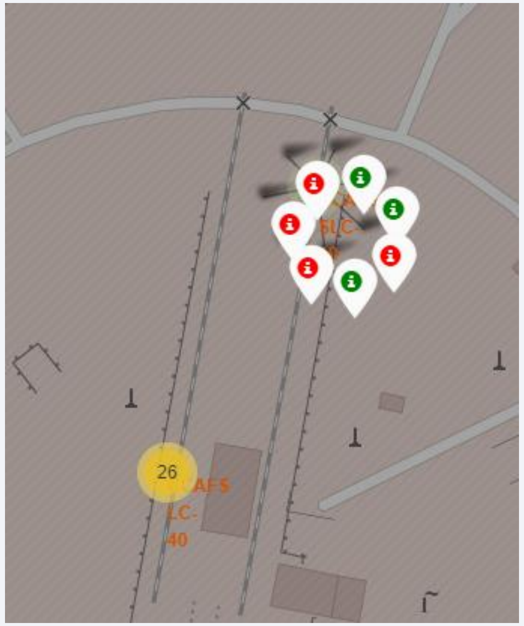
Launch Sites Proximities Analysis

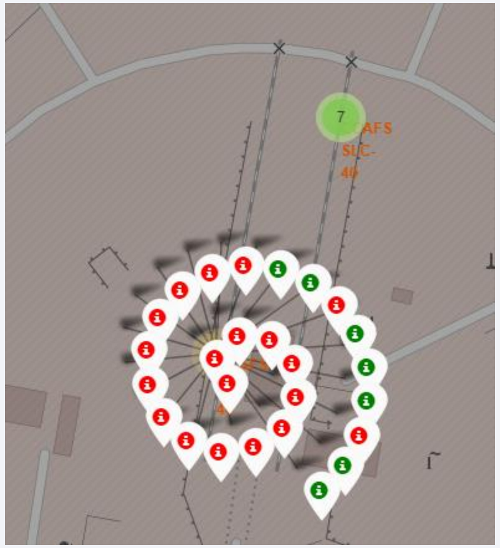
Location of launch Sites

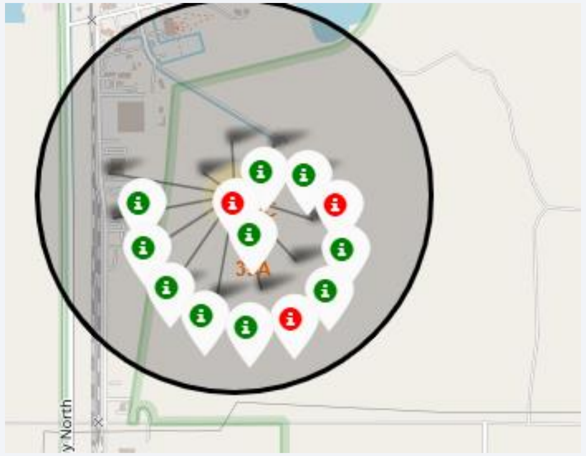


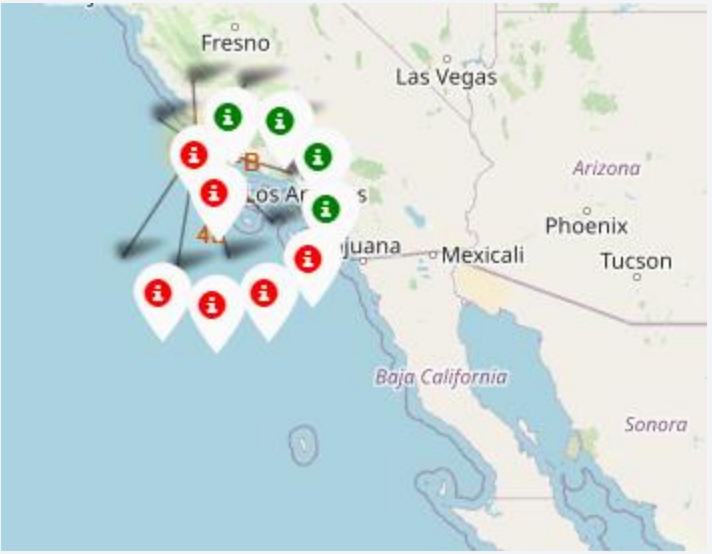
- All are within US South coastal region indicating:
  - the closeness to equator being targeted to get most benefit of earth's rotation
  - safety in instance of failure the rocket to be crash landed in sea to avoid US asset and human damage

Launch sites Markers with color labels



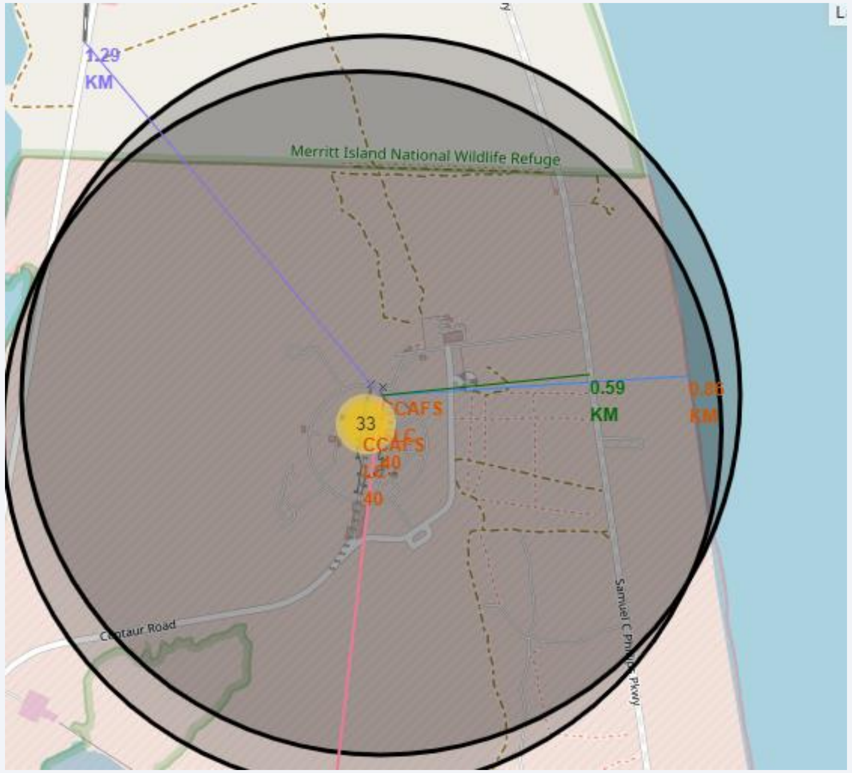


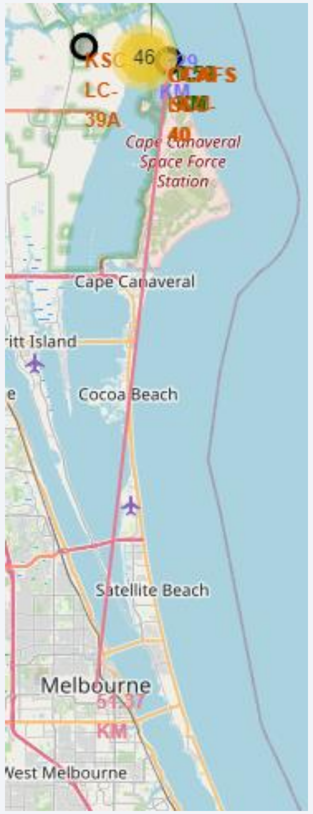




- Green indicates successful launches
- Red indicates failures

Launch sites distance from Landmarks





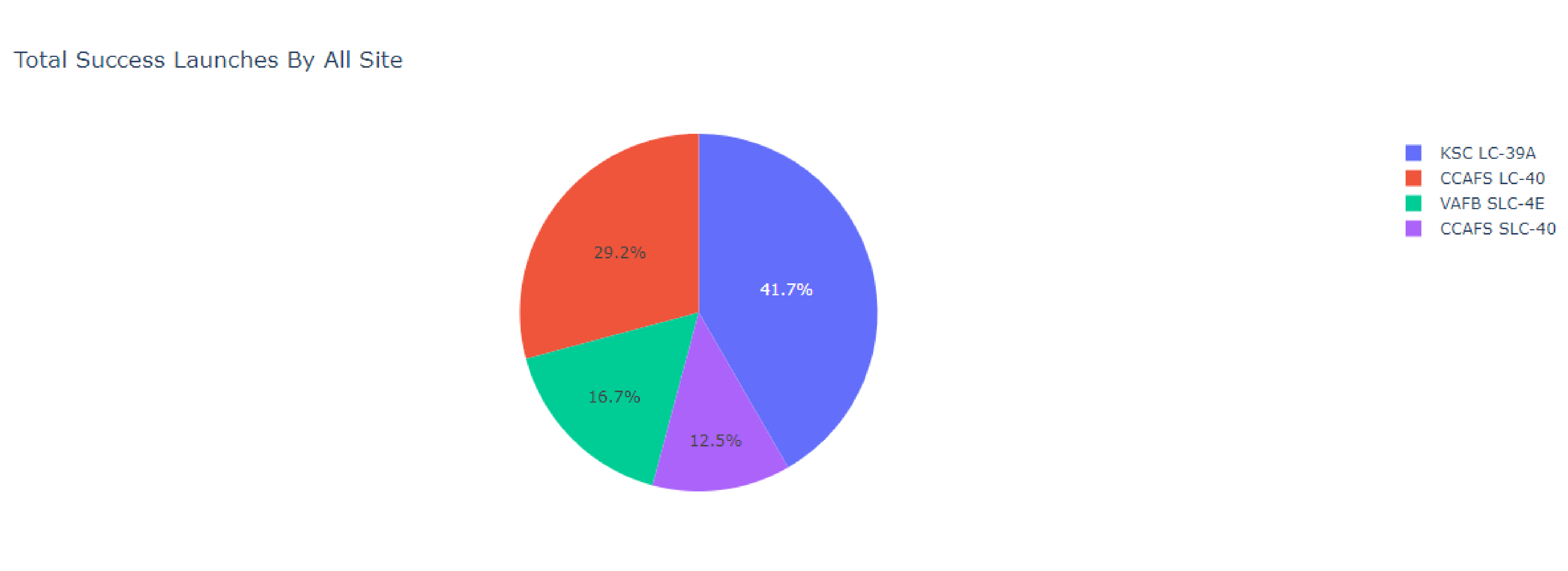
- Launch sites are extremely close to coastline as compared to any other major infrastructure like railways and even major cities

Section 4

Build a Dashboard with Plotly Dash

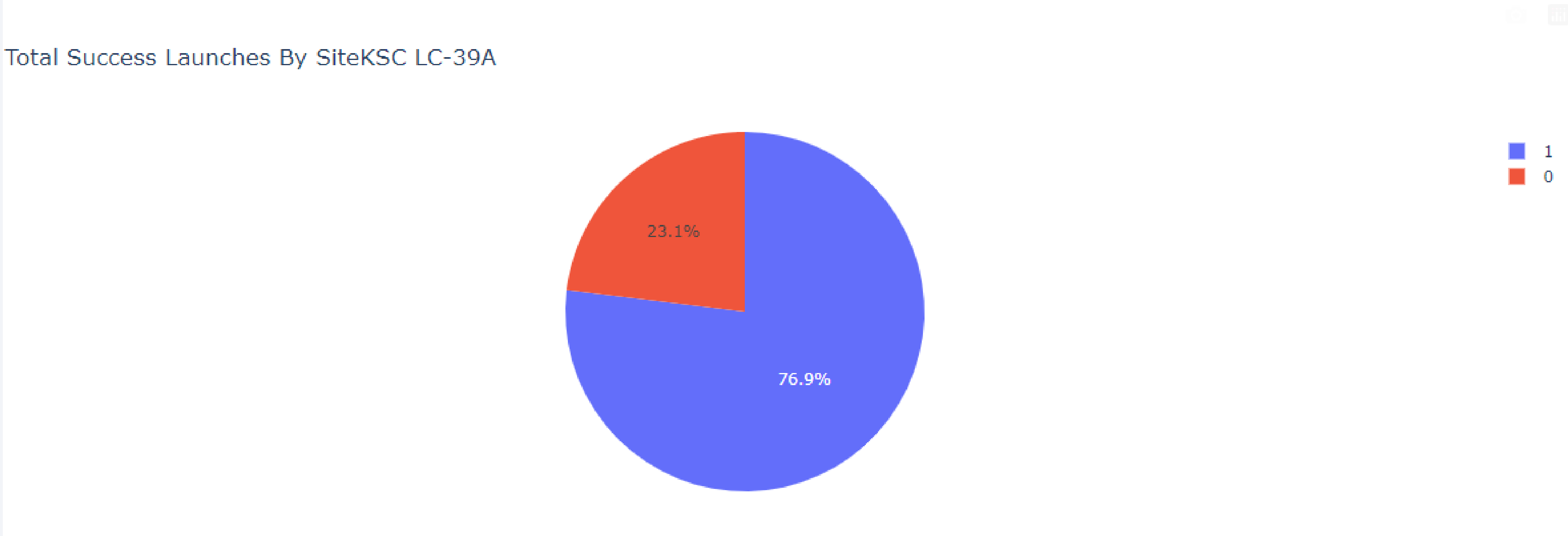
Success percentage by each sites

- KSC LC-39A is the most successful among all sites



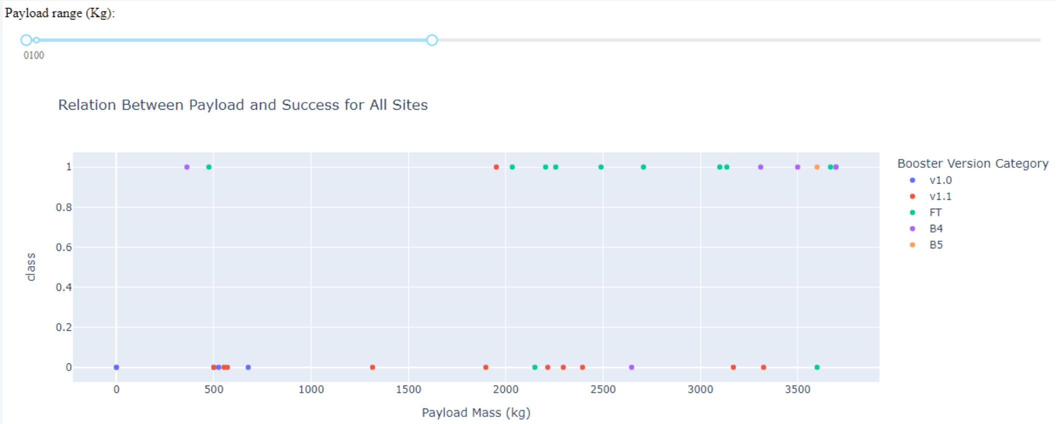
Pie Chart showing the most successful Launch Site: KSC LC-39A

- KSC LC-39A with success rate of 76.9% and 23.1% failure rate



Payload vs. Launch Outcome scatter plots

- Low Payload launches were most successful
- FT booster version category is most successful

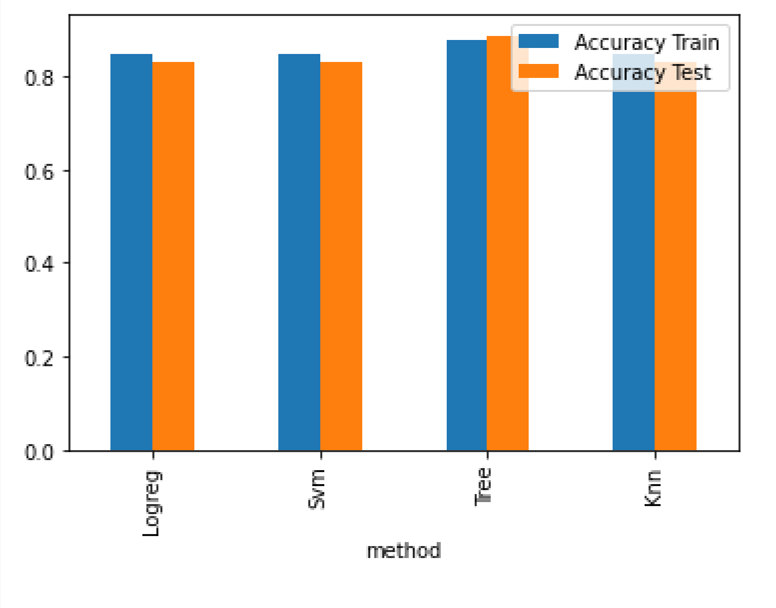


Section 5

Predictive Analysis (Classification)

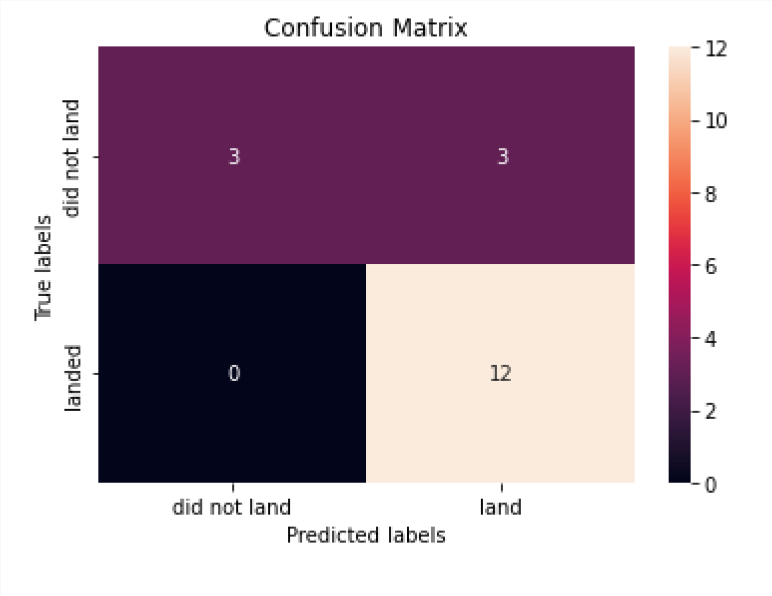
Classification Accuracy

- Decision tree has the highest classification accuracy



Confusion Matrix – Decision Tree

- While best in terms of accuracy, considering the situation in matter – 3 false positives – failures predicted as successful landing



Conclusions

- Success rates increased over time
- SSO and VLEO to be key opportunity considering high success rates
- Success rates increased significantly from 2013 and right now stabilizing
- Launch sites are closer to coastline and equator
- KSC LC-39A most successful launches
- Decision Tree is best classifier

Appendix

- Libraries used:
  - Numpy
  - Pandas
  - Folium
  - Seaborn
  - Dash
  - Plotly
  - SQLite
  - BeautifulSoup
  - Re
  - Requests
  - Sklearn

Thank you!